From Project to Product

Matthew Philip[1](✉) and Yoan Thirion[2]

[1] St. Louis, USA
[2] Luxembourg, Luxembourg

Abstract. As technical advances have enabled organizations to deliver software to the market faster, in turn shortening the feedback loop for new ideas and spurring innovation, legacy organizations need to update their mindset from a project-driven to a product-driven approach or risk being displaced by product-native organizations. This poster shows the high-level principles that represent our experience guiding organizations with a project-to-product approach.

Keywords: Product · Project · Outcomes · Experimentation · Value · Flow · Vision

1 History and Background

Organizations that are product-native – that is, product-oriented from their beginning – typically do not need to take a project-to-product journey, inasmuch as they have never known a time when IT was considered anything but integral to the organization's success. On the other hand, organizations who have inherited a legacy approach to information technology such that they have traditionally regarded IT as a cost center and separate from the business often experience the existential threat of being disrupted (or worse) and therefore choose to orient or reorient themselves toward products and product development, resulting in a more or less intentional rethinking of their ways of working.

For example, one of the organizations in our experience had a highly matrixed, globally distributed IT group of more than 12,000 people that optimized for project-management concerns rather than its customers and the flow of value to them. This organization had a history of so-called "Agile" and "DevOps" transformations, but none of them clearly dealt with moving from project to product. These transformations yielded some gains but ultimately, because they were internally oriented, did not focus on true customer and user outcomes.

2 Defining "Product"

A digital product (and therefore product management) is fundamentally different from a project (and project management). According to Jez Humble, some assumptions of *projects* can include:

- Once we've built it, it doesn't change much

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 207–212, 2021.
https://doi.org/10.1007/978-3-030-88583-0_21

- In the course of building it, we don't learn much significant information
- You must complete it because you can start using it.

In contrast, digital products [3]:

- Will change a lot over their lifecycle
- Allow us, in the course of building them, to discover large amounts of information that can inform decisions and directions
- Can be used and provide value before they are "complete".

A working definition of product is (adapted from the Scrum Guide):

> A product is a vehicle to deliver value. It has a clear purpose, functional boundary, stakeholders, users and customers. A product could be a single or a group of services, physical products, applications, platforms, systems, and data.

3 Principles for Moving Toward Product-Orientation

As technical advances have enabled organizations to deliver software to the market faster, in turn shortening the feedback loop for new ideas and spurring innovation, legacy organizations need to update their mindset from a project-driven to a product-driven approach or risk being displaced by product-native organizations.

This change in mindset is significant, covering a wide range of concerns from psychological safety to budgeting to success measures. This poster shows the high-level principles that represent our experience guiding organizations with a project-to-product approach.

In moving toward product-orientation, we prefer the following principles:

3.1 Outcomes over Outputs

Outputs can be defined as delivering features, user stories or other work item types without respect to whether they are the right things to build or make a difference in user success or perceived value. A project mindset prioritizes outputs with metrics such as conformance to plan and feature delivery. A feature delivered that does not produce a desired outcome may still be considered a success in a project-oriented approach. However, by preferring *outcomes* – that is, measurable impact toward user and/or customer goals – we will acknowledge that not every feature we plan will necessarily achieve what we hoped it would but instead optimize for user success. It is the difference between simply doing work and doing the right work.

As Barry O'Reilly writes [4]:

> An outcome-based approach… demands that you clarify the success you seek—in quantifiable terms—by crisply defining what you're trying to achieve so people know why it matters… Clarity of destination allows people to explore different options to discover if the investments you make are moving you in the direction you desire… An outcome-based approach allows you to be accurate, more adaptive, and take action to course-correct and own the results you gather relative to the final destination.

3.2 Solving Problems over Building Solutions

Solving problems orients the team toward the user/customer. In particular, a product-orientation enlists engineers and cross-functional teams in generating the ideas for features rather than merely carrying out the solutions of a single product owner. John Dewey wrote that "a problem well put is half solved" [5]. Product-minded teams should build competency in and spend time identifying the problems they're trying to solve, as this will improve their chances of building the right things.

3.3 Options over Requirements (and Optionality over Linearity)

By framing potential work as options rather than requirements (which carry the connotation that they must be done), product teams emphasize their ability to discard unhelpful work in the face of new information. Teams make plans when they have less information than they will discover, so even the language of "requirement" can inhibit the ability to change course.

As Donald Reinertsen writes [6]:

> Fast feedback loops give us the ability to truncate unproductive paths quickly, which unlocks resources for other purposes … In contrast, the traditional approach to development focuses on up-front planning rather than adaptation. We tried to forecast everything and failed to do this accurately.

