Problematic legacy: prevention is better than cure

A guide to legacy systems

Adviescollege ICT-toetsing

Problematic legacy: prevention is better than cure

A guide to legacy systems

Executives and users often view old IT systems within their organizations as a straitjacket. Modifications to these legacy systems, for example to support new policies, are frequently perceived as problematic. The media also regularly report that outdated systems prevent the implementation of new legislation, particularly legislation intended to support citizens in vulnerable situations. Legacy systems may also be more susceptible to cyberattacks. Some rely on programming languages or software development platforms for which technical expertise has become scarce and vendor support has become expensive or has been discontinued altogether. This issue is not limited to government organizations; banks, insurance companies, and large technology firms also struggle to adapt their extensive existing systems. In this guide—based on our research—we provide three recommendations that demonstrate how organizations can better manage legacy systems.

Many IT systems that have been in use for decades operate reliably and provide extensive functionality. They are essential for implementing policy and supporting organizational operations. Whether legacy is problematic, in part or in whole, is a nuanced matter and varies by situation. In our research, we certainly encounter problematic legacy systems. However, we also observe that legacy is often blamed for delays or an inability to implement changes without the actual cause being investigated and substantiated. In such cases, executives and sponsors decide to undertake large-scale legacy replacement based on assumptions and perceptions. *Our first recommendation is therefore: develop a thorough understanding of your systems.*

Organizations often lose sight of the value that an existing system provides, despite the many functionalities it has delivered over the years. They opt to replace the entire system, even though this is costly, risky, and not always necessary. Unnecessarily large projects are launched and subsequently run over budget and behind schedule. *To prevent this, our second recommendation is: choose the smallest possible solution*.

When organizations are faced with genuinely problematic legacy systems that require substantial replacement efforts, they often fail to reflect on how they arrived in that situation and what must be done to prevent recurrence. Underestimating, undervaluing, and underfunding maintenance and management activities, a lack of proactive evolutionary renewal, and in some cases years of neglect are insufficiently recognized. This leads to our *third recommendation: prevent problematic legacy from arising in the first place.*

1. Develop a thorough understanding of your systems

A sound understanding of existing systems is the foundation for maintaining them sustainably. Without this understanding, decisions are made based on perceptions rather than facts, investments may be initiated too late, legacy systems may be incorrectly labeled as problematic, and systems may be retired unnecessarily. It is therefore important to obtain factual insight into the strengths and weaknesses of both the systems and the management organization. What are the urgent issues, and what risks exist over the medium term?

A proper assessment of a system and its adaptability to changing requirements should be approached systematically. Consider the technical aspects and the functional quality of the system (how well it supports

operational needs), knowledge retention in both the present and near future, and cost efficiency. These aspects should be analyzed in relation to one another. We elaborate on them below.

a) Develop a thorough understanding of your systems

In our research, we frequently encounter situations in which technical issues are the reason a system is classified as problematic legacy. For example, technical debt may have accumulated over time because less robust solutions were previously chosen to save time or money, or because technical updates were postponed. Insight into technical aspects is therefore essential. Since a system never operates in isolation, interfaces and integrations with other systems must also be included in the analysis.

Analyses that provide a solid foundation should address the following:

I. Technologies in use
Determine how long the technologies currently in use are likely to remain mainstream. This includes infrastructure (servers and networks), database management systems, system software, software libraries, supporting software, and programming languages. Also consider developments in the supplier market. For example, are new market entrants emerging or acquisitions taking place? Is a vendor moving to the cloud, and do customers have a choice or are they required to follow? Consider the risks associated with obsolete technologies or market developments, particularly in areas such as security.

II. Adaptability
An important indicator of adaptability can be found in a system's history. Analyze how easily modifications have been implemented to date, how effectively those changes could be tested, and whether past modifications resulted in incidents. This historical information can often be derived from version control systems, change management records, and incident logs. Source code analysis and measurements of system size and maintainability can be useful, but they should always be supplemented with historical information to obtain a complete and reliable picture. Understanding future technical requirements helps determine whether the current technology will become a barrier.

