To endure or to modernise?

Feb 6th, 2018
In Digital Business,

The dilemma of a legacy system. By Sergey Bludov, Senior Vice President of Media and Entertainment Practice at DataArt.

DataArt is engaged in industries where technological advances have made the burdens of legacy systems painfully acute. Over the years, our teams have been involved in modernizing legacy system architectures and we are happy to share some answers to common questions.

What is a legacy system?

English dictionaries define the word “legacy” as something left over or handed down by a predecessor. In computing, the concept usually denotes or relates to software or hardware that has been superseded but is difficult to replace because it is in use. In a technologist’s everyday routine, what people mean by “legacy system” or “legacy code” may vary, but it is rarely a compliment.

Common attributes of a “legacy system” include:

- Lack of comprehensive technical documentation describing how the system is designed, what are the core concepts, etc. Often the original developer(s) of such systems are no longer working on the system and such technical knowledge is essentially lost.
- Brittle codebase, where changes in one part of a system may cause problems in seemingly unrelated areas; such systems are typically extremely hard to extend and maintain.
- Obsolete technology stack or development methodology.

Some industry experts, such as Michael Feathers - the author of “Working Effectively with Legacy Code”, go as far as to say that a “legacy system” is any system without sufficient automated test coverage. So, according to this particular definition, even a brand-new system may be considered a legacy system if it was developed without the use of proper development methodologies.

Which industries suffer the most from legacy problems?

It's hard to say that any particular industry suffers more or less than others. Any long-standing business will ultimately face the dilemma – to keep and maintain its systems or to modernize. And the longer the solution is postponed, the more costly and painful it becomes. Legacy code is a natural consequence of continuously evolving technology. Therefore, consistent, on-going modernization must become a core strategy of any business that wants to stay relevant in the long term.

The areas where legacy issues are perhaps most noticeable and cause the most pain are the ones most affected by the digital revolution. The music and publishing industries certainly belong to this category. For example, missing metadata, old formats incompatible

with modern systems, or a lack of standardization are notorious, and they are in part caused by the continued use of legacy systems.

When does the legacy system actually become a problem?

The legacy code is not a problem in itself, as it may simply be an evolutionary stage that the system will pass. The IT community has been dealing with the necessity of supporting legacy code for years and it's a part of routine responsibilities of many engineers. Legacy code does, however, hinder progress and slow down innovation.

While support can be cumbersome, a legacy system only becomes a real problem when a company has to extend its features and functionality, or when trying to integrate it with new infrastructure or some other modern software.

Legacy systems are typically very hard to change or extend: it takes significant development and QA effort to implement and test the required changes, increasing the total cost of a new release. A fast-paced release schedule and shorter user feedback loop are proven success factors for end-user satisfaction, but are difficult or economically infeasible to achieve with a legacy system.

A brittle codebase also results in low quality releases and a fairly large number of major bugs discovered after a release is delivered to consumers. Such issues are much more expensive to fix, than if found at earlier stages of the product lifecycle.

How to solve my legacy system issues?

There are 3 major ways to approach the “legacy system” problem:

- To Refactor

The legacy system is gradually restructured to be more maintainable. When a new change is expected, a timeframe is planned to refactor the affected parts of the system, before new code is written. The work starts with “characterization” - automated tests that capture the existing behavior of the system, before modifications are made to ensure that the changes will not harm the existing processes.

- To Rewrite

A completely new system is built from the ground up with or without a plan to eventually migrate all existing users of the legacy system to the new one.

- To Retire

The active development of new features ceases and the legacy system is switched to a “maintenance only” mode, when only critical bug fixes are released.

There is no single solution that fits all and a particular approach can be applied depending on existing business needs.

Often “rewriting” the system may seem easier than refactoring, but there are strong arguments against it. A complete blank slate rewrite of any non-trivial system is a risky endeavor with a long and not-well-predictable time frame to complete. Few organizations can afford to discontinue all activities on an existing system due to the changing business environment, pressure from competitors, and consumer demands. So both the legacy and the new systems will have to be developed and maintained simultaneously, with the legacy system having an “edge” over the new system, in terms of implemented and tested features. More importantly, without a company-wide shift in IT strategy, there is no guarantee that a new system will work any better or be of higher quality than the legacy system.

Generally, the “refactoring” approach should be the first option to consider. Ultimately, it’s up to the business stakeholders, with the help of advisors and development team, to evaluate the situation, assess the risks and decide on the best course of action.

What’s the best way to handle the change – in-house or by a third party vendor?

The answer requires a thorough consideration of numerous factors.

It makes sense to keep the job in-house if the company has an adequately-sized IT department and its engineers are already familiar with the workflow and features that need an upgrade. However, the engineers who created the system and are most familiar with its architecture may not be available; or the in-house team may not be sufficiently staffed for the scope of the project; or the engineers may not be motivated to work with the legacy code; or an outside team may be more cost-effective. The list of factors is long.

The IT market is large and there are many companies that can help with development and support legacy code. If you should decide on a vendor collaboration, keep in mind that working with an in-house team and third-party vendors does not have to be mutually exclusive. In fact, many companies choose to augment their teams with outside experts and such collaborations can prove beneficial in many cases, such as dealing with legacy systems, keeping pace with business expansions, and more.

Original article can be found here: https://digitalisationworld.com/article/53302/