Build Software

Discovery Phase I-Report

Version 1.0

Created on – 8th November 2024

Contents

[1. Executive Summary 4](#_Toc182045303)

[2. About this phase 5](#_Toc182045304)

[3. Phase I - Scope and Deliverables 6](#_Toc182045305)

[4. Build-Software’s goals for this phase I 7](#_Toc182045306)

[5. Projects / Sub Projects / Modules taken up in this Phase I 8](#_Toc182045307)

[6. Status of the elements of the scope 10](#_Toc182045308)

[7. Overview of Information Received 14](#_Toc182045309)

[7.1. Build Software - C# 14](#_Toc182045310)

[7.2. ECQ – VB 14](#_Toc182045311)

[7.3. Start Up Projects – Build.Net & Emma.Net (VB) 15](#_Toc182045312)

[7.4. Database Objects 16](#_Toc182045313)

[7.5. Source code metrics 16](#_Toc182045314)

[8. Inventory of the Documents Received 18](#_Toc182045315)

[9. Overview of the Meetings Held 22](#_Toc182045316)

[10. Major Discoveries 23](#_Toc182045317)

[10.1. Business Flow 23](#_Toc182045318)

[10.2. Technology Stack 23](#_Toc182045319)

[10.3. SQL server version and support 24](#_Toc182045320)

[10.4. Development Tool version release & extended support information 24](#_Toc182045321)

[10.5. Integration of VB.NET Code in C# for Key Business Functions 24](#_Toc182045322)

[10.6. Impact of Unused VB.NET Code on Code Quality: 25](#_Toc182045323)

[10.6.1. Recommendations for Code Quality and Future Development: 25](#_Toc182045324)

[10.7. Documentation 25](#_Toc182045325)

[10.7.1. Size of existing documentation 25](#_Toc182045326)

[10.7.2. Multiple versions with different departments 26](#_Toc182045327)

[10.7.3. Hard to find any relevant documentation 26](#_Toc182045328)

[10.7.4. Risk in Documentation 26](#_Toc182045329)

[10.7.5. Recommendation 26](#_Toc182045330)

[11. Observations & Findings 28](#_Toc182045331)

[11.1. Source Code and DB Connectivity (our observations) 28](#_Toc182045332)

[11.2. Coding Standards (Define the quality of the coding standard) 28](#_Toc182045333)

[11.3. Unit testing 28](#_Toc182045334)

[11.4. Test Cases 28](#_Toc182045335)

[11.5. Regression Testing 28](#_Toc182045336)

[11.6. Knowledge resides in the heads of the current maintenance team 29](#_Toc182045337)

[11.7. Hard coding 29](#_Toc182045338)

[11.8. Challenges of Using Non-English Loggers and Error Messages 29](#_Toc182045339)

[11.9. Challenges of having VB.NET integrating with C# in a Project 29](#_Toc182045340)

[10.10. Recommendations from Observation 29](#_Toc182045341)

[12. Conclusions & Suggestions on further project approach 31](#_Toc182045342)

[12.1. Documentation Setup 31](#_Toc182045343)

[12.2. Analysis & Document Calculation & Invoice Module 31](#_Toc182045344)

[12.3. Existing Documentation to catalogue 31](#_Toc182045345)

[12.4. Document the store procedure in DB 31](#_Toc182045346)

[12.5. Identify all the unused files(VB code) 31](#_Toc182045347)

[11.6. Document other modules in VB in sequence of priorities 32](#_Toc182045348)

# Executive Summary

The primary objective of this phase is to identify and document areas for improvement in the existing documentation, which will serve as the foundation for future development. The report outlines the scope, goals, and findings from the discovery phase, focusing on the modernization of Build Software’s 20-year-old on-premises system. The scope includes the collection and review of existing documentation, such as technical, functional, API, cloud, and coding standards, as well as the analysis of source code and databases. It also involves identifying high-level risks and mitigation strategies and preparing a detailed proposal for Phase II: Documentation. The deliverables for this phase are an observation and understanding document, along with financial, effort, and technical estimations for Phase II.

Build Software aims to modernize its software to address challenges like limited cloud and mobile capabilities, complex integrations, and lengthy deployment times. Key focus areas include transitioning to a cloud-based architecture, developing a mobile app, enhancing core features such as security, multi-language support, and multi-currency capabilities, standardizing integrations, and streamlining processes. The report covers several projects and modules, including Build Software (core system written in C#), Build.net (developed in VB and integrated with Build Software), ECQ (a basic application in VB, partially in use), and a Web API for third-party communication.

Major discoveries include a structured business flow for managing project and order workflows, from tender to financial reporting. The technology stack comprises C# and VB.NET programming languages, Entity Framework ORM, SQL Server 2014 database, and ASP.NET Web API. The report details support timelines for various SQL Server versions and discusses the integration of VB.NET and C# code, highlighting challenges and recommendations for a unified codebase. The impact of unused VB.NET code on code quality is noted, with recommendations for code cleanup and transition to a unified C# codebase.

The current state of documentation is problematic, with over 1 TB of poorly labelled and outdated documents, multiple versions across departments, and significant inefficiencies. Recommendations include centralizing documentation in a single repository with version control, organizing documents into categories like “Executed Ideas,” “Pending Ideas,” and “Declined Ideas,” and implementing structured logging and enhanced error handling.

Observations and findings highlight the use of Entity Framework and Auto Mapper for data management, the lack of inline comments in the source code, limited unit testing, and no comprehensive regression testing. Critical business logic information resides with individual team members, posing risks to project stability. Challenges include non-English loggers and error messages, and integration issues between VB.NET and C#. Recommendations focus on improving documentation, code quality, testing, and error handling to enhance the maintainability, efficiency, and stability of the software.

In conclusion, the report suggests setting up a hierarchical documentation structure, analysing and documenting high-priority modules like calculation and invoicing, cleaning up unused VB.NET code, transitioning to a unified C# codebase, and developing detailed test cases for end-to-end regression testing. These steps aim to provide a clear, organized framework for future development and ensure the long-term sustainability of Build Software.

# About this phase-I

Discovery Phase 1: 14-Oct-2024 to 11-Nov-2024

Team Members and their focus

* Rajeev:
* Thomas
* Gopinath
* Sailesh
* Koteeshwar
* Geetha
* Kalpana
* Siddharth

Time spent -

Major parts

# Phase I - Scope and Deliverables

The primary objectives of pre-analysis and discovery phase, INSOFT will perform and interact with Build Software for the following steps to discover, establish and document the current requirement to document all listed applications.

* Collection, review and random audit of all existing documentation like
  1. Technical Architecture documentation of the project and subprojects
  2. Functional documentation of the project and subprojects
  3. API Documentation of the projects
  4. Cloud and Premises application documentation and features
  5. Coding standards of the same
  6. Database relationship documents of the same
  7. Connectivity documentation of all subprojects to all of their relations
  8. System flow diagram(s) of the current application (if any)
  9. Installation, implementation and user guide(s) for all applications (if any)

The above said documents should be provided by Build Software to INSOFT as soon as possible after the kick-off of the assignment. They will be the base for this pre-analysis Phase.

