New Life for Legacy Systems

Nearly every higher education institution depends on a core administrative student information system (SIS). Because the SIS sits at the center of so many of the day-to-day operations of managing students, courses, and grades, it becomes extremely important, expensive to operate, and hard to change. Every software system has limitations, of course, and system administrators soon find they need to either modify the way they work or modify the SIS. Because changing the way people work requires changing human behavior, changing the software is often simpler and more expedient.

Over time, these changes accumulate. Eventually, the resulting complicated and deeply embedded system can no longer support modern interfaces and new ways of doing business. At some point, campus leaders find themselves investing in the complex, risky, expensive, and politically fraught process of replacing their SIS in the hope of providing better service to students, improved access to data, and a more flexible technology environment for the future. The drumbeat to “replace the SIS” simply becomes too loud to ignore.

The California State University (CSU) operates a complex and expensive SIS environment, with a single software system currently servicing 22 campuses. While these campuses all use one product (PeopleSoft from Oracle), the code base for the system is modified first by the central system office and then by the various campuses, representing 15 years of accumulated changes in some cases. In this environment, there’s a natural appeal to the idea of tossing it all out and starting over. However, after a careful review, technology, administrative, and academic leaders decided in 2018 to postpone consideration of a new SIS system, partly because they felt that the current generation of Software-as-a-Service (SaaS) systems was not mature enough to meet institutional needs and that it is too soon to know whether the emerging generation of SaaS systems will fulfill the promise of better service and longer-lasting technology.

In the meantime, the CSU needed to address the demands for digital transformation and improved student outcomes. Was there a way to rethink the existing system while also preparing for the time when the old technology will no longer be tenable? We started by considering the ways in which the legacy SIS was inadequate or suboptimal. We identified a number of common frustrations and are embarking on strategies to address most of them. But perhaps most critical was the issue of integration. Integrating legacy systems, which are designed to work as monoliths, with modern SaaS systems in use in other campus areas can be difficult and costly. So we developed a way to significantly mitigate this problem. Although there are some special circumstances at the CSU, our overall strategy and tactics can be emulated elsewhere, including at many smaller institutions. In fact, if these strategies can work at the CSU, they can work almost anywhere else.

Integration refers to the connections between the SIS and other campus systems, including learning management systems, human resource systems, library services, parking and dining systems, and system directories such as Microsoft Active Directory. Traditionally, these integrations have been built one of two ways—through the export and import of “flat files,” consisting of lines of data separated by a delimiter such as a comma, or through the modification of the SIS code to communicate directly with the outside system.

The flat file, or “batch,” approach to integration is relatively easy to implement, although over time, managing these files can incur significant overhead and add new security and reliability risks. In addition, feeds typically operate on a daily schedule, meaning that a change in one system won’t reach a downstream system until the next day, resulting in poor student service. For example,
a student might return a library book on Tuesday afternoon to clear a hold, but the hold might not be lifted until the information reaches the SIS on Wednesday morning. Adding more frequent updates may be an option but may also be difficult or costly. Student services operating on a 24- or 48-hour schedule don’t meet the expectations of today’s students or their families.

Modifying the underlying code of the SIS may address the need for timeliness. But it is far more costly in both the short and the long term. Modifications to an SIS need to be carefully tested and sometimes updated every time the SIS code is updated by the vendor. As the number of modifications grows over the years, the update process becomes progressively more complex, accumulating technical debt along the way.

An alternative strategy is to create an intermediate data repository, which can consist of a simple set of files, often called a “web view,” or a more complex database that may do double-duty as a data warehouse. While this strategy can provide a partial solution, it falls short of current requirements in two ways. First, these systems typically are updated only periodically, so the information is almost always stale and thus cannot respond to users’ needs in real time. Second, a data repository is a one-way strategy—systems can access SIS data, but they can’t update it.

Modern software is built using a different model, in which individual programs can communicate in real time, or with a small delay (“near real time”), using a well-defined set of interfaces called an Application Programming Interface (API). The most commonly used approach to APIs, called RESTful, provides a simple model of application interaction consisting mostly of GETs and PUTs. For example, to find out whether a library book with a particular identifier is currently checked out, a user might send a GET Library-Book-Status (ID) message to the library management system and receive a message “On Loan,” indicating that the book is checked out. By composing GET and PUT messages, two systems can communicate without requiring flat files or code modification.

What if we could do the same with the legacy SIS? Some colleges and universities have done so, implementing software systems that can translate the data and business functions of the SIS into these RESTful APIs. Brigham Young University has made notable strides in the area, for example. And a number of other institutions have also created APIs. A list maintained by Kin Lane identifies nearly 50 colleges and universities around the world with API initiatives, many including SIS functionality. Still, a standard API model would simplify integration with third-party software. IMS Global Learning Consortium has initiated the development of EDU-API, which is intended to provide a standard API for accessing higher education systems, not just in the United States but internationally. The payoff for success in reducing interface costs for institutions, and for vendors, would be significant.

There are two other advantages of an API-based integration strategy. First, implementing integrations via API can be accomplished much more quickly than traditional development. Not only are well-designed APIs simple and easy to use; they also isolate the add-on development from the SIS, reducing the risk that an integration will cause a performance problem or will damage the integrity of the SIS. Second, API-based programs can give veteran software developers the opportunity to work with modern tools, and by eliminating the large amounts of specific knowledge needed to work in an SIS, they multiply the number of individuals who can develop campus software. This has the advantage of making the IT organization less of a bottleneck and encouraging a wide range of innovation in applications and interfaces.

Creating a comprehensive SIS API supports not only integration in general but also the development of an improved student, faculty, and staff user experience. Arguably, access to the SIS via an API is more flexible and “future proof” than choosing a new SIS, which will have an embedded model of user interaction that may or may not be consistent with the overall user experience a campus is creating for its constituents. Interaction requires two-way communication with the SIS, so APIs need to include both GET and PUT capability, and they need to be real-time or near real-time.

The CSU is in the early phase of developing an API that will simplify integration with our PeopleSoft system across 23 campuses by 2021, eventually extending beyond IT departments to include staff, faculty, and even students who want to develop innovative interfaces. By developing an SIS API, we will greatly simplify SIS interaction and abstract it from the underlying software; eventually, when most or all interfaces are via API, we will be able to change the underlying SIS while minimizing the impact on these integrations, thus reducing the cost of a future SIS replacement.

For higher education institutions, replacing an SIS may become inevitable in the long run. In the meantime, using strategies to extend its life while eliminating many of its weaknesses can represent good stewardship of institutional resources.

A longer version of this column can be found online at https://er.educause.edu/columns/new-horizons.

Notes

Michael Berman (mberman@calstate.edu) is Chief Innovation Officer and Deputy CIO at California State University, Chancellor’s Office. He is the 2019 Editor of the New Horizons column for EDUCAUSE Review.

© 2019 Michael Berman. The text of this article is licensed under the Creative Commons Attribution 4.0 International License.