Overcoming the Challenge of Technical Debt in Start-Ups and Mergers

How does technical debt begin? It spirals up from a combination of features that were never accounted for in the initial implementation, or as a result of poor implementation, and of patches rather than fixes in a technical product or process. How does this debt accumulate? Oftentimes, there's a rush to market: the pressure for a minimal viable product (MVP) plus a general lack of understanding about what the process should be contribute heavily to debt build-up. In the case of a merger and acquisition, technical debt rises because it isn't taken into consideration as a component of merging systems. The M&A industry is great at due diligence when it comes to accounting, intellectual property, traditional debt, licenses, cap tables, etc. But without awareness of an acquired company's technical debt, it's impossible to reasonably assess the risks and advantages of the acquisition.

Here are the risks you incur from technical debt:

1) **Legacy issues can inhibit growth potential. All too often, the concept is to rollout an MVP to gain traction and then to try to make it become what was originally envisioned. If the rollout version becomes a hit, this can actually work against your ultimate plan. The product lacks the feature set required to expand. So this snowballs into a huge problem for companies that launched a product with great success but can't get out of the vertical created by your MVP.**
2) **Not including provisions in a product so that it can be integrated into an acquirer's technology stack. It takes time and money to accommodate this**

exit strategy. Lacking both an effective exit strategy and easy integration repels potential suitors.

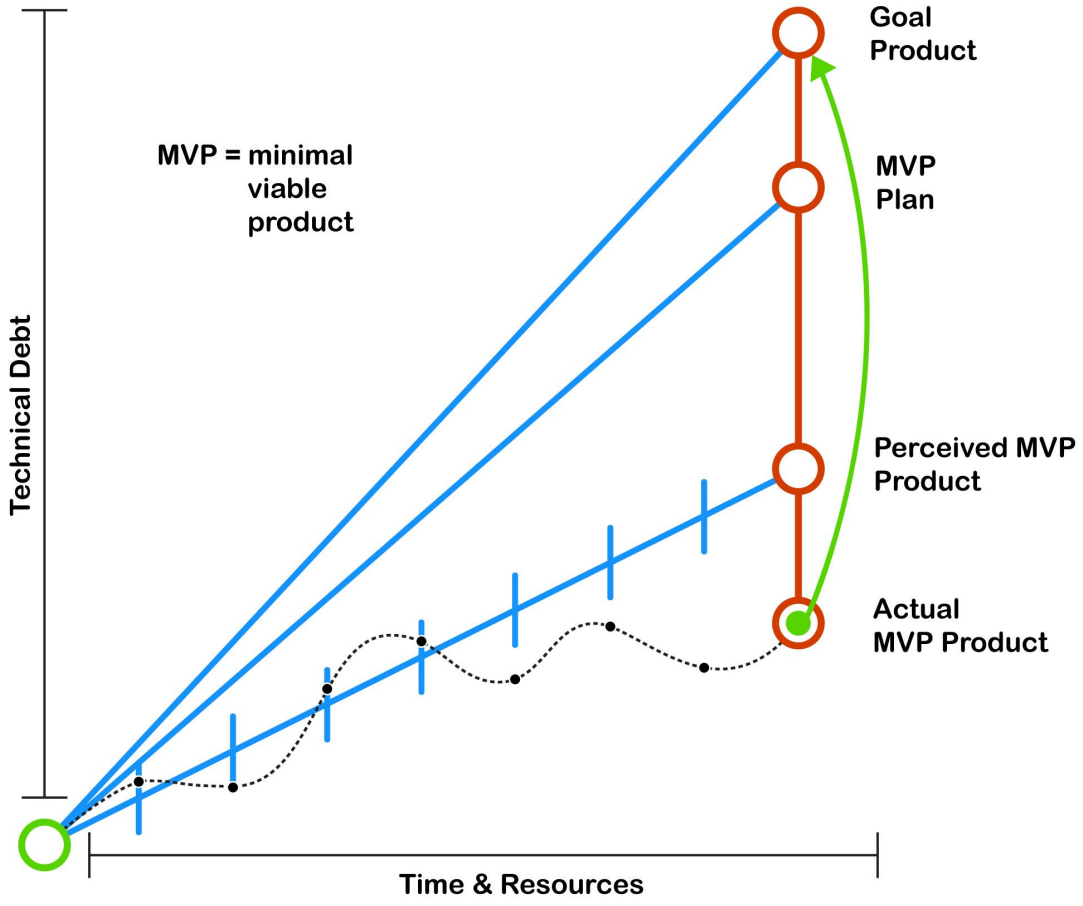
3) **Poor design or code base inhibit scalability.**

So we've determined that technical debt is like financial debt in many ways. Putting off decisions, taking shortcuts ... these make the correct action more expensive in the future. Just as in financial debt, technical debt that's managed correctly can be an advantage. Too much technical debt, however, can sink your company or dramatically increase its risk of being acquired. How do you manage this accumulating debt and keep it under control? The answer: you need to account for it, just like financial debt. Decide at an early stage what the product is supposed to be and what subset of the plan is the MVP. Take the following steps to achieve and manage your goals.