* Collection and analysis of the source code files and databases of the listed applications.
* Collection of the details of all external/internal application components, such as reporting tools and services which directly connect to the database.
* Analysis of current architecture and dependencies from the documentation.
* Analysis on external factors in the application environment.
* Identification of high-level risks and their possible mitigation strategies.
* Analysis of the used tools, identified in all projects. Their licensing cost, authorization to access DB, Login credentials to be provided to INSOFT (keeping in mind privacy rules).
* A detailed financial and technical proposal for Phase II: Documentation, which includes an effort and delivery schedule will be prepared and shared with Build Software.

The following are the deliverables planned for this Phase

* An observation- and understanding document of the project and its subparts.
* A financial- effort- and technical estimation and proposal for Phase II (Documentation)

# Build-Software’s goals for this phase I

Team Insoft had a meeting with Team BLDSW about this topic specifically on Oct 21, 2024.

Team BLDSW mentioned their following key challenges in the AS-IS situation

* The On-premises nature of the software, while many users are looking for cloud-based solutions.
* Lack of mobile-friendly interfaces for construction industry users who often work outside the office.
* Long deployment times for updates and hotfixes (e.g., 4 hours for a recent hotfix).
* Difficulty in tailoring the software for different customer needs due to its monolithic structure.
* Issues with integrations, particularly with accounting software and logistics systems.
* Limited translation capabilities, causing difficulties for users working in different languages.
* Lack of multi-currency support, limiting international business capabilities.

The company’s goal is to undertake a significant modernization of its 20-year-old on-premises Build software, with the aim of addressing key challenges, including limited cloud and mobile capabilities, complex integration issues, and lengthy deployment times. Build-Software’s core functionality should be accessible to both the web app and the mobile devices, and flexibility to support diverse customer needs.

Build-Software wants to focus on:

1. **Cloud and Mobile access**: Transitioning to cloud-based architecture and developing a mobile app for field accessibility.
2. **Modular Structure**: Splitting functionalities into core and non-core modules, supporting specialized licensing and customizations.
3. **Enhanced Core Features**: Strengthening security (adopting NIST standards and role-based access), multi-language support, multi-currency capabilities, and construction-specific functionalities.
4. **Standardized Integrations**: Developing standardized APIs and simplifying connections to external systems like accounting and logistics software.
5. **Streamlined Processes**: Improving documentation, release management, and communication; implementing automated testing to reduce hotfix deployment time.

These strategies leverage a three-layer architecture, new DevExpress screens, and API products, ultimately aiming for a flexible, future-ready ERP solution.

Team Insoft wanted to narrow down the scope of this Phase I, based upon a well defined goal, set by BLDSW. However, since the conveyed goals are very broad and does not have a clear definition yet, Team Insoft elected to also maintain a broad scope, without detail, to offer a way ahead in any further direction, to be chosen by BLDSW leadership.

# Projects / Sub Projects / Modules taken up in this Phase I

The objective of the following projects is to support civil engineering companies in optimizing their business processes. It helps to win tenders, gain higher returns and more satisfied final customers.

**Project Parts**:

A diagram of a diagram

Description automatically generated

A yellow background with grey text

Description automatically generated

Build Software is a comprehensive solution for managing construction and project workflows from start to finish, designed to streamline processes, improve accuracy, and enhance profitability. It enables efficient importing of architectural files and supports extensive budget calculations, making it easier to manage project finances

![A grey background with white text

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAGYAAABRBAMAAADSjs5EAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAbUExURXl5eb29vfPz86ysrODg4NbW1unp6cvLy5WVlSXpS4cAAAAJcEhZcwAADsQAAA7EAZUrDhsAAADBSURBVFjD7ZCxCoMwFEWvNlZHcXDWFDVj8AukVLK6dFcpdPcLJF/eRC10MUrnd5ZkyOHkPRAEQRAEsY/H+c2erHsDYQ+8AHXkZEjmX2eMV6eeXY6Wy+3byc44qXne2s5YGGdojTOVEReuecTmBI1vHS2VJ4LG3anl6vix/dsQNIp1F+F27Leso+XiIFfMxM85WweJ6RztoJaPqLDz9Ms8YJUnwkw7HM5z6PJpY/eqT40TVZj4fN3fG0EQBEEQfwB8AGwXKH7wIWrnAAAAAElFTkSuQmCC)

**Build.net**, developed in Visual Basic (VB), integrates with **Build Software**, which is written in C# and acts as the core system. This integration allows Build.net to leverage the advanced capabilities and base functionality of Build Software, providing a unified workflow despite using different programming languages

![A blue background with white text

Description automatically generated](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAGYAAABRBAMAAADSjs5EAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAbUExURR5/q8HY4tLh6K7N3PPz80+au3ewyZXA0+Hp7RAjFeEAAAAJcEhZcwAADsQAAA7EAZUrDhsAAADaSURBVFjD7ZAxbgIxEEU/s16vyw3iABZIkNIkkWgdmGUpt0CCEiGKlMAJkptnDAUVXtHPKyzb8tP3fCiKoiiK8pyBx1dXrmVH9f0iomJeYfjcsb94P9v4cMwMVcT2kHHQoPWFfzjXa3Kozjkz6yOdHQcabW5n0yUn5Jx5EaamG6OlhiSujIMgzin7tyMdPvdSwpDqNFYha18HMAbHS8m8pGAb/nbMbZUqyTn0AbdIZd9zfuR1r1NMIHOM8SfzSHNvwGnX59g1pGrH/tZbCnSLPkdRFEVRlNcB/gEmYRqlSp84bQAAAABJRU5ErkJggg==)

The ECQ application, built in VB, is the basic application but is currently partially in use within organization modules designed to process data efficiently and deliver precise outcomes, forming the backbone of the application's operational capabilities.

.

The Web API project serves as Third party communication.

A grey background with black text

Description automatically generated

**Sub Projects**: This project consists of 200+ subprojects/modules, which include both major and minor components. The Key modules are

A close-up of several blue squares

Description automatically generated

# Status of the elements of the scope

**Methodology used** - Quantitative Research to evaluate the quality and trustworthiness of data, findings, and information. We have used Completeness and Reliability as parameters.

|  |  |
| --- | --- |
| Completeness | Reliability |
| Completeness refers to the state or condition of having all necessary or appropriate parts. It signifies wholeness or the quality of being entire, without any missing elements. | Reliabilityrefers to the quality of being trustworthy or dependable.  It indicates the ability to be relied upon for accuracy and performance in AS-IS situation |

## Technical Architecture documentation of the project and subprojects

The document review indicates partial availability of technical architecture documentation, with primary focus on the core C# and VB.NET frameworks. The project includes integration between old (VB.NET) and new (C#) architectures through a bridge project (Build General), however, there is a lack of comprehensive architecture diagrams & descriptions for all the projects, especially for those that that have been migrated from VB to C# & those that have not been migrated.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 60% | 20% |

## Functional documentation of the project and subprojects

Sylvia highlighted an issue where numerous functional documents containing the ideas discussed are scattered across multiple locations. Within these documents, some ideas have been successfully executed, while others remain unimplemented. This unstructured & non-organization creates confusion and makes it difficult to track the status of each idea, leading to inefficiencies in the functional workflow and project management.

