Product management in edtech:

Build vs buy decisions

Learnosity

Contents

Authors:
John Kleeman,
Founder of Questionmark, and EVP Business Development & Industry Relations, Learnosity.

Mark Lynch,
Co-founder and Chief Strategy Officer, Learnosity.

Introduction

With recent rapid changes in the edtech landscape—from the explosive growth in learning startups since the onset of the COVID-19 crisis, to the post-pandemic reduction in funding[1] and rising interest rates—there has never been greater pressure to deliver software in a timely fashion and still be profitable.

The goal of this paper is to give product owners a fair analysis of the advantages and disadvantages of using third-party components in edtech—a classic build versus buy decision. We'll also share some worksheets you can use to quantify the decision and work out the total cost of ownership (TCO).

We'll use the Learnosity Assessment Engine as an example because it's used by many learning platforms to author and deliver questions and tests. However, the approach discussed here can be applied equally well when evaluating other components.

Legacy code & tech debt kills progress

How do you sustainably fund your core product and simultaneously invest in the next big thing to push your business forward?

Long-term growth and value creation of edtechs requires the capability to do both efficiently and effectively. It's a delicate balance, especially as devs spend less than half their time writing functional code, with much of this dedicated to maintaining or enhancing existing functionality[2,3]. This limits the time available for new projects.

Simply put, choices have to be made.

With limited development capacity, most product managers don't have the freedom to build everything they want into even their most important applications.

Resources need to flow to where return is greatest—namely, to the features and capabilities that are highly valued and will differentiate your products from the competition.

> Devs spend less than half their time writing functional code

[1] Edtech Funding Falls Sharply
[2] Today was a Good Day: The Daily Life of Software Developers
[3] How Much Time Do Developers Spend Actually Writing Code?

Return, not burn

The granular data available from modern agile practices can be falsely reassuring when it comes to making product decisions. Knowing the estimated effort by way of story points and epics, velocity, and burn down and burn up charts may clarify how much effort a feature might take, but not where and how quickly a return will be made.

Time to value

Using the lens of time to value, product owners and engineers should work together to determine the appropriate resource allocation to achieve growth and margin goals. This focus helps to make clear-sighted decisions on new investments.

You want to use your own development team to build capabilities that are unique and help differentiate you in the market. On the flipside, it's best to buy third-party software that is considered table stakes in the market and meets well understood needs.