III. Operational performance
Operational performance concerns the stability, reliability, and performance of a system over time. Are agreed service levels still relevant, and are they being met? Have there been any information security incidents? Available records can help establish a factual picture of a system's quality and operation. Relevant sources include technical monitoring data, logging information, availability metrics, incident and problem reports (including resolution times), and source code analyses. Do not focus solely on high-impact incidents.

b) Determine functional quality

It is equally important to obtain a clear picture of a system's functional quality. If a system no longer provides adequate functional support, this will affect the effectiveness and efficiency of its users. To assess functional quality, we recommend at least the following:

I. Determine whether users receive the support they need
Inventory the functionality provided by the system and identify the user groups involved. These may include not only organizational staff but also citizens or businesses. Determine whether the functionality sufficiently supports expected or desired activities, or whether gaps have emerged over time.

Practical example: inadequate user support comes at a cost

Functional support is not always evaluated. In one government organization, a budget must be distributed among various institutions based on a large number of legally prescribed criteria. To accomplish this, the organization uses a system in which these legal criteria are embedded. However, users do not trust the system. As a result, every month all amounts allocated to each institution are manually verified using Excel. Although discrepancies are almost never found—and when they are found, they can be explained—the manual verification process continues.

To gain insight into functional support, an overview of outstanding change requests is a useful tool. The number of issues and workarounds surrounding a system also indicates the extent to which it supports current business processes. Examples include spreadsheets or other supplementary tools. By discussing future changes to organizational processes, it can be determined whether the system will continue to meet future needs.

II. Assess whether the system complies with Laws and regulations
Determine which laws and regulations the information system must comply with, either now or within a specified period, and assess the extent to which the system meets those requirements. Look ahead to upcoming legislation. Consider not only Dutch legislation but also European regulations. This includes not only sector-specific legislation but also requirements such as the General Data Protection Regulation (GDPR) and information security standards. Investigate whether data is stored consistently and whether it is cleansed or archived in a timely manner.

Practical example: count the blessings of a maintainable system

The Advisory Council on IT Assessment (AcICT) conducted a review of the maintenance and management of a case management system used by an executive agency that had been in operation for fifteen years. The review found that the system provides strong support to its user organization. It also offers a valuable foundation for further development. The system is reliable, feature-rich, and maintainable. However, the review identified the need for greater attention to information security and privacy risks. It also concluded that further system development should be aligned more closely with the organization's ambitions. Our recommendation was therefore: retain the system, while addressing these areas for improvement.

c) Determine whether available knowledge is future-proof

The availability of knowledge is essential for the continued use of aging IT systems. This includes knowledge of the technologies in use, the system itself, and the business processes it supports. Assess the level of available technical and functional expertise both within and outside the organization. Also determine how many people with specific skills will be required now and in the future. Can suppliers provide the necessary expertise? Can organizations collaborate with one another? Can staff be trained? In addition, investigate whether sufficient high-quality documentation exists to support the onboarding of new employees or the transfer of system management to a new supplier.

d) Gain insight into system costs

Analyzing the costs of a system helps determine whether sufficient investment is being made or whether underfunding exists. Underfunding shortens a system's lifespan and increases the likelihood of problematic legacy emerging. Conversely, organizations sometimes continue investing in systems that provide relatively little value. We often observe that insight into costs versus organizational value is incomplete; only part of the total cost picture is known. To improve financial transparency, the following steps are essential:

I. Identify current costs
Identify the costs required to keep the system operational and adaptable. These may include: License fees, Maintenance contracts, Housing and hosting costs, Personnel costs for management, developers, administrators, and help desk staff. Do not overlook costs incurred elsewhere in the organization, such as infrastructure expenses (locations, servers, networks, and storage) and depreciation or replacement costs. Although these indirect costs can be difficult to estimate, they should nevertheless be included.

II. Assess whether costs are reasonably proportional to benefits
Determining whether costs are still justified by a system's benefits or added value is not straightforward. Nevertheless, useful reference points exist. For example, compare costs with those of systems used by other government agencies or public-sector organizations. Do these organizations incur higher or lower costs, and can the differences be explained? The cost of implementing changes (for example, per function point) can be compared with industry benchmarks. It is also useful to assess whether annual maintenance and support costs are proportionate to the replacement value of the system. Most importantly, analyze where cost outliers occur and estimate whether those expenses could be reduced through alternative solutions.