Functional documentation is incomplete and inconsistently organized, as highlighted in the document.While there are documents for major modules like Inventory Management, Invoicing, Contract Management, and Asset Management, they lack sufficient detail on workflows and business logic

|  |  |
| --- | --- |
| Completeness | Reliability |
| 60% | 20% |

## Documentation of the API projects

The document includes some API-related information, primarily configuration files and limited technical descriptions, but it lacks full API documentation. Specifics about endpoints, data models, and client-server interactions are not fully detailed, which hinders the project team’s understanding of how various modules and external systems communicate.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 10% | 10% |

## Cloud application documentation and features - Not provided

|  |  |
| --- | --- |
| Completeness | Reliability |
| 0% | 0% |

## Coding standards

Coding standards have been noted, especially around the use of Entity Framework for database interactions, avoidance of hard-coded SQL statements. There is absence of inline comment which adheres to minimalistic documentation within the code and impact future maintainability. While high-level coding standards are followed, there’s a lack of specific documentation on standards enforcement across modules and languages, given the mix of VB.NET and C# components.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 60% | 60% |

## Database relationship documents of the same

A comprehensive review of the database, documenting the total count and type of objects, including tables, views, functions, procedures and sequences. The relationship between the tables have been identified and thoroughly documented.

However, while the inventory of the procedures and functions is complete, their individual purpose and the functionalities remain largely unclear. Further analysis is needed to understand the specific operations performed by each of the functions and procedure within the database.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 75% | 75% |

## Connectivity documentation of all subprojects and their relations

Connectivity documentation for subprojects is minimal, with no detailed or standardized documents explicitly outlining how each module or subproject connects. The lack of standardized connection methods to the database and limited access to version control add further complexity to this aspect, suggesting that connectivity details remain a critical gap in the documentation. Documents are distributed in SharePoint and the latest version couldn’t be identified.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 10% | 5% |

## System flow diagram(s) of the current application (if any)

|  |  |
| --- | --- |
| Completeness | Reliability |
| 0% | 0% |

## Installation, implementation and user guide(s) for all applications (if any)

An overall walk through has been given on installation, implementation but they haven’t provided any other documentation regarding this. However Build Software mentioned they have an user guide but it is out dated.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 0% | 0% |

## Collection of source code

The source code files have been successfully provided by Build Software through their SharePoint site collection. This collection is complete & indicating a high level of trustworthiness in the provided materials.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 100% | 90% |

## Collection of Database

Build Software has successfully provided the database files through their SharePoint site. This collection is fully complete & accuracy of the database materials.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 100% | 90% |

## Analysis of the source code

Build Software provides full source code which does not have inline comments and comprises a total of 222 sub-projects/modules which include both major and minor components. Build.net and Emma.net are the main startup projects within the solution, and both are implemented in VB.NET.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 60% | 70% |

## Analysis of the Database

Build software have all the Store procedure, tables & functionalities but don’t have much information which are used in production and latest code.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 100% | 90% |

## Reporting tools

Build Software uses DevExpress report for reporting purpose in the application.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 100% | 90% |

## Analysis of current architecture and dependencies from the documentation

Analysed current architecture which integrates both VB.NET and C#, creating dependencies between older and newer modules. While there is a “Build General” project to facilitate connections, this documentation doesn’t provide the development details on how the integration was done.

|  |  |
| --- | --- |
| Completeness | Reliability |
| 20% | 20% |

**Status Overview of Project Elements**

1. Technical Architecture and Functional Documentation are partially complete but lack organization, leading to inefficiencies.
2. API Documentation and Cloud Application Documentation are severely lacking, which could hinder integration and understanding.
3. Coding Standards are noted, but the absence of inline comments may affect future maintainability.

| Item Description | Completeness | Reliability |
| --- | --- | --- |
| 6.1 Technical Architecture Documentation | 60% | 20% |
| 6.2 Functional Documentation | 60% | 20% |
| 6.3 Documentation of API Projects | 10% | 10% |
| 6.4 Cloud Application Documentation | 0% | 0% |
| 6.5 Coding Standards | 60% | 60% |
| 6.6 Database Relationship Documents | 75% | 75% |
| 6.7 Connectivity Documentation | 10% | 5% |
| 6.8 System Flow Diagrams | 0% | 0% |
| 6.9 Installation, Implementation, and User Guides | 0% | 0% |
| 6.10 Collection of Source Code | 100% | 90% |
| 6.11 Collection of Database | 100% | 90% |
| 6.12 Analysis of the Source Code | 60% | 70% |
| 6.13 Analysis of the Database | 100% | 90% |
| 6.14 Reporting Tools | 100% | 90% |
| 6.15 Analysis of Current Architecture and Dependencies | 20% | 20% |

# Overview of Information Received

## Build Software - C#

We analysed the various categories and Files/Package name, and their purposes are listed below:

* It is stored in the "SamenwerkingInsoftBuild" SharePoint library, under the folder path:
* Transfer -> Source Code
* File Size: The file size is 435 MB.
* The file was last modified by Peter Grauwels on October 14,2024. He moved, edited, and created the file on this date.
* Path: SamenwerkingInsoftBuild -> Documents > Transfer -> Source Code -> Build.net\_v7.9.7.7z

A screenshot of a computer

Description automatically generated

| Category | File/Package Name | Description | Purpose | |
| --- | --- | --- | --- | --- |
| Source Code & Packages | Build.net\_v7.9-A | Main source code package with all associated dependencies. | Core application and package dependencies | |
| Module Walkthrough | Functional & Technical Walkthrough | Documents or materials for functional and technical understanding, organized by module. | Guides module-specific development and updates | |
| Business Logic Files | .BL Files | Files containing the core business logic for each module. | Implements primary business rules and logic | |
| Data Layer Files | .DL Files | Files related to data handling or data layer functionality (exact purpose to be confirmed). | Possibly manages data storage and retrieval | |
| Resource Files | Resource Files | Binary files for storing multilingual resources, supporting Dutch, English, and French languages. | | Enables localization and language support | |
| Contracts Project Files | Contracts Project Files | Files defining business logic contracts, data contracts, or facilitating API calls. | | Specifies data structures and API interactions | |
| Client Configuration | winclient Files | Configuration or setting files for the client application (exact function to be verified). | | Likely configures client application parameters | |
| Testing Files | Unit Test & Integration Test Files | Files for unit and integration testing frameworks, covering isolated and end-to-end tests. | | Ensures functionality through testing frameworks | |

