

Legacy Systems: a Synonym for Success

The scale of a large organization virtually demands automation of many processes. As a consequence, most large commercial organizations have been developing software specific to their needs since mainframes were introduced in the 1960s. These systems collectively called “legacy systems” and are the lifeblood of day-to-day company operations. In fact, one could imply “legacy system” is a synonym for **success**. Indeed, many legacy systems have been successfully running for more than 30 years in original software technologies offered with mainframes in the 1960s and 1970s.

Arguably, the most successful legacy mainframe technology is the COBOL programming language. Today, there are more than 180 billion lines of COBOL in legacy applications still in existence. Many large organizations often have systems with more than 100 million lines of COBOL on their own. The application owners continue to add new code to existing COBOL applications, and modern COBOL programs can use modern software support technologies such as GUIs, Web interfaces, and relational databases.

Historically, the mostly manual process of analyzing large COBOL systems is slow and often misses critical information regarding the impact of changes. Inaccurate or incomplete analysis can jeopardize the entire project as missed requirements and the resulting rework inflates timelines and project costs.

Why is extending COBOL legacy systems so hard?

- **Sheer Size:** COBOL systems of interest are enormous and complex.
- **Technology Diversity:** Most COBOL systems have multiple dialects of COBOL, database structures, on-line and batch processes, report writers, impeded third party products, etc...
- **Information spread:** A large system in essence has a huge specification and has a vast design relationship between the specification and the code. Engineers attempting to change the system must understand why the system is structured the way it is to effectively make changes.
- **Software Entropy:** Large systems may have many layers of complexity represented in 30 years of code extensions resulting in dead code and redundant processes, resulting from so many years of change.
- **Legacy System Analysis tools are inadequate:** The only information legacy systems analysts have available are dusty design documents of dubious accuracy, out-of-date code documentation, and string/text based scan utility tools for hard to read source code.
- **Data expansion and truncation rules of COBOL:** Straight forward IT maintenance tasks like expanding a field on a report can represent a significant challenge. A change in the physical representation of the data field value can have far reaching impacts across the entire application. COBOL's automatic expansion/truncation rules can easily lead to data corruption if all related data

fields which hold a target value aren't modified to ensure adequate space is allocated. Finding and analyzing all possible data impacted can be quite difficult as field names are not always easily related to their contents, and working fields often have "generic" names. In addition, data may be processed indirectly through COBOL Redefinitions, or Group operations, and may be passed between programs using different names. What begins as a simple change can often take weeks if not months in testing and rework if impacted data fields are missed during analysis. Many organizations have experienced this first hand and spent millions of dollars on projects such as Y2K, HIPAA rules or the Euro conversion where financial non-truncation validation is required.

Integrating COBOL

Limitations of COBOL have led most organization to choose more modern programming language technologies such as Java or C# for their new applications development projects because of the significant advantage provided by object-oriented engineering methodologies that these languages were designed to support. Consequently, integration is costly and difficult due to the fundamental differences of the legacy COBOL and modern software technologies. Thus the no-win decision IT executives face every day: Is it better to replace or extend the life of legacy technologies? Pay the significantly higher cost of support and integration, or attempt to convert the legacy system into a modern one? Of course, conversion can jeopardize the organizations ability to replicate hard-won business rules woven into code.

Changing the COBOL analysis paradigm

evolvelT brings automation and interactive documentation to the traditional manual process of COBOL analysis.

evolvelT works much like a compiler creating an interactive centralized model from the systems source code that brings visualization to component relationships, system structure, data dependencies and the business logic/rules embedded in the code.

Where evolvelT really sets itself apart is handling the variants and uniqueness found in COBOL systems and by detecting and understanding data as processed within COBOL. evolvelT is unique in its ability to understand data by its offset and length and trace the impact of changing the data through its relationships and redefinition of the data in COBOL code. evolvelT is superior in its' ability to do impact analysis by finding data and all associated usage even when data is being changed by a different name or has been passed across program boundaries.

For more information about evolvelT, contact blackboxIT at: sales@blackboxIT.com.