IT Craft

Out with the old

COBOL

In with the new

.NET Core

OVERVIEW

The client company is a major manufacturer of frozen processed food prepared with adherence to the highest-quality standards. It ranks among the US Top 100 food manufacturers. The company offers an extensive product line with over 1500 products on its list.

The company's ERP system was developed over 30 years ago using COBOL. An in-house development team has been maintaining it since launch. The developers' average age was 60+ years by the time the client contacted us. However, their age was not the issue: a complete lack of any project documentation was.

The client approached IT Craft 7 years ago with the idea of revamping their old system. This became a successful collaboration both in system redesign and maintenance.

ERP SYSTEM
OVER 30 YEARS OLD

COBOL
PROGRAMMING LANGUAGE

DEVELOPERS' AGE
60+ YEARS

SUCCESSFULL COLLABOTRATION
FOR 7 YEARS

AIMS

The client had a strong, flourishing business. But the business depended heavily on reliable work of an outdated ERP system.

The client's goal was to renovate the ERP system and add new capabilities:

- The business owners knew they needed a system available from any mobile device. The old COBOL system could not provide enough capability.
- The client and IT Craft agreed that simultaneous maintenance of two systems would be impractical. Employees' time outlay would be too much.
- Also, maintenance of the COBOL system was expensive.
- Still, the business owners could not simply throw the COBOL system away without an immediate replacement. The entire enterprise could not stop for even a minute.

Both parties agreed the best solution was a gradual migration. Replace sub-programs of the COBOL system with modules of a new system one after another.

CHALLENGES

The development team immediately faced the following challenges:

1

SIZE OF THE SYSTEM
The enterprise operated a tightly coupled, COBOL-based system. It contained myriad interconnected sub-programs to cover all internal process requirements and document flow.

2

LARGE TO-DO LISTS
The number of project stakeholders was huge. Their needs varied. The development team transmitted old logic to the new codebase and added new features. Lists of requests for improvements and add-ons were massive after every launch.

3

OBSOLESCENCE
Both user interface and system capabilities were outdated long ago. All employees kept using the outdated system daily..

4

COSTS OF ANY DOWNTIME
The entire enterprise would grind to a standstill if the system went down for even a couple of minutes during work time.

5

TECHNICAL DEPT
The system contained an immense technical debt. This debt had been accumulating for the full 30 years of its existence. For example, the system had a large amount of unused sub-programs, failed features, and errors. Employees had built workarounds to handle errors and discrepancies.

6

CONFLICT OF INTERESTS
Conflict of interests arose between in-house COBOL developers and the outsourcing development team. It took time until the in-house team was ready to cooperate.

7

LACK OF STRUCTURED REQUIREMENTS
Unfortunately, none of the employees knew the business process from front to back. They could describe only their direct interactions with the system. Employees used the system to perform daily routines without knowing the underlying logic. As a result, they could not provide details on system requirements. Nonetheless, they needed a new system to behave identically to this one.

8

DEARTH OF COMMUNICATION BETWEEN CLIENT'S DEPARTMENTS.
Sometimes, different company departments had different views and preferences. This led to multiple iterations of system requirements. Some of the changes were done even after the development stage was completed.

9

NO DOCUMENTATION
Source code was the system's only knowledge base.

10

LACK OF MOTIVATION
Employees had nothing to gain by helping. Some of them even considered their jobs threatened because of anticipated business process automation.

PATH FROM REDESIGN OF ARCHITECTURE TO REPLACEMENT

The first action was to make changes to data storage to ensure sustainable progress in later steps. The development team:

- transferred data storage from file system to database (opting for PostgreSQL).
- started using the same database after data transfer.

When both old and new systems were connected to the same database, no data could be lost. Hence, the development team could do its work in a logical, step-by-step manner.