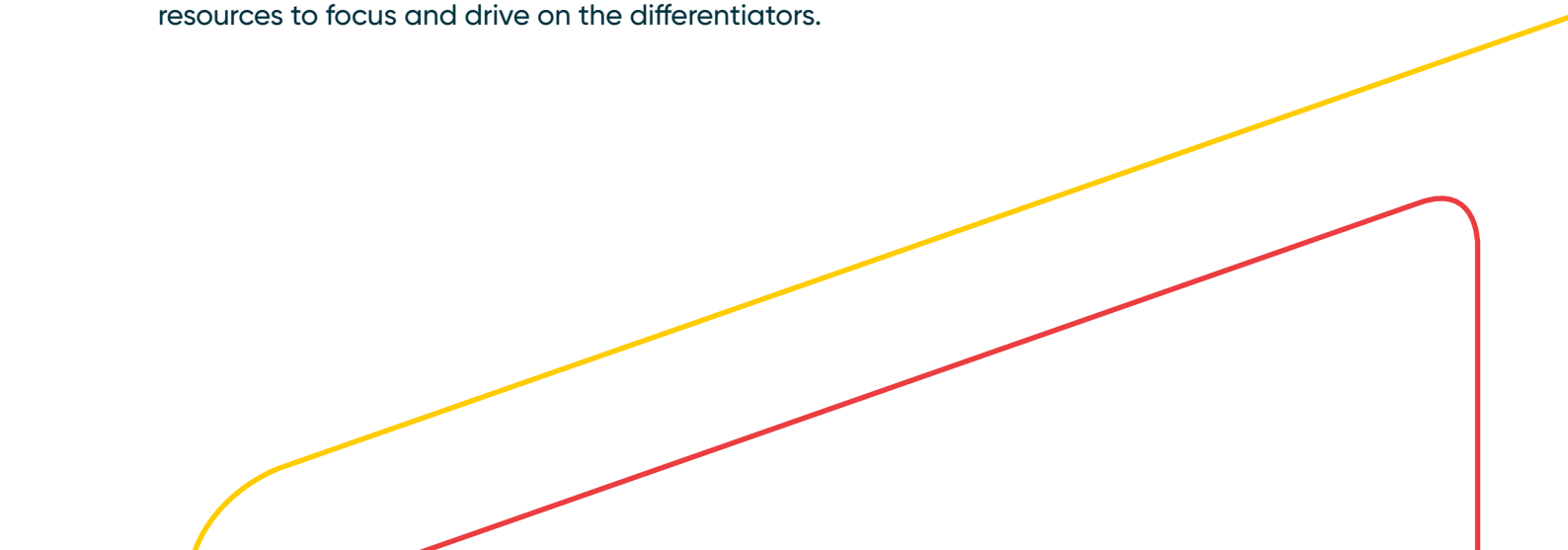
Buy versus build for assessment tech

When it comes to a classic buy versus build equation for assessment technology perhaps the biggest risk is underestimating a favored course of action. There are obvious concerns that will likely appear in your initial evaluations, such as ensuring user interfaces are compliant with accessibility guidelines and being confident your designs are responsive and work well on mobile devices.

But it's also easy to overlook issues that may not have an immediate impact on development. For example, will integrating a third-party solution support or compromise privacy efforts? If you build your own application, will it have the infrastructure necessary to cost-effectively scale with your success? Will your product be versatile enough to function not only in your own environment, but in a customer setting such as an Learning Management System (LMS) or mobile application?

Even team building is central to this issue. Building your own team means you have more control over timelines, but you also need to juggle team size to be sure your feature set is achievable with your release schedule. Assembling your own staff also means you pick and choose who works on which aspect of your application, but you may need to hire additional subject matter experts. This is common when the software you're developing is highly specialized (such as in the case of a math-scoring engine).

The right third-party software supports velocity and value. It can give a fully developed capability—better features, faster time to market, with less execution risk. It also liberates resources to focus and drive on the differentiators.



A framework for decisions

When it comes to what goes into your software product, every decision is strategic. While a decision may appear to be quantitative, qualitative factors should not be ignored—for example, the role of trust and security.

To account for these scenarios, we propose using two decision matrices. One is financial and shows the total cost of ownership, and the other is qualitative. Both are based on our experience with the Learnosity Assessment Engine.

Total cost of ownership

Below is our TCO worksheet. Included are some specific calculations, but you can input your own numbers to suit your particular circumstances. You'll need to decide on a timeframe. In the worksheet below, we suggest analyzing the cost of ownership over three years.

You'll need to determine your internal cost for development personnel in order to include the costs of employing and managing them. The worksheet assumes an average (blended) cost of US$7,500 per person per month, but feel free to adjust them to your circumstances.

	Build: Proof of Concept	Build: Basic Release Quality	Buy: Learnosity Assessment Engine
Scope	• 3-5 basic question types • No media or interactives • Limited authoring UX • Partial mobile responsiveness • No accessibility compliance • Manual testing • Not a hardened solution	• 3-5 question types • Some media or interactives • Basic authoring UX • Mobile responsive • Partial accessibility • Manual testing • Basic non-functional testing	• 50+ question types • Rich interactives and media • Fully featured authoring - expert & casual users • Mobile responsive • Accessible compliance (508 & WCAG) • Robustly tested software foundation • Proven scalable solution • Data security and IP protection
Estimated build effort	6 person months	24 person months	3 person months†
Developer costs ($7,500 per person month)	$45,000	$180,000	SaaS license + integration costs
Maintenance & support* (Total amount for years 2 and 3)	$18,000	$72,000	Included
Compute, storage & data transfer costs (3 years)	TBD	TBD	Included
TCO over 3 yrs	**$63,000**	**$252,000**	Talk to us

Table 1: Total cost of ownership

*Maintenance and support estimated at 20% of build cost. M&S includes bug fixes, feature improvements, adapting to new browsers/devices, and other maintenance/support activities. M&S costs kick in for years 2 and 3—assumption being year 1 is included in dev costs.
† Estimate for moderate complexity integration for standard/scale license

Summing up the three columns will give a TCO over three years of each approach.

