

Tool Versus Product

Background

To clearly understand the task of productizing an in house tool, a background understanding of the nature / environment of such development efforts will provide a base level foundation. Internal development efforts are often (or inevitably) performed in "live" production environments. Such efforts are often initiated or have a tendency to address the "current problem" facing the business. In today's world, it is not uncommon for the business unit to have expectations of instant gratification, therefore development effort often proceeds by limiting the number of variables, or proceeding with a "fixed environment" mentality. To expand on this thought, the actual development effort can take advantage "hard coding" many aspects to the current infrastructure, which exponentially shortens the delivery timeframe of specific tool functionality. In general, the overall requirements are greatly over simplified under the premise of "putting a dent in the problem, with the intention of coming back at a later point to fill out functionality." As the saying goes, "the problem with quick & dirty, is that dirty remains long after the quick has come and gone."

Crossing The "Productization" Chasm

Although this phrase is actually the title of book by Geoffrey A. Moore, the "chasm" being referred to is chasm that must be crossed to successfully transition from the concepts / problems addressed by "in house" tools to that of commercial product. Human nature tends to minimize this chasm by rationalizing under the guise of: "the working solution / prototype exists" and therefore we are relatively close to a solution. However, the following will provide a great deal of insight on this exercise from direct experience on the issues, as our original business plan embarked upon such a strategy. Here are some of the high level categories (already navigated by DataFrameworks) that make up this chasm:

What DataFrameworks has already developed is a solution that

1. Coexists with existing infrastructure

The coexistence includes heterogeneous storage along with in house tools which are traditionally hardcoded, applications, workflow, and operational processes. One might be tempted to eliminate this requirement by making the usage assumption that the functionality will be deployed on a "go forward" basis, we initially made this assumption which we soon discovered to be invalid. Hard coded infrastructure components often restrict or introduce difficulties for a company to adopt new application, workflows, and storage without additional software development.

2. Coexists with existing data

Here again we made the initial "go forward" usage assumption, which subsequently required the development of additional functionality (such as importing & scanning existing data). The only upside to this requirement is the realization of minimal services revenue.

3. Incorporates Valid Usage Assumptions

Throughout the productization process, one soon discovers the hidden costs associated with invalid usage assumptions. As outlined in the background section, uncovering the usage assumptions made during internal development (of the in house tool) and then re engineering the solution to accommodate the requirements of a commercial product involve substantial business risk.

4. Eliminate / Reduce Onsite Developer Support

By their very nature in house tools come with onsite developer support, a luxury not afforded commercial products. Here are some examples:

- Internal customers no longer have technology choices. IT begins to enjoy advantages afforded traditional monopolies.
- Internal customers receive support and training directly from development resources. Supportability concerns inevitably arise due to employee turn over within development staff (or key developer).
- When the tool throws error(s), the developers have intimate understanding of both the infrastructure and tool, and therefore can resolve the error condition(s) in a manner "friendly" to both the tool and the infrastructure. Commercial products must guide users to appropriate (or desired) resolutions.

Without the advantage of onsite developer support, inevitably additional functionality, training, and services are required of a commercial product

Summary

Based on our experience, throughout the process, requirements will be discovered that will necessitate the development of additional functionality to address the high level categories previously outlined (coexistence with existing infrastructure, coexistence with existing data, invalid usage assumptions, and eliminate need for onsite developer support).