## ECQ – VB

Each of the component categories and their corresponding File/Element details along with the description are listed below:

| Component Category | File/Element | Description |
| --- | --- | --- |
| Controller | RibbonCommandController | Implemented in VB.NET, this controller manages the main menu and parent form  objects for the ribbon-based UI. |
| Forms and UI | frmAboutBox.Designer | Design file for the "About" form, inaccessible due to errors, preventing visualization of layout. |
| mdiMainBuild.vb | Uses DevExpress.XtraBars components to create bar graph designs for data visualization. |
| RibbonBarItemLoader.vb | Dynamically generates bar graphs to display product items, purchase calculations, and other metrics. |
| Templates | .dot Files | Templates in .dot format (compatible with Microsoft Word) but unsupported directly by .NET Framework. |
| Security and Authentication | .pfx Files | Personal Information Exchange files used for cryptographic functions, enhancing secure communication and authentication. |
| Build.Net.snk (Strong Name Key) | Strong Name Key file used for versioning, security, and deployment of assemblies to the Global Assembly Cache (GAC). |
| Control Types | DevExpress Controls | Third-party UI components from DevExpress provide advanced UI functionalities. |
| User Controls | Customizable controls are created specifically for this application. |
| Custom Controls | Controls with tailored functionality designed for specific application requirements. |
| Code and Logic | Business Logic and Custom Controls | Custom controls and calculation routines implemented in VB.NET. |
| File Structure | XML Files | Contains only XML files, likely representing project configurations and settings. |
| Source Code Absence | - | Actual executable code files are missing; XML files provide configuration and settings details only. |

## Start Up Projects – Build.Net & Emma.Net (VB)

The startup projects (Build.Net & Emma.Net(VB)) component category and File/Module along with the description are listed below:

| Component Category | File/Module | Description |
| --- | --- | --- |
| Batch Processing | BatchService | Contains only project configuration XML files. The purpose is not fully clear but likely intended for batch processing tasks. |
| Build API Projects | Build Api | Includes only configuration XML files; missing actual code files. |
| BuildSoftware.Build.Api | Likely contains API definitions, including controllers, interfaces, methods, and endpoints for client-server communication. |
| BuildSoftware.Build.Api.Models | Expected to hold model classes representing database entities and related POCO (Plain Old CLR Object) classes. |
| BuildSoftware.Build.ApiEncryption | Presumably, it contains classes for encryption, implementing security mechanisms like symmetric encryption for API transactions. |
| Console | Console | Configuration XML files only; actual purpose is unclear. |
| Data Relationship Framework | BuildSoftware.Xm.RelationImporter | Framework for data operations, likely defining mappings and relationships between database tables and entities. |
| Web API Modules | WebApi | Contains only configuration XML files; functionality is not specified. |
|  | BuildSoftware.WebApi.TEAplus.Api | Expected to contain API functions and leverage BuildSoftware modules for task execution. |

## Database Objects

We have four databases, three hosted in the cloud and one on-premises. The cloud databases are T.Build.Cloud.Settings, T.Build.Subcontracting, and T.Build.Cloud.XRM, while T.Build.DataEmpty\_Alg is our on-premises database. During the discovery phase, the cloud databases are not within the scope of this phase. We focused on the on-premises database, analysing the total count and types of objects it contains, including tables, views, functions, procedures, and sequences are listed below:

|  |  |
| --- | --- |
| Database Object | Count # |
| Tables | 802 |
| Views | 480 |
| Stored Procedures | 626 |
| Functions | 113 |
| Sequences | 1 |

## Source code metrics

The source code and database structure for various projects within the Build Software, ECQ & startup project systems. This type of data is typically obtained from

| Project | Base Folder | Tech | Projects/ Module | Source code's Line # | Namespace | File # |
| --- | --- | --- | --- | --- | --- | --- |
| BuildSoftware\BackOffice | BuildSoftware | C# | 5 | 295363 | 34 | 17569 |
| BuildSoftware\CloudSync | BuildSoftware | C# | 2 | 1000 | 5 | 46 |
| BuildSoftware\Common | BuildSoftware | C# | 1 | 7879 | 2 | 7897 |
| BuildSoftware\Customizations | BuildSoftware | C# | 5 | 51762 | 55 | 5100 |
| BuildSoftware\External | BuildSoftware | C# | 28 | 92795 | 159 | 51635 |
| BuildSoftware\Framework | BuildSoftware | C# | 9 | 177478 | 101 | 23593 |
| BuildSoftware\HRM | BuildSoftware | C# | 5 | 513079 | 16 | 4484 |
| BuildSoftware\Invoice | BuildSoftware | C# | 5 | 47107 | 19 | 4054 |
| BuildSoftware\Material | BuildSoftware | C# | 6 | 244925 | 30 | 11000 |
| BuildSoftware\Project | BuildSoftware | C# | 6 | 1092941 | 52 | 63209 |
| BuildSoftware\Purchase.Invoice | BuildSoftware | C# | 7 | 121682 | 49 | 9002 |
| BuildSoftware\Sales.Invoice | BuildSoftware | C# | 9 | 297212 | 63 | 19289 |
| BuildSoftware\Settings | BuildSoftware | C# | 6 | 892754 | 80 | 44086 |
| BuildSoftware\Stock | BuildSoftware | C# | 5 | 44870 | 17 | 6093 |
| BuildSoftware\Tools | BuildSoftware | C# | 6 | 102132 | 28 | 6757 |
| BuildSoftware\XRM | BuildSoftware | C# | 6 | 298523 | 25 | 21465 |
|  |  |  |  |  |  |  |
| ECQ\01. Controls | ECQ | VB & C# | 8 | 162557 | 33 | 8659 |
| ECQ\03. Import\_Export | ECQ | VB | 4 | 143597 | 29 | 7248 |
| ECQ\04. Andere | ECQ | VB | 6 | 99252 | 18 | 5237 |
| ECQ\50. DevExpress | ECQ | VB | 2 | 75246 | 10 | 3292 |
| ECQ\90.Contracten | ECQ | VB | 5 | 6254 | 24 | 1073 |
| ECQ\Build | ECQ | VB | 1 | 243918 | 14 | 9860 |
| ECQ\EMMA | ECQ | VB | 10 | 3157127 | 124 | 135842 |
| ECQ\General\ | ECQ | VB | 1 | 10022 | 4 | 464 |
| ECQ | ECQ | VB | 21 | 6990272 | 306 | 294589 |
|  |  |  |  |  |  |  |
| Startup projects\Build.Net | Startup projects | VB | 1 | 29349 | 6 | 1336 |
| Startup projects\EMMA.Net | Startup projects | VB | 1 | 7673 | 4 | 445 |
| **Total** | | | **171** | **15206769** | **1307** | **763324** |