An added layer of sophistication could be added to your model by factoring in the time to market (TTM) to building software. TTM is the time between the start of work and the first sale. While the typical unit of measurement is time for TTM, one could include earlier recognition of revenue lost/deferred by choosing a build approach.

One can further extend that idea by quantifying the value of other innovations/features that could have been developed. Applying a product portfolio approach[5], makes the cost of resource-allocation choices clearer. By using resources differently, new revenue generating capabilities could be developed.

Qualitative factors to consider

As flagged earlier, focusing on quantitative factors alone doesn't go far enough.

We believe that, for qualitative factors, it is important to consider some key categories; for example, functional fit, trust, and proficiency in the domain. For each category, try to frame some questions that are applicable both to an internal build and to an external supplier.

A table is provided below for you to adjust to your circumstances. Each question should be answered using a rating between 1 and 5, where 1 is a lowest score, meaning significant remediation work is required, and 5 is the highest score. Summing together the scores across categories will give you a comparator across qualitative factors, with the highest score being the winner.

> Applying a product portfolio approach, makes the cost of resource-allocation choices clearer

[5] McKinsey: The Innovation Commitment

Category	Build	Buy
1. Functional fit	**1-5**	**1-5**
Solution meets all "essential" requirements		
Solution also meets many "should" requirements		
Works on all needed mobile devices and browsers		
Learner user interface matches the rest of the product		
Administrator and author user interface matches the rest of the product		
2. Scalability/Performance	**1-5**	**1-5**
Scales robustly to meet peak load		
Solution provides a fast response time		
Handles low bandwidth and intermittent connectivity well		
3. Trust/Risk	**1-5**	**1-5**
Confident costs will remain stable over time		
Meets high availability requirements (99.9+%)		
Will support new browsers and devices		
Trusted to deliver a solution that is reputation enhancing		
4. Proficiency (expertise in the domain area)	**1-5**	**1-5**
Development team understands assessment requirements		
Team has practical experience working in this domain		
Team has skill and competencies to get accessibility right		
5. Data security and privacy	**1-5**	**1-5**
Security with external certification (e.g. ISO 27001)		
Team has commitment and capability to address security issues promptly		
Solution keeps content and other proprietary IP safe		
Application shields identities of learners to prevent identification in the event of a breach		
6. Able to support customers and learners	**1-5**	**1-5**
Proven software already in place being used by other organizations		
Coverage and quality of documentation		
Solution can be localized and translated		
Team is committed to and capable of promptly fixing bugs		
Total		

Table 2: Qualitative factors to consider for build versus buy

Conclusion

The decisions organizations make as to what they want to build—and consequently maintain—determines future success.

Building an application puts you firmly in charge of which features you can add and when. With that control, however, comes significant cycles of testing, documentation, training, support, and technical debt.

Development teams often prefer to build everything themselves and the dialogue between product managers and engineers moves to getting more from the team.

However, just because your development team can build something, doesn't mean you should. The key question to ask is what distinguishes your organization from others?

Where third-party software is available, and it meets your functional needs, it can offer a proven way of generating value faster and more efficiently.

Working with a third-party solution means your vendor is responsible for many of these phases, as well as keeping up with evolving standards such as security, privacy, and accessibility.The above framework will help you make the dialogue more effective.

Focus on building your competitive advantage and distinguishing capabilities,
buy in the rest where possible.

About Learnosity

Learnosity is the global leader in assessment solutions. Serving over 700 customers and more than 40 million learners, our mission is to advance education and learning worldwide with best-in-class technology.

Our APIs make it easy for modern learning platforms to quickly launch fully featured products, scale on demand, and always meet fast-evolving market needs. More at learnosity.com