See below for the attached "Worksheet for Overcoming Technical Debt."

During the design process and early mentorship phase of a start-up, it's difficult to know what you want to be or can be. Think Big only if it is appropriate; not all great ideas are necessarily big. Knowing what the idea is, and whether it's "Big," has a dramatic effect on your early decisions. Fill a niche or disrupt an existing industry. Just know *where* the idea falls and make decisions accordingly. Investors and advisers have expectations that differ from yours. Find the right balance, and the process will be much more successful for all.

During the building process, we tend to not look at the big picture. Something to consider in this early stage: How many times am I going to do this same thing? How many times am I going to do similar things? If something only occurs once in a process, then it's difficult to leverage and make it more productive. If something only happens twice, then it's also hard to optimize. But if something needs to be done three or more times, then consider how you can do it so effectively the first time that the next time you do it the framework is set in place to repeat. This takes more effort on the first pass, but significantly less on subsequent passes. This method has an added benefit of increased quality, fewer defects, and easier troubleshooting/updating later on. As an example, let's look at batch productions versus one-off productions on a manufacturing assembly line. With the batch process, the worker can identify defects easier because every iteration should be exactly the same as the last; if something differs between them, then an error exists. Hence, the error can be identified more



accurately and much sooner. In a one-off, these errors are commonly overlooked. Another benefit to the batch process: it forces some systematic standardization and quality control.

During an acquisition, how many companies pay millions or even billions of dollars to acquire a company for its vertical success only to realize that it doesn't fit into the grid of their horizontal plans? The technical decisions a start-up makes early on can have a profound effect on its future. Look at the decisions made during the development of DOS. Those choices have had long-lasting impact on the designs and legacy concerns of Microsoft. Microsoft was very adept at dealing with these issues and made some wise initial assumptions. The outcome could have been markedly different, if they had not had great foresight to imagine the potential of moving beyond DOS.

Technical debt can be quantified and managed efficiently. You just need to know that it exists and how much it's costing you now and what will be its the impact in the future. The cost of *not* managing technical debt can be staggering and result in failed acquisitions, failed start-ups, and reduction not only in the growth path of a product, but in its market size as well.

Worksheet for Overcoming Technical Debt

Pre-Build

1) What is our product supposed to do?
2) How much will it cost to acquire paying customers?
3) Categorize customers and determine their customer lifetime value.
4) What is the MVP supposed to do?
5) Can an MVP user upgrade to the product's ultimate level without reconfiguration and loss of usability of previous product implementations?
6) How large is the target market and total addressable market for the final product?
7) How large is the target market and total addressable market for the MVP?
8) If you add-on to another product, how do the potential changes affect your product? Will it make yours obsolete, harder to enhance, etc.?
9) Develop a detailed product design.
10) Develop an MVP design.
11) Test the MVP product design.
12) Test the final product design.

During Build

1) How are we deviating from the MVP design?
    a. Compare the changes to your MVP testing and determine if the revision is still in alignment with the initial test results.
    b. If in doubt about the changes, do quick and random new user tests.
2) Document the design process and keep a log of design deviations.

a. Note why changes were made (time, technical difficulty, money, etc) and assign a budget to these changes.

3) Document the shortcuts that are being made to save time.
   a. As in step 2, note why changes were made (time, technical difficulty, money, etc) and assign a budget to these changes.
4) Document the usability changes that occur due to technical difficulties in implementing the intended design.
   a. Technology changes quickly, and revisions in the design during development are inevitable. This is one area where the process should include a stepped approach to development. Constant discussion about potential changes impedes progress; it also extends the development process and reduces moral. A stepped approach, by contrast, defines time and milestone events that allow the team to accomplish the design, and then discuss the pros and cons of revisions. But take extra care during the revision part of the process: therein lies the greatest potential for "technical creep," which increases development time and costs. Failure to manage this aspect is one of the primary contributors to technical debt. The project needs to get back on time, track, and budget, so more sacrifices are made.

Buying a Start-up

1) How is the MVP different from the company's vision?
2) What sacrifices did the company make to get to the MVP?
3) Do my existing technologies integrate well with this product or am I simply acquiring new talent and ideas?
4) What is the acquirer's ultimate vision for the product and how would they realize that vision?

About Us

Qikspace (www.qikspace.com) specializes in social collaboration software with a personal relationship management (PRM) component. Qikspace was started as a research project in 2011 the emphasis was the analysis of contextual relevance in relation to human interactions. The result of this research became the platform that is being developed and enhanced today. Our unique philosophical and technical approach has allowed us to create a solution to the complex world of online human interactions and the consequent collaborations.

About the Author

David Smith is the founder of Qikspace and has an MBA from Columbia University, is a Professional Engineer, and studied Electrical Engineering at the University of Washington.

Qikspace ™

Contact Information:

w. www.qikspace.com

e. david@qikspace.com

c. +1 732-673-2370

P.O. Box 24043

Seattle, WA 98124