# Inventory of the Documents Received

| Database Object | Folder and Files Description |
| --- | --- |
| Architecture | **Folder Structure Overview:** |
| **Project Architecture Folder & Files**: This folder contains essential resources related to the overall project structure, including files and subfolders for various projects. |
| **Migration Files**: This section holds migration-related documents, detailing the transition from .NET 6 to .NET 8, along with relevant technical analysis and descriptions of changes. |
| **API Documentation folder**: Comprehensive details on the APIs, including endpoints, technical descriptions, and their respective analysis. This folder also includes demo videos, email correspondence, and documentation on tools such as EF Core Power Tools, Graph API, and Microsoft Graph. |
| **Install Folder**: |
| Contains database backup files (.bak) and relevant SQL Server 2019 installation documents. |
| Includes a scripts folder along with the SQL Server 2019 executable software. |
| **Training Folder:** Stores all training materials and documents. |
| **Translation Folder**: Contains documents and resources related to project translations. |
|  |
| Architecture Web | Contains technical analysis on build software along with a video showcasing the Proof of Concept (POC). |
| **Books Folder:** Includes eBooks on Clean Architecture and Clean Code, offering valuable insights into best practices in software design and development. |
| **.POC Documents Folder:** Stores comprehensive documentation related to various Proof of Concept (POC) projects. |
| **.Portal APK Folder**: Contains documents related to the portal, including best practices for Azure implementation. |
| Bgo2Cloud | **Apps Folder:** |
| Contains key resources, including: |
| 1. Demo videos |
| 2. Project management documentation |
| 3. Silver App project presentation |
| **SaaS Folder:** |
| Holds important documents related to: |
| 1. Resource files |
| 2. Graph API documentation |
| 3. Database backups |
| 4. Security documentation |
| 5. Translation files |
| 6. Database scripts |
| 7. Design notes |
| General | **Folder Structure Overview:** |
| **Analysis Documents**: Contains comprehensive documents detailing various technical and business analyses. |
| **Installation & Release Notes:** |
| Installation notes provide guidance on setting up software or infrastructure. |
| Release notes include details on version updates and new features. |
| Database Queries: Holds queries related to database operations and management. |
| **Attendance & Holidays Lists:** Contains records of attendance and lists of holidays. |
| **Infrastructure Documents**: Includes technical documentation related to the organization's infrastructure setup. |
| **Personal Folders:** Stores individual files and documents categorized by personnel. |
| **Budget Reports**: Contains detailed financial reports, including budgeting and expenditure information. |
| **Marketing Documents:** Includes materials and documents related to marketing strategies and campaigns. |
| **Product-Related Documents:** Stores information, specifications, and documentation on various products. |
| **Project Folder:** Contains numerous subfolders and files for different projects. |
| **Sales Folder:** Includes comprehensive sales data and reports. |
| **Support Folder:** Holds support-related documents and videos to assist users and staff. |
| Marketing | **Folder Structure Overview:** |
| **Accounts List:** Contains financial or account-related information. |
| **Build Analytics Videos:** Includes videos that analyze various aspects of the build process. |
| **BIM Videos and Documents**: Stores videos and documents related to Building Information Modeling (BIM). |
| **Company Presentation**: Contains a formal presentation showcasing the company’s profile, offerings, and achievements. |
| **Events**: Includes documentation, materials, or media related to company events. |
| **Mobile App Folder:** Holds resources, documentation, and files pertaining to mobile app development or deployment. |
| **PowerPoint Presentations (Build Software):** Contains presentations on build software, providing insights and technical details. |
| **Social Media Files:** Stores assets and materials related to social media campaigns or engagements. |
| **Testimonials**: Contains customer or client testimonials, showcasing feedback and success stories. |
| **Webinar Data:** Includes recordings, presentations, and other materials related to webinars conducted by the company. |
| **Images:** Contains image files used for various purposes within the company. |
| **Build Software PDF Documents**: Holds PDF documents that provide detailed information about Build software and related technologies. |
| Partners | It contains company partner related information in folders and files |
| Planning |  |
| Product | **Folder Structure Overview:** |
| **Product Information & Documents:** Contains detailed documents and resources related to product specifications, features, and updates. |
| **Sprint Planning**: Includes documents and plans related to sprint cycles, outlining project milestones and tasks. |
| **Testing**: Holds testing documentation, results, and related materials for product quality assurance. |
| **Translation**: Contains files related to language translation for product localization. |
| **Bug Videos**: Stores video recordings demonstrating identified bugs and issues. |
| **POC Videos**: Contains videos showcasing Proof of Concept (POC) demonstrations. |
| **Financial Connecto**r: Includes documentation and resources related to financial integration or connectors. |
| **Reports**: Holds various folders and files containing important project or financial reports. |
| **Winbooks Folder:** Contains text files, including notes and data, stored in Notepad format. |
| Project | **Project Folder Overview:** |
| **Project folder Description:** Contains detailed documentation outlining the scope, objectives, and key deliverables of the project. |
| **Contracts**: Stores legal agreements and contracts related to the project. |
| **Functional Analysis**: Includes documents detailing the functional requirements and analysis of the project. |
| **Installation**: Contains installation guides and related documentation for the project setup. |
| **Client Documents**: Holds important documents provided by or for the client, including specifications, feedback, and communication records. |
| Sales | **Folder Structure Overview:** |
| This folder contains individual subfolders for each company, with comprehensive documentation on the following: |
| **Budget:** Detailed financial reports and budget information for each company. |
| **Meeting Details**: Records of meetings, including agendas, minutes, and key decisions. |
| **Architectural Planning Documents:** Contains plans and documents outlining architectural strategies and designs. |
| **Visual Planning Documents**: Includes visual representations and planning materials, such as diagrams, sketches, or design mockups. |
| Security-NSI2 | **Folder Structure Overview:** |
| **Asset Management:** Contains documents and resources related to the management and tracking of company assets. |
| **Data Retention**: Includes policies and guidelines on data retention and archiving practices. |
| **Queries:** Stores database or system queries for analysis, reporting, or troubleshooting. |
| **Development Security Assessment**: Contains reports and assessments focused on the security of development processes and systems. |
| **Videos**: Includes video content related to security training, assessments, or informational resources. |
| **Cybersecurity Policy Documents:** Holds official policy documents outlining the organization's approach to cybersecurity. |
| **Network Security Documents**: Contains guidelines and protocols for ensuring network security. |
| **Self-Assessment Documents**: Includes self-assessment reports evaluating internal compliance and security measures. |
| Staff-Briefing | **Folder Structure Overview:** |
| **Staff Briefing Folders (2016-2024)**: Contains documentation of staff briefings, organized by year, from 2016 to 2024. |
| **Postman API Testing Videos**: Includes video tutorials and recordings demonstrating API testing using Postman. |
| Support | **Folder Structure Overview:** |
| **Cloud Azure Folder:** Contains resources and documentation related to Microsoft Azure cloud services and solutions. |
| **Data Configuration Encryption Error:** Includes documentation addressing issues and solutions related to data configuration encryption errors. |
| **External Installations**: Holds documents and resources pertaining to the installation of external software and applications. |
| **Support Documents**: Contains support materials and documentation for various software applications, including Power BI, SQL Server, and others. |
| Other files in folders | Teams meeting videos, build software-technical analysis documentation and presentation and it contains image files |

# Overview of the Meetings Held

**Technical Code Walkthrough**: Reviewed code structure, architecture, and key dependencies within the Build Software system.

