uniform

Uniform whitepaper

Switching vendors for digital architectures without replatforming

Change can be exciting, but it can also present major technical issues. In the past, to overcome technical obstacles or adopt new technologies, companies turned to replatforming, enhancing their digital-experience platform by switching vendors. That's a challenging, dramatic change that takes months or years. In a world where new technology emerges monthly, those projects can be doomed before they even start.

The cost of replatforming

Even in clear-cut cases, many companies are understandably reluctant to replatform, which can be extremely costly and time consuming, adding no immediately obvious business value during and after the process.

Consider this scenario: During replatforming, engineering executives cannot have the tech team start from scratch. The team must still maintain the existing platform to sustain revenue while others, usually working with an agency or consultants, plan and implement the new architecture. As a result, an expensive "double stack" arises, which is complex to manage with teams strained by additional workloads, skills to learn, and concerns about their roles with new systems.

Replatforming typically months of scoping and vendor meetings to build the architecture, then six to 12 months of intense development to complete. Plus, companies cannot truly move forward during the process, let alone that it takes time to redefine workflows and retrain people after replatforming is complete. Ironically, the motivation to be more future focused and competitive could actually slow progress and erode your competitive edge for quite some time.

Moreover, the post-replatforming benefits can be short-lived since, during replatforming, challenges might surface that the new platform cannot handle. Simultaneously, retraining teams could be more time-consuming and expensive than anticipated, especially if they are accustomed to a previous workflow. Also, the more complex the system is, the harder retraining becomes.

Cases in which replatforming is the only choice

Nonetheless, cases exist where replatforming is unavoidable, e.g.:

- The technology you depend on has reached its end of life. If you stick to the bitter end, you might find it difficult to locate qualified developers for your current platform, and they demand higher pay. Not only that, support options from the vendor might be limited or even unavailable, as well as the security issues of relying on unsupported software.
- Black-box technologies that were once turnkey and convenient are now impossible to integrate or repurpose.
- The constantly changing business climate renders a company's platform incompatible

with its business requirements. If no part of your system is worth integrating or repurposing, you must start afresh.

Other options

Separately, many old rationales for replatforming are now invalid. For example:

- If your current system is more user friendly than developer friendly—or vice versa—you might be able to add tools to the mix for a better balance without abandoning the system.
- If your system's advanced age makes it tough to recruit qualified technicians to maintain it, you might be able to combine the team you have with vendor support to retain the system while gradually replacing it.
- If the entire system is becoming too expensive to operate or if the vendor changes its licensing model, which often results in higher costs, consider decoupling its components, retaining only those that can still run at an acceptable cost.
- If integrating your old system with a new one seems unduly expensive or the workflows involved are so complicated or the technology is so complex that integration seems impossible, see if the old system offers newer, more integration-friendly, and cost-effective versions. Bear in mind, however, that switching to a newer version of a vendor's product usually results in higher costs and could involve upgrade work that is comparable to replatforming.

To their credit, most monolithic systems have added features, such as headless capabilities and support for GraphQL, to address some of the pressing needs that would otherwise drive customers to replatform. Unfortunately, those enhancements take time to implement and have conspicuous downsides. For instance, to leverage a monolithic system's new features, you must upgrade to the latest version, for which the vendor might be charging a premium. Additionally, these features are always 'add ons' and unlikely to perform as well as a platform that is explicitly built for the modern features. A successful upgrade might require extensive testing—a complex and time-consuming project itself. No matter how you cut it, you'll likely end up paying more for upgrades that may not give you the results you need.

Pluggable components

Adopting products tailored for integration or incorporation into a **composable architecture**, aka composable products, is just the first step. Someone must connect them to make them work together—a task that's a major obstacle to that adoption because integration is expensive, hard to do right, and liable to require long-term maintenance of custom code. That's where Uniform comes in by serving as a bridge for connecting the systems in a composable architecture.

To succeed with such an architecture, however, you need more than just connected systems. Tools for creating and managing digital experiences must also be available to marketers and merchandisers. A digital experience composition (DXC), which offers the right tools for working with composable architectures, is well worth considering.

As a new category alongside "headless" and commerce systems, DXC separates the experience-creation process from the underlying back-end architecture and empowering business teams to take charge without help from developers. Another convenient feature of DXC is that, to add systems to your stack, you need not change its infrastructure. Instead, simply plug in components or replace the back-end system from which a component is pulling data.

Ultimately, you're free to choose your tech stack's components, assembling and deploying them at will.

Uniform as the leader of the composable revolution

Composable architectures boast momentous ramifications with the potential of unlocking numerous exciting possibilities for improving performance, scalability, and security while reducing costs. Instead of being locked into a tool or vendor that doesn't appeal to you, you're free to pick the best-of-need services. Also, rather than having to specialize in a system, developers can work with their preferred tools and languages.

Connecting those different services yourself is a huge risk. Not only is extensive custom coding required, but assigning developers such a task by no means ensures a consistent end-user experience, which is a discipline all by itself. That's where Uniform greatly reduces the speed of replatforming: By leveraging our prebuilt integrations, you can complete the process in only a day or two.

Most important, this modular composable architecture is extremely flexible. In addition to offering DXC capabilities that foster the productivity of business teams, Uniform aims to make composable mainstream by serving as the foundation on which brands can build their own composable architecture. That's because connecting composable systems, even though they are designed to be connected, requires a deliberate, difficult, and risky effort. Not only do you have to become adept with a slew of APIs, you must also write the code for the process. Plus, simply connecting the systems doesn't mean that the end product is immediately usable by business teams.

In addition, what about post-integration maintenance and support of such a complex product?

Uniform handles all that by providing the foundation for a composable architecture: You get the integrations for the systems and the tools for business teams without having to figure out how to tackle them yourself. You can then focus on the parts of the system that are unique to your business and on building features that attract customers and boost revenue.

Nonetheless, not all composable architectures are sustainable. Some of them might become "composable monoliths." In other words, selecting a bunch of composable products doesn't mean that you're going to end up with a truly composable architecture that conforms to the true spirit of composability.

What's more, the technical foundation for a composable architecture offered by Uniform supports modern digital production and delivery processes. Consequently, your teams can concentrate on their core competencies by building and delivering omnichannel experiences even as your business needs or customer preferences evolve. That is, technical, creative, and business teams can collaborate innovatively to deliver a modern production pipeline that doesn't require them to work in a specific manner. For example, business teams can start building digital experiences before their creative and technical counterparts have finished their work. Marketers can leverage Uniform's no-code tools to automate workflows and orchestrate tasks across multiple systems. Such a simple technique reduces complexity, improves efficiency, and fosters focus on value-generating activities.

Uniform as a MACH-certified member

Formed by a group of companies that have joined forces to forge a new standard for developing more scalable, reliable, and manageable applications, the MACH Alliance focuses on promoting microservices-based, API-first, cloud-native, and headless software. As a leading provider of composable orchestration, **Uniform, a MACH-certified member**, has stepped up its involvement with the MACH Alliance as **cochair of its Marketing and Community Council.**

Positive changes: Uniform and composable architectures

Not long ago, replatforming was unavoidable even though wasteful and potentially paralyzing. Uniform offers a path that spells its end. Change remains a constant, but with Uniform and a composable architecture, you can confidently tackle changes on your terms.

To learn more about Uniform and its important role in composable architectures, **contact us for a demo.**

uniform

www.uniform.dev

hi@uniform.dev