2. Choose the smallest possible solution

When the analysis of the current system shows that it, or parts of it, may cause problems, has become less useful or unusable, is becoming a risk, and/or cannot be adapted within a reasonable timeframe and at reasonable cost, it is time to take action. To do so, we recommend the following steps:

a) **Develop several solution options**
Formulate alternative solution approaches, ranging from continued enhancement of the existing system to complete replacement, along with several realistic options in between. Weigh the advantages and disadvantages of each alternative against one another. Base this assessment on the insights gained from the analysis described in Chapter 1. For every alternative, the guiding principle should be: keep the project as small as possible. Our experience shows that solutions implemented in small increments, with regular evaluation along the way, offer the greatest chance of success. Below, we outline a number of problems we frequently encounter, together with potential solution approaches.

I. The analysis may reveal that the system has accumulated deficiencies that make changes more difficult to implement, while still being entirely suitable for renovation. In such cases, a good strategy is to undertake a renovation effort aimed at repaying the technical debt that has accumulated over the years. If a system runs on an outdated technical platform and/or obsolete hardware, the solution may be to migrate it to a new platform with only limited modifications to the system itself.

Practical example: a surgical approach to a localized problem

A public service organization operates a large application that supports the execution of its statutory responsibilities. The application relies on a hierarchical database—an outdated database model. The limitation for future enhancements is not the application code itself but the database. Rather than replacing the entire application, the organization decides to replace only the way in which the data is accessed.

II. A programming language may no longer be widely used, or the necessary expertise may no longer exist within the organization. In that case, converting the code to a more widely supported programming language may be a solution. However, this option carries considerable risk, especially when the conversion is performed on a large scale and in a fully automated manner. We therefore do not recommend this approach without careful consideration. A phased approach, potentially involving manual conversion steps, is generally preferable.

Practical example: not everything can be automated

A public service organization uses a large application to distribute monthly payments. The application was built on the now-obsolete operating system OpenVMS and is written in COBOL. Expertise in both OpenVMS and COBOL has become scarce. The organization therefore decides to replace the application with a functionally equivalent system written in Java. Initially, an automated conversion approach is selected. It quickly becomes apparent that this results in Java code that is excessively large and difficult to maintain. As a result, alternative approaches must still be explored.

III. The analysis may show that a system no longer meets requirements in so many areas that renovation is no longer a realistic option. In such situations, a complete rebuild, development of a new system, or procurement of a commercial off-the-shelf package may seem like the obvious choice. Although attractive at first glance, this approach can be highly risky. For this reason, we recommend avoiding a "big-bang" approach, as the project risks associated with such initiatives are often substantial. For large systems, we observe failure rates of up to fifty percent. Incremental rebuilding offers the highest probability of success. Each step remains manageable in size, making the overall project easier to control.

Practical example: you eat an elephant one bite at a time

A public service organization operates a large application built twenty years ago using a code-generating development environment. That development environment is no longer widely used. Vendor support has become prohibitively expensive, and global adoption continues to decline. The application consists of many small modules. The organization chooses an approach in which modules are rebuilt one at a time. This strategy has a high likelihood of success.

IV. When expertise in a particular platform or technology is declining in the market, investing in knowledge—through training and education, for example—can be an effective solution strategy. This option is often overlooked. Regardless of the chosen strategy, internal expertise remains essential, since organizations are often dependent on their existing systems for many years to come.

Practical example: good software can be reused

During an assessment of the replacement of a safety-critical system, it becomes clear that the software itself is of excellent quality. As a result, replacing both the software and hardware would be unnecessary and would introduce additional risks. The organization can instead replace the obsolete hardware and migrate the existing software to the new platform. At the same time, it invests in training new employees.

b) **Determine the Costs and Benefits of the Solutions**
For each alternative, identify both the one-time costs associated with development and implementation and the ongoing operational costs. Also estimate the expected benefits—both financial and non-financial—as accurately as possible. Together, these elements form the business case.

Ensure that the business case is complete and avoid common pitfalls by following these guidelines:

- Develop and evaluate a sufficient number of alternatives, including the zero scenario (continuing with the current approach).
- Investigate actual, observable problems when an organization wants to replace an IT system that cost millions simply because it is considered "old."
- Do not focus solely on software; include hardware considerations as well.
- Be cautious about overestimating the benefits of new technologies with which the organization—and sometimes the supplier—has little or no experience. This often goes hand in hand with underestimating the costs of developing and implementing new solutions.
- Include the costs of risk mitigation measures in the analysis.
- Consider historical realities. If it took twenty years for a system to reach its current level of functionality, it may be unrealistic to plan a complete rebuild within two years.

c) **Decide on the best solution for the present and the future**
Based on the information gathered in the previous steps, make a well-reasoned decision regarding the preferred solution. Before making the decision, establish the criteria on which the decision will be based. When evaluating alternatives, consider both Operational risks, such as continuity, reliability, and compliance; and Project risks, such as experience with the technology, availability of expertise,

and project timelines. Document the decision-making process so that it remains transparent and traceable.