**Functional Walkthrough**: Focused on core functionalities and workflows to align technical and business requirements.

**Documentation Review**: Discussed Build Software documentation for both technical and functional insights, emphasizing clarity and accessibility.

**DevOps Review**: Explored the deployment pipeline, CI/CD practices, and the DevOps toolchain.

**Q&A on blocks**: Addressed specific challenges blocks, clarify complex points, and maintain project momentum.

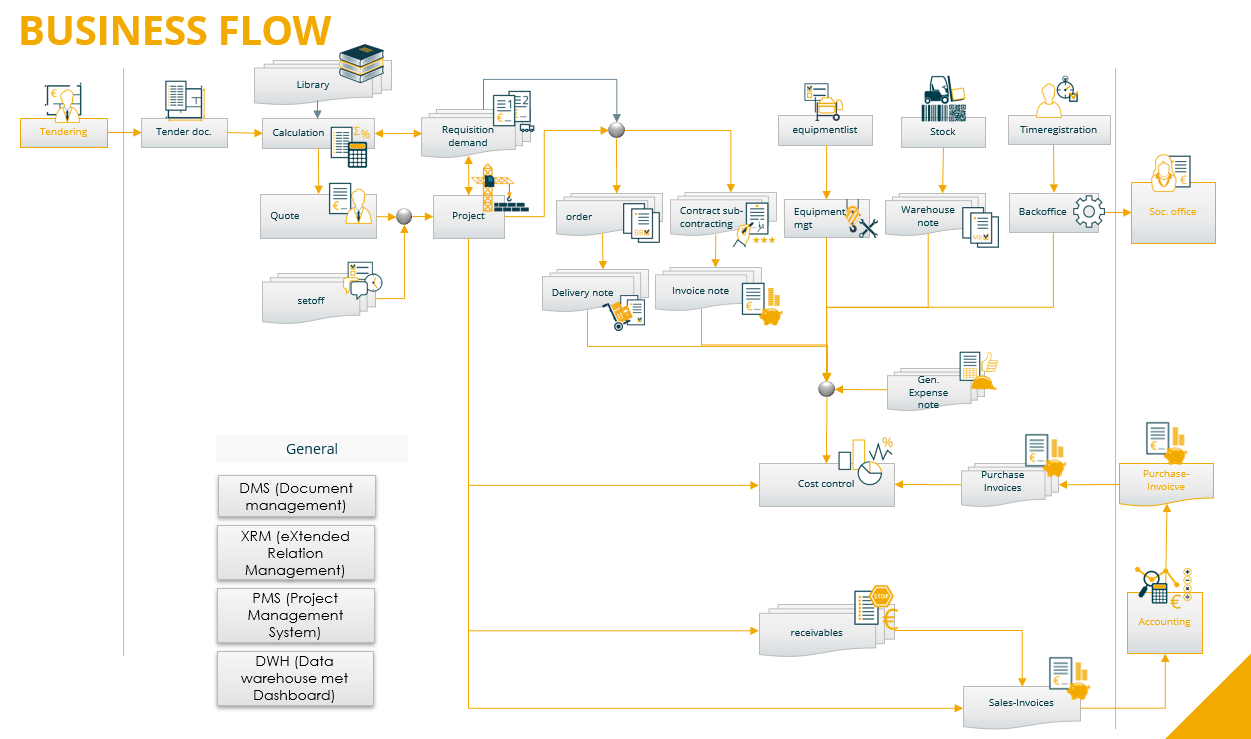
A screenshot of a software project

Description automatically generated

# Major Discoveries

## Business Flow

The business flow shown here illustrates a structured process for managing project and order workflows, from initial tender and calculation through to project execution, order management, and inventory handling. Key activities include managing equipment lists, tracking warehouse notes, controlling costs, and processing invoices, ultimately leading to financial reporting and accounting.



Build Software provided a few modules walk through like calculations, contracts (subcontracts), materials.

## Technology Stack

|  |  |
| --- | --- |
| Programming Language | C#, VB.NET |
| ORM Framework | Entity Framework |
| Tool | Version Studio 2022 |
| Database | SQL Server 2014 |
| Database Access | Stored Procedures, LINQ |
| API | ASP.NET Web API |
| Build Security | SNK (Strong Name Key) file for assembly signing in .NET |

## SQL server version and support

| SQL Server Version | Release Date | Mainstream Support End Date | Extended Support End Date |
| --- | --- | --- | --- |
| SQL Server 2005 | 7-Apr-05 | 12-Apr-11 | 12-Apr-16 |
| SQL Server 2008 | 8-Jul-08 | 8-Jul-14 | 9-Jul-19 |
| SQL Server 2008 R2 | 20-Jul-10 | 8-Jul-14 | 9-Jul-19 |
| SQL Server 2012 | 20-May-12 | 11-Jul-17 | 12-Jul-22 |
| SQL Server 2014 | 5-Jun-14 | 9-Jul-19 | 9-Jul-24 |
| SQL Server 2016 | 1-Jun-16 | 13-Jul-21 | 14-Jul-26 |
| SQL Server 2017 | 2-Oct-17 | 11-Oct-22 | 12-Oct-27 |
| SQL Server 2019 | 4-Nov-19 | 7-Jan-25 | 8-Jan-30 |
| SQL Server 2022 | 16-Nov-22 | Expected: December 2027 | Expected: December 2032 |
|  |  |  |  |

## Development Tool version release & extended support information

Build Software used Visual studio 2022 which has extended support till January 2025.

| Visual Studio Version | Release Date | Mainstream Support End Date | Extended Support End Date |
| --- | --- | --- | --- |
| Visual Studio 2005 | 27-Oct-05 | 12-Apr-11 | 12-Apr-16 |
| Visual Studio 2008 | 19-Nov-07 | 10-Apr-18 | 10-Apr-18 |
| Visual Studio 2010 | 12-Apr-10 | 14-Jul-15 | 14-Jul-20 |
| Visual Studio 2012 | 15-Aug-12 | 9-Jan-18 | 9-Jan-23 |
| Visual Studio 2013 | 17-Oct-13 | 9-Apr-19 | 9-Apr-24 |
| Visual Studio 2015 | 20-Jul-15 | 13-Oct-20 | 14-Oct-25 |
| Visual Studio 2017 | 7-Mar-17 | 12-Apr-22 | 13-Apr-27 |
| Visual Studio 2019 | 2-Apr-19 | 10-Apr-24 | 8-Apr-29 |
| Visual Studio 2022 | 8-Nov-21 | Expected: January 2030 | Expected: January 2035 |

## Integration of VB.NET Code in C# for Key Business Functions

**Cross-Platform Integration for Core Modules:**

The project has incorporated VB.NET code within the C# environment to support crucial modules, including Inventory Management, Invoicing, Contract Management, Back Office Operations, and Asset Management. By leveraging existing VB.NET components, the project achieved immediate functionality for these modules without reimplementation in C#.