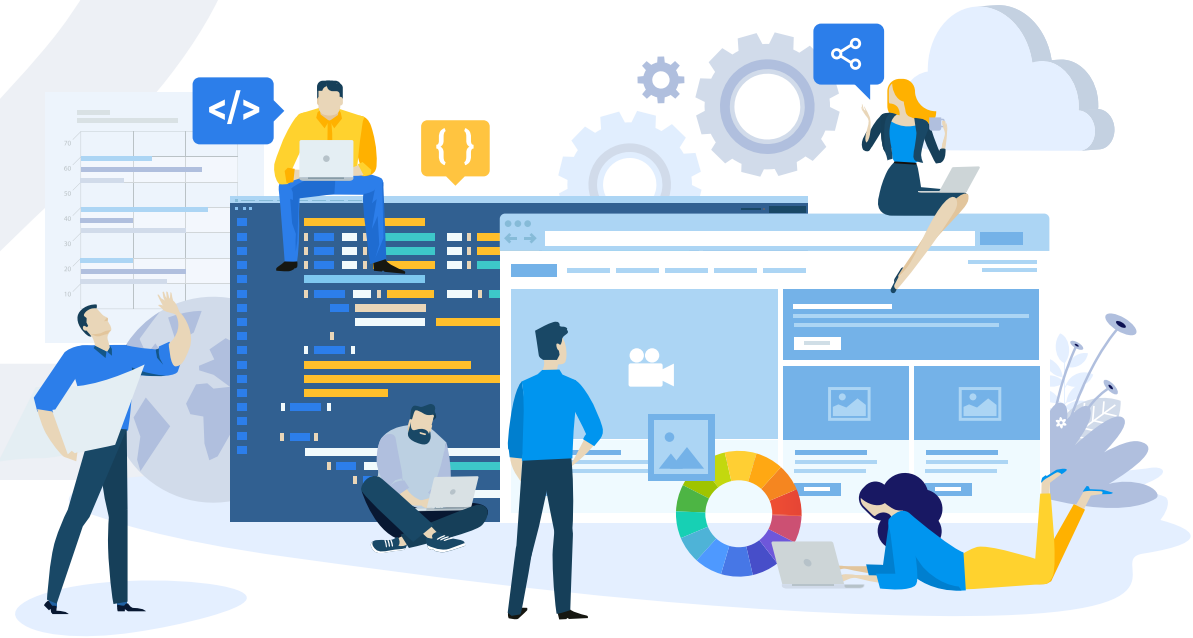
DISCOVERY AND DECOMPOSITION

Development team:

1. divided the entire system into subsystems, each having been used by different departments within the company
2. focused on one functionality of the system at a time
3. did reverse engineering of sub-programs responsible for each functionality in the current COBOL system
4. conducted interviews with stakeholders to figure out their real needs

REDESIGN AND REDEVELOPMENT

The development team prepared technical tasks and mockups of the future sub-project. They based these on the outcome of reverse engineering and on the stakeholders' requests. It was crucial to develop beyond similar functionality. The team designed features that users had required for a long time but had never been implemented in the old COBOL system.



After stakeholders' project approval, the team started development.
It is crucial to listen to stakeholders' recommendations and design an intuitive system. This creates additional cases. For example, employees had devised best practices and habits to optimize time spent working with the system. They got used to:

- certain hotkeys
- sequences of data entering (not always logical but rather deep-rooted)
- layout of certain items in certain, anticipated places

This helped break the barrier of rejecting innovations. Users easily found the right things in the right places, so their comfort level working with the new system was high.

MIGRATION AND REPLACEMENT

User Acceptance Testing, delivery and approval of project, and staff training consumed a significant amount of time. This happens in any software development project which requires multiple stakeholders' approval of requirements.

After the functionality went live, staff training started. At this stage, both old and new functionality were available simultaneously. Users compared both systems and analyzed results.

To avoid cost increase, the developers had to keep time to a minimum when both old and new systems were working in parallel. For the same reason, the development team ensured only direct compatibility. The new system performed the same activities as the old system with additional, new features required by stakeholders.

The old COBOL system did not support the new features. Because of this, both developers and stakeholders had to always remember that simultaneous use of both systems could lead to indiscernible technical limitations. For example:

- User creates a new order in the new system.
- He or she edits this order using the old system.
- User selects a new option. The new system has this option but the old system does not. Because the old system lacks this new option, it might not save "unknown data" when updating the order even when the data is essential.

However, as soon as users ensured the new system could fully replace the old sub-programs in the COBOL system, the development team switched off the old COBOL functionality. Users continued working only with new modules.

RESULTS

After the switch to the new system was completed, the development team continued working on the system to help cover all business processes, including billing, tracking, warehouse, etc.

The tactics development team used on the project made it possible to:

- streamline workflow
- add remote work with the system
- automate processes that required previously manual operations

This way, the development team incrementally transferred the entire enterprise to a new system.

IT Craft

www.itechcraft.com

site@itechcraft.com

USA: +1 469 730 0216
Germany: +49 302 067 3534
Estonia: +372 634 7354