Technical Debt, Risks and Transformation

Description

In my previous article, ***Second-Level Transformation***, I focused on these issues:

- You need to undergo a Digital Transformation to achieve sustainability in the VUCA environment in which we will be operating for the foreseeable future. You must also evolve your Risk Management approach to keep in sync with the pace at which the required changes you will need to make create, amplify and morph your risks,
- The ultimate goal of transformation is business agility with a subsidiary goal of developing and honing your ability to transform and
- As you transform, you will unavoidably create debt of various types. Debt can impair your ability to transform and is, therefore, antithetical to business agility.

Risk-Based Thinking, Decision-Making and Problem-Solving are intrinsic to management (from the book: â€˜*All Management is Risk Management*â€™) and you will need to apply them to your process of planning and executing your Digital Transformation and Agile ERM adoption.

The See-Saw

There are inherent contradictions in transformation, and they can become especially meaningful in the context of the rapid responses that you are equipping yourself to execute. Essentially, you are looking to find the sweet spot between knowing nothing and doing everything, (shooting from the hip,) and knowing everything and doing nothing, (suffering from paralysis by analysis.) You will need to achieve balanceâ€”not being stuck hanging in the air and not left sitting on the ground.

It might be easier to do this if there were a single measuring stick that you could apply to make judgements about how to proceed, but there are many competing priorities. You need to balance near-term and longer-term considerations while also making value judgements about numerous factors that can determine your businessâ€™ success. For instance, you might have to choose between racing to market with an incomplete MVP offering and waiting a bit to mature it with additional features. You might have to decide whether itâ€™s more important to accelerate delivery of new features for an existing product or to refactor and rebuild operational components that will ultimately support later

versions of it and other products to come.

Up-to-date software development tools and approaches enable rapid implementation. No-code, low-code tools can enable â€˜citizen developersâ€™ to build their own applications, also at speed. However, ungoverned implementation by individual groups distributed across the company has the potential to create *technical debt.* Some Technical Debt is an unavoidable side effect of developing anything, but uncontrolled implementation by disconnected and ungoverned groups exacerbates the volume, if not the severity of the impacts it creates.

This is hardly a new problem. Since computers were first applied to business operations, departments have competed for services from their IT organizations, and balancing between deliberate planning and design and speedy delivery has driven the evolution of innumerable SDLCs and management paradigms. Agile development approaches are a good example of something that evolved out of this. Early on, the development costs and expected operating life of systems were such that the time and effort spent on planning and design paid back in lower operational and evolution costs over their lives. Now, applicationsâ€™ lives are shorter, and they are easier to modify, which favors iterative and rapid implementation with less time spent at the front end on requirements engineering and design. *Big Design Up Front* (BDUF) is viewed by many as an anachronistic form of paralysis by analysis.

However . . .

Architecture (the big brother of design) and governance still matter. If your enterprise-level systems run on an easy-to-manage, elastic cloud environment it will be easier and faster to develop and deploy new applications, products and services. If your systems are built using standards-based architectural components, such as microservices, containers and API integration, you can reuse them more easily to accelerate deployment of solutions you need instead of building things that you donâ€™t have to. If you employ Knowledge Management disciplines to ensure consistent tagging, you will have an easier time identifying and locating reusable assets and avoid creating redundant versions of them. Not building unnecessary things is the easiest way to avoid creating technical debt, to be sure, provided that you are not bastardizing existing components in order to support requirements for which they may not be optimized or appropriate.

Governance is how you can maintain control of your architecture to preserve the value of the things that you spend effort, time and money implementing. However, it adds overhead and can slow development efforts so this is the trade-off you must manage proactively. In the book I identify ***Capabilities*** and ***Enablers*** that you will require in your battle against technical debt and other structural ills. In addition to enabling a more deliberate approach to managing your business, these will constitute the foundation of your Agile Enterprise Risk Management discipline.

These include:

- **Change Management Team**â€”This group oversees a touchpoint for changes in your enterprise through which you will monitor for risks and opportunities for improvement and prevent accumulation of unnecessary debt. Your Project Management Office (PMO) should be an on-the-ground element of this.
- **Architecture Subject Matter Experts (SMEs)**â€”These folks should probably be housed within your CTOâ€™s organization. They contribute to debt avoidance in numerous areasâ€”research,

standards setting, solution architecture design and advisory consulting. Steering users toward reusable solutions or helping to define new ones that are developed to be reusable is what you should expect from them.

- **KM Team and Taxonomy**â€”consistent identification of artifacts facilitates their being found and reused when they are needed.
- **EA/BA models**â€”These models serve to document companyâ€™s current anatomy and, through extension, the design of its future anatomy. Changes to your company that induce transactions in your model repositories should trigger prescribed governance or management activities, such as appraising risks and formulating treatments.
- **Pattern Library**â€”this is an archive of potentially useful and reusable knowledge and artifacts in diverse formats. A common implementation of such an archive might be a Data Lake, indexed with terms from your taxonomy. A data mesh is an up and coming alternative architecture for this purpose. (see several articles on Martin Fowlerâ€™s site for more detailed information)

So, even though the evolution of technology and tools available for you to manage your company have changed the cost/benefit balance between debt avoidance and speed to market, the value of planning and design hasnâ€™t gone away. In the longer term, persisting in taking shortcuts to achieve maximum change velocity will shortchange your ability to transform, overall.

In the next article in this series, I will examine transformation in a little more detail and explore why short-termism can undermine your ability to do it.

Date Created
2021/11/16
Author
howardmwiener