**Long-Term Challenges with VB.NET Code Retention:**

Keeping VB.NET components integrated within a primarily C# framework presents potential long-term challenges:

* **Compatibility Issues**: Future updates and versions of the .NET platform may create compatibility challenges between VB.NET and C#, requiring additional refactoring and adjustments.
* **Resource Availability**: VB.NET is less commonly used in modern development, which may create a skill gap as it becomes harder to find resources proficient in both languages.
* **Increased Maintenance Overhead**: Maintaining two languages within a single project adds complexity and demands additional resources for debugging, updates, and aligning functionality across languages.
* **Performance Concerns**: The integration of VB.NET may impact performance, as inter-language communication can introduce latency and additional complexity in execution.

## Impact of Unused VB.NET Code on Code Quality

A lot of unused VB.NET code remains in the codebase, which adversely affects the quality of the project. This redundant code not only adds unnecessary bulk to the project but also complicates maintenance efforts and reduces the clarity of the code structure. Regular removal of unused code is essential to streamline the codebase, enhance readability, and improve maintainability.

### Recommendations for Code Quality and Future Development

* **Code Cleanup**: It is recommended to identify and remove all unused VB.NET code to reduce technical debt, enhance code clarity, and simplify future maintenance.
* **Unified C# Codebase Transition**: For long-term sustainability, transitioning critical VB.NET modules to C# would be beneficial. This would provide a cohesive codebase, reducing compatibility concerns and maintenance overhead, while aligning with modern .NET development standards.
* **Establishing Code Standards**: Implementing clear coding standards and routine code reviews can help maintain code quality, prevent redundancy, and ensure optimal performance for future iterations.

## Documentation

### Size of existing documentation

The documentation provided is **over 1 Terabyte (TB)** in size, which presents several critical issues that hinder its effective use and accessibility. These challenges include:

**Lack of Proper Labeling**: The volume of documentation, combined with inconsistent and unclear labeling, makes it difficult to quickly locate and identify specific files. Files and folders are not organized in a systematic manner, which leads to confusion when trying to find relevant documentation for specific project components or tasks.

**Outdated Information**: A significant portion of the documentation appears to be outdated, which results in discrepancies between the documented information and the current state of the project. The team members refer to older versions of documents that do not reflect recent changes or updates in the system, process, or technology.

### Multiple versions with different departments

The documentation handoff process across departments lacks adequate tracking and update procedures. As a result, the following issues are present:

**No Standardized Process for Handoff:** Documentation is transferred between teams without proper updates or version control, leading to multiple versions of the same document existing in different departments and leading their own independent life. These versions may contain conflicting information, further compounding the confusion.

**Knowledge Gap**: With developers retaining most of the project knowledge in their heads, critical information is not being systematically shared across departments through documentation. As documentation moves between teams, there is a lack of coordination, and knowledge gaps emerge, particularly for teams that are not directly involved in the development process.

### Hard to find any relevant documentation

One of the main challenges with the existing documentation is its disorganization and lack of easy access. The documentation is scattered across multiple storage locations and formats, making it very difficult to find the right information when needed.

**Storage Locations**: Documentation is dispersed across different folders & locations. Without a centralized repository, team members struggle to locate documentation even if they know what they’re looking for.

**Inconsistent Labeling and Versioning**: Files are often mislabeled or not consistently named, making it nearly impossible to quickly identify the content. Furthermore, multiple versions of the same document may exist in different places, increasing the risk of using outdated or incorrect information.

### Risk in Documentation

Sylvia highlighted an issue where numerous functional documents containing the ideas discussed are scattered across multiple locations. Within these documents, some ideas have been successfully executed, while others remain unimplemented or have been declined by the team. This disorganization creates confusion and makes it difficult to track the status of each idea, leading to inefficiencies in the workflow and project management.

### Recommendation

1. To address this issue and streamline further development, it is recommended to centralize all functional documents into a single, accessible repository, such as Odoo**.**
2. Odoo has version **control** would also ensure that the most current information is always accessible, and changes can be tracked over time.
3. This repository should be organized into clearly defined categories, such as "**Executed Ideas**," "**Pending Ideas**," and "**Declined Ideas**," with each document tagged by its status.
4. Implementing version control would also ensure that most current information is always accessible, and changes can be tracked over time.

|  |  |  |
| --- | --- | --- |
| Odoo | Benefits | Description |
| Benefits of Indexing | Structured Organization | Odoo provides the ability to create hierarchies and structures for organizing information. You can create Parent Pages for high-level categories (e.g., "Backoffice Module," "Sales Invoice," "Email Service," "Unit Testing," etc.) and Child Pages for detailed sections (e.g., specific ideas, test cases, or progress reports). |
| Searchable Tags and Labels | To further enhance indexing, Odoo allows the use of labels and keywords. You can tag each page with relevant terms like "Executed," "Pending," "Declined," etc. This allows quick searches for specific statuses or categories of documents. |
| Table of Contents (TOC) | Each page in Odoo can automatically generate a Table of Contents based on the headings within it, which provides users with a quick overview of the content structure on each page, enhancing navigability. |
| Benefits of Cataloging | Document Categories | Establish a clear categorization system for the documents. |
| Module-Specific Pages | Organize documentation by module (e.g., "Backoffice," "Sales Invoice," etc.). Each module page can include sections for Functional Design, Test Cases, Execution Status, Issues & Resolutions, and Related Documents. |
| Status Tracking | Odoo integration with tools like JIRA & Project management allows you to sync status updates from tasks or ideas directly within the documentation. You can use JIRA & Project management to pull in issue statuses and track progress from both technical and non-technical perspectives. |

# Observations & Findings

## Database

**SQL Statements:** All SQL operations were abstracted through Entity Framework or stored procedures which are proper in standard.

**Data Management and Mapping**

**AutoMapper**: AutoMapper was used extensively to simplify object-to-object mapping, optimizing data transformations and enhancing code maintainability which are proper in standard.

**Data Transfer Objects (DTOs)**: DTOs were incorporated to manage structured data transfer across various layers, reducing unnecessary data exposure and improving efficiency which are proper in standard.

**Custom Resolvers**: Custom resolvers were implemented to handle specific mapping scenarios, providing flexibility for complex data interactions while maintaining clean, standardized code which are proper in standard.