3.4 Experiments over Backlogs (and Hypotheses over Features)

Ultimately, product or service development is a process to test an hypothesis about system behaviour in the environment or market it is developed for [12]. Project-minded teams start planning with feature ideas, whereas product-minded teams plan with hypotheses. The language of experimentation expresses the uncertainty inherent in software development and changes the definition of success from features that are implemented but not validated (and therefore uncertain progress) to validated learning and true progress.

3.5 Customer-Validated Learning over PO Assumptions

This is why it is important to take a user-centered approach. We found that the vast majority of project-oriented teams did not use experience-design and design-thinking techniques such as personas; product-oriented teams did.

According to Daniel Vacanti, "True business value can be determined only after delivery to the customer. Choices about what to work on and when, then, are really just you placing bets on what you think the customer will find valuable" [11]. This statement emphasizes the limitation on a single product owner's knowledge and ability to forecast the future.

3.6 Measuring Value over Measuring Cost

Project-oriented organizations often prioritize measurements such as scope, cost and time. However, we have found, like Humble, that these concerns are fundamentally unsuited for product management inasmuch as the success of a product isn't dependent on these factors. Or, as Humble says, "How much it costs doesn't matter if people don't send you money" [3]. Additionally, Douglas Hubbard found that the concern is unwarranted and can actually mislead: "Even in projects with very uncertain development costs, we haven't found that those costs have a significant information value for the investment decision… The single most important unknown is whether the project will be canceled. …The next most important variable is utilization of the system, including how quickly the system rolls out and whether some people will use it at all" [1]. An obsession with cost undermines a focus on outcomes and experimentation.

3.7 Flow over Utilization

Reinertsen found that "Capacity utilization increases queues exponentially" and that "operating at high levels of capacity utilization increases variability". As a result, he recommends that product teams control queue size and not capacity utilization [7].

3.8 Product Vision, Strategy, Personas and Principles over Product Roadmaps

Product vision, strategy, personas and principles (aka product manifesto) enable the experimentation and problem solving referred to earlier, whereas roadmaps connote a fixed scope-and-time-based approach. In a product-oriented team, we use the former – which typically don't change as much – to guide development.

3.9 Small-Batch Delivery over Big-Batch Delivery

Reducing batch size has many quantifiable benefits that support product thinking, including reduced cycle time, faster feedback, increased employee motivation and reduced variability in flow [7].

3.10 Optimizing for Assumptions Being Wrong over Optimizing for Assumptions Being Right

Product-oriented teams embrace the reality that most of their ideas will not work. One of the reasons we de-emphasize traditional feature-based roadmaps is because "at least half of our ideas are just not going to work" [2]. Kohavi found that "evaluating well-designed and executed experiments that were designed to improve a key metric, only about one-third were successful at improving the key metric!" [9]. Additionally, "Netflix considers 90% of what they try to be wrong" [8]. This reality requires a strategy to deliver and validate as quickly as possible.

3.11 Teams of Missionaries over Teams of Mercenaries

Product-team culture is perhaps best described by John Doerr's "missionaries and mercenaries" metaphor. "Mercenaries are driven by paranoia; missionaries are driven by passion… Mercenaries focus on their competitors and financial statements; missionaries focus on their customers and value statements … missionaries are obsessed with making a contribution … and… are fundamentally driven by the desire to make meaning" [10].

3.12 Business-Driven over IT- or PMO-Driven

The movement from project to product requires not only business involvement and collaboration, but business orientation, subordinating IT and the Project-Management Office, which are often driven by project-management concerns that undercut product success discussed earlier.

References

1. Douglas Hubbard. https://www.cio.com/article/2438748/the-it-measurement-inversion.html
2. Cagan, M.: Inspired: How to Create Tech Products Customers Love. Wiley, Hoboken (2017)
3. Jez Humble. https://lectures.leanagile.pm/
4. Barry O'Reilly. https://barryoreilly.com/explore/blog/your-mission-is-to-produce-outcomes-not-outputs/
5. Dewey, J.: Logic, the Structure of Inquiry. Henry Holt & Co, New York (1938)
6. Reinertsen, D.: Towards Developing Accelerators in Half the Time. https://accelconf.web.cern.ch/ipac2011/papers/weib02.pdf
7. Reinertsen. http://lpd2.com/sample-page/the-principles-of-flow/
8. Moran, M.: Do it wrong quickly: how the web changes the old marketing rules (2007)
9. Kohavi, R., et al. http://ai.stanford.edu/~ronnyk/2013%20controlledExperimentsAtScale.pdf
10. Doerr, J.: Two Kinds of Internet Entrepreneurs, UPenn. https://knowledge.wharton.upenn.edu/article/mercenaries-vs-missionaries-john-doerr-sees-two-kinds-of-internet-entrepreneurs/
11. Vacanti, D.: Actionable Agile Metrics For Predictability: An Introduction Kindle Edition (2105)
12. Barry O'Reilly. https://barryoreilly.com/explore/blog/how-to-implement-hypothesis-driven-development/