For further guidance on developing business cases, we refer readers to our Dutch publication 'Bezint eer ge begint'. For guidance on benefits management, we refer readers to the Dutch publication 'Wat je zaait, wil je oogsten'.

3. Prevent Problematic Legacy from Arising

To prevent future legacy-related problems as much as possible, it is necessary to establish effective system maintenance practices and implement sound lifecycle management. The measures described in the previous recommendations improve the quality of a system in terms of its technical characteristics, functional quality, available knowledge, and cost efficiency. New developments may subsequently affect that quality, either positively or negatively. It therefore remains important for organizations to maintain insight into the current status of their systems. Well-designed lifecycle management helps achieve this objective. Based on our research, we offer the following recommendations.

a) Maintain Knowledge of the System

In existing systems, we see that the availability of technical and functional expertise within the organization—or access to such expertise in the market—has a significant impact on an organization's ability to remain agile and implement new requirements. This is true regardless of whether mainstream or innovative technologies are being used. It is therefore essential to ensure that sufficient knowledge holders are available for critical systems. These individuals should be able to replace one another when necessary, and workforce planning should take staff turnover into account. A few practical recommendations:

- Organizations should understand which technical expertise is required for the various components of a system, both now and in the future. This may include knowledge that the organization wishes or needs to retain internally, as well as expertise that can be sourced externally. Collaboration with other government organizations may also be an effective way to jointly develop and maintain expertise.
- Functional knowledge must be safeguarded. This concerns the operation of an information system in relation to the business processes it supports. Relevant topics include user functionality, interfaces and interactions, information and data flows, integrations, and dependencies. In older systems—but also in new systems developed using agile practices—there is often a tendency to neglect the maintenance and documentation of this knowledge. As a result, resolving issues becomes more complex and expensive.

b) Keep systems healthy

A healthy system can continue to serve the organization effectively for many years, provided that both the system owner and the IT support organization undertake sufficient activities to remain technically current and ensure that the system continues to meet business needs. This requires attention to the following aspects:

- Technical health involves keeping all system components up to date, including hardware, software, networks, and interfaces. Avoid costly extended support arrangements by planning upgrades and updates in a timely manner. Also ensure compliance with new or revised government standards, for example in the area of information security. Organizations should allocate sufficient time and resources each year to carry out the activities required to maintain the technical health of the system.
- A functionally healthy system continues to support the work for which it was designed effectively. Achieving this requires ongoing attention and maintenance. Regular and productive consultation with users, combined with their active involvement in further system development, is essential for maintaining alignment with business needs. In addition, compliance with applicable laws and regulations should be verified on a regular basis, for example through targeted assessments or audits.

Not every requirement or request should automatically be implemented. Only changes that genuinely add value should be introduced, and they should be managed in a structured manner. This requires a mature and controlled change management process in which proposed changes are evaluated from multiple perspectives. Such an approach prevents the accumulation of unnecessary additions and modifications that gradually make a system difficult—or ultimately impossible—to maintain.

c) **Ensure sustainable and appropriate funding**

Preventing legacy-related problems requires a multi-year plan for ongoing operations, maintenance, and system evolution, supported by corresponding long-term funding. Organizations should ensure multi-year budget coverage for these activities and periodically reassess funding requirements. One approach is to calculate annual budgets as a percentage of the replacement value of the system. Plans and assumptions should be reviewed regularly and, where possible, benchmarked against comparable organizations.

In addition to funding for ongoing development, organizations should also allocate resources for innovation. Where system evolution builds upon existing capabilities, innovation focuses on introducing new technologies, methods, or applications to fundamentally improve a system or create entirely new possibilities. To maintain a clear and accurate picture of the structural costs of operating systems, we recommend that organizations establish separate funding streams for innovation initiatives.

This publication is issued by:

Advisory Council on ICT Assessment
info@adviescollegeicttoetsing.nl
www.adviescollegeicttoetsing.nl
P.O. Box 16292 | 2500 BG The Hague | The Netherlands

March 2025

Adviescollege
ICT-toetsing