**SQL Script Format Received from Client**

Build Software provided SQL scripts in PDF format. While this format provides a readable overview, it limits our ability to directly implement or modify the scripts for development and testing purposes. For optimal integration, we recommend that SQL scripts be shared in a text-based format (e.g., .sql) to facilitate accurate execution and efficiency in handling database operations.

**Issue**: Converting PDF content to executable scripts introduces additional steps and potential errors, which can be avoided with direct access to script files. We kindly request that any future SQL scripts be provided in a development-ready format to streamline the process.

## Coding Standards (Define the quality of the coding standard)

**Compliance with Coding Standards**: The project adhered strictly to standardized coding practices, ensuring consistency, readability, and maintainability throughout the source code.

**Object-Relational Mappin**g: Entity Framework was employed to map and define relationships between database tables and entities effectively. This framework facilitated seamless interaction with the database, minimizing manual data handling.

**Incline Comments** - No inline comments were present in the source code. While this adheres to a minimalistic documentation style, future maintainability could benefit from in-line explanations to clarify complex sections of the code.

## Unit testing

* Unit testing has been conducted in back office, sales module, email service module and Encryption & Decryption (framework testing).
* No unit testing has been performed in the Build.Net and Emma.Net VB projects other than the above modules.

## Test Cases

Functionality test cases are not fully defined and not executed. Information on critical business logic and operational processes information rest with different teams and individuals.

## Regression Testing

Build Software undergoes limited testing focused solely on tickets level rather than comprehensive end-to-end (E2E) regression testing, it poses significant risks to the stability and quality of the software.

## Knowledge resides in the heads of the current maintenance team

According to the BS team, the technical project knowledge completely resides in the heads of the staff.

* Serious dependency on specific staff members
* lower maintainability of the subprojects due to low availability of staff with knowledge on the project.
* Staff cannot be exchanged across projects

## Hard coding

No SQL statements were hard coded in the source code, enhancing security, performance, and ease of maintenance.

## Challenges of Using Non-English Loggers and Error Messages

When source code uses loggers and error messages in a non-standard or localized language like Dutch, and it lacks specific details (e.g., the method that throws an error), it can lead to significant challenges in a professional development environment.

**Lack of Clarity**: Developers working on the codebase, especially those who join the team later or work on other components, may struggle to understand logs, hindering effective documentation and knowledge transfer.

**Comprehensive Error Logs**: Always include the name of the method or component that throws an error in log messages. This improves traceability and speeds up debugging.

**Structured Logging**: Utilize structured logging frameworks (e.g., log4net) to include details such as timestamps, severity levels, method names, and stack traces automatically.

## Challenges of having VB.NET integrating with C# in a Project

**Code Consistency**: Maintaining a consistent coding style and convention becomes more difficult when different languages are used in the same solution.  
**Knowledge Gap**: Developers may need to understand both VB.NET and C#, which could require additional training and slow down development if the team is not fluent in both languages.

**Readability Challenges**: Switching between C# and VB.NET in a codebase can be challenging for developers who are more comfortable with one language. This can impact code reviews, debugging, and overall collaboration.

## 10.10. Recommendations from Observation

INSOFT focus is on improving documentation, code quality, testing, and error handling to enhance the maintainability, efficiency, and stability of the software.

**Centralized Documentation**: Use a version-controlled repository (e.g., Odoo) to organize and track all functional documents, categorizing ideas based on status for easy access and version tracking. This should include categories like "Executed Ideas," "Pending Ideas," and "Declined Ideas" for clear visibility of document status.

**Code Quality and Transition**: Clean up unused VB.NET code, adopt a unified C# codebase, and establish coding standards to reduce technical debt and ensure consistency. This shift would reduce compatibility issues, improve performance, and make the codebase easier to maintain and enhance in the long term.

**Enhanced Logging**: Standardize log messages in English with detailed error information to improve clarity and debugging efficiency. Use structured logging frameworks for better traceability. Structured logging frameworks like log4net can automatically add details like timestamps and stack traces, speeding up debugging and reducing potential for miscommunication.

**Testing**: Limited testing focused only on ticket-level changes presents risks to overall software stability. To improve quality assurance, it’s recommended to implement end-to-end (E2E) regression testing. Collaboration between the client, stakeholders, and testing team on creating detailed test cases will ensure functionality and performance expectations are clear. Defining testing standards, including acceptance criteria and regression cases, will enable consistent practices and reliable outcomes.

# Conclusions & proposal for continuation

*The size of this project is the size BLDSW was a few years ago: 40-50 people. The current amount of staff is unable to do justice to the requirements of the Company and its clients.*

In a nutshell, wha

## Documentation Setup

Insoft will prepare a hierarchical structure documentation setup which provides a clear, organized documentation framework of AS-IS and allows team members to navigate through project information with ease. Things we consider to be followed in the Documentation set-up:

1. **Establish a Page Hierarchy for Each Branch**
2. **Add Labels for Quick Filtering and Searchability**.
3. **Embed Links and Dynamic Elements for Navigation**
4. **Set Permissions and Page Restrictions**
5. **Rules & Standards for Documentation**
6. **Document environment setup**

This approach leverages hierarchical and collaborative features to build a multi-branch documentation system that’s scalable, easy to navigate, and maintains consistency across the project.

## Analysis & Document Calculation & Invoice Module

After the documentation set-up, Insoft analyzes **calculation and invoice** module which is high priority and more complex. It is crucial to analyze and document the “calculation” and “Invoice” modules from this application.

INSOFT will ensure to prepare a clear understanding of the module's scope and interactions document within the broader application. This documentation serves as a foundation for troubleshooting, future enhancements, migrations and knowledge sharing across the team.

## Existing Documentation to catalogue

Based on the documents received from the Build software in SharePoint, INSOFT will use this as a reference and go through all the documents for grouping, indexing & cataloguing it to form an structure. We will then validate and verify the information to ensure accuracy and consistency.

## Document the store procedure in DB

INSOFT will group and identify the stored procedure which are used and unused .Also, INSOFT will document the used stored procedure with its functionality and purpose.

## Identify all the unused files(VB code)

INSOFT will remove the unused VB files, which is a crucial part of maintaining a clean and efficient codebase. This task requires complete technical support from the Build Software Technical team to ensure comprehensive and identify unused code. Below is a structured guide for successfully identifying and removing unused files from the project.

* **Create a Branch:** Perform the cleanup in a separate branch to prevent potential disruptions to the main development workflow.
* **Delete Unused Files:** Identify and remove unused VB files from project.
* **Remove Unnecessary References:** Clean up broken or unnecessary references from the project properties to keep the solution organized.
* **Integration Tests:** After removing files, run the full suite of tests to ensure that the removal has not introduced any unexpected issues.

## 11.6. Document other modules in VB in sequence of priorities

After documenting the VB code related to the calculations and invoicing modules, INSOFT will proceed with additional modules in the sequence of priorities of Build Software.