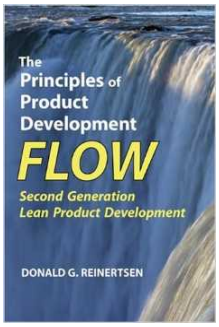
BPTrends **BOOK REVIEW**

The Principles of Product Development Flow
Second Generation Lean Product Development

By Donald G Reinertsen

Celeritas Publishing, 2009, 304 pages, $29.14 (Kindle $27.68)

Reviewed by Tom Bellinson



Let me preface this review by stating for the record that I read a lot of non-fiction books. Many of the books are about management topics, which should come as no surprise to those who know me. By the time a book gets on my radar, it probably has some new information to offer. Unfortunately, this is often limited to what I refer to as a few "golden nuggets." Even good books have lots of fluff in between the nuggets, usually in the form of extended stories to illustrate the point.

I mention this because Reinertsen is an author who has stood the typical model on its head. If anything, he left me wanting more stories that would further illustrate the mountain of golden nuggets he has provided. As the name implies, the book is broken down into a series of principles that have been categorized into eight major groups with several sub-classifications for each. Although the book is under 300 pages long, it felt like many 500 page books I have read. Let's just say "it's dense!"

The Categories

In addition to evaluating the work, a good nonfiction book review also provides a taste of some of the information the reader will gain. Given what I have already said, you can imagine that my attempt will be wholly inadequate, but at least I can try to pique your curiosity. Let's take a look at each of the eight categories in brief.

The Economic View

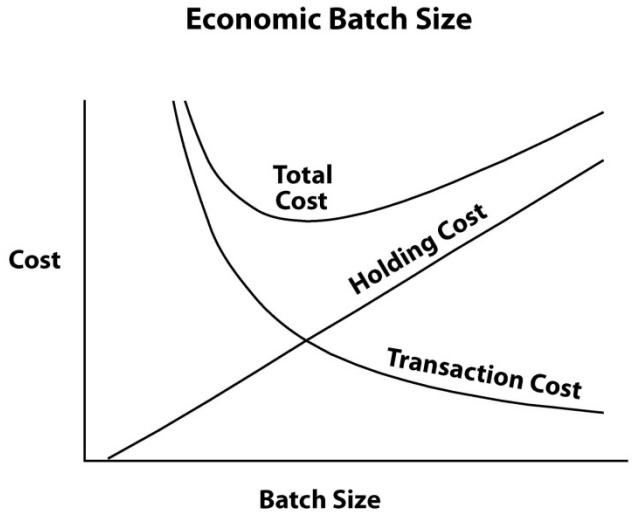
For me, this chapter packed the largest wallop. I have become a skeptic of concepts and practices that are not measurable and Reinertsen dives right in with this chapter about ascribing economic value to the work we do as product developers. He points out that we often use surrogate (or proxy) measurements that offer little or no economic connection with the marketplace. Even in an organization in which product developers are building tools for internal consumption, there are clear economic drivers that can guide us in good decisionmaking.

Two things jumped off the pages of this chapter:

1. If you only measure one thing, measure the cost of delay
2. Measure the work, never the worker

The Cost of Delay is a premonition of the next category of principles. Almost all economic factors can be traced back to managing delay. One of the great additions to the lexicon of product development comes from the notion of quantifying factors that those of us in product development often treat as "soft" measurements. This is not to say that we should spend extensive effort on computing economic value to the last dime. In fact, the author argues that just getting close delivers almost all of the value of measurement with a fraction of the effort.

The reason for this can be represented in the graph below:



From "The Principles of Product Development Flow," by Donald G. Reinertsen.
Celeritas Publishing: 2009. Copyright 2009, Donald G. Reinertsen

Here we see two opposing metrics: transaction cost and holding cost. Throughout the book, this is a recurring theme. Two opposing forces create a U-Curve (depicted here as total

cost), which gives us a range of good choices near the trough. Approximating this point nets you optimal economic advantage without extreme accuracy. More later about the specific factors listed here.

Managing Queues

Anyone familiar with Lean thinking understands the importance of queues. Unfortunately, many people assume that the principles that apply to the repetitive cadence of manufacturing have no place in the less predictable world of product development. This set of principles dispels this misconception. Reinertsen argues that queues are the *most* important factor in maintaining optimal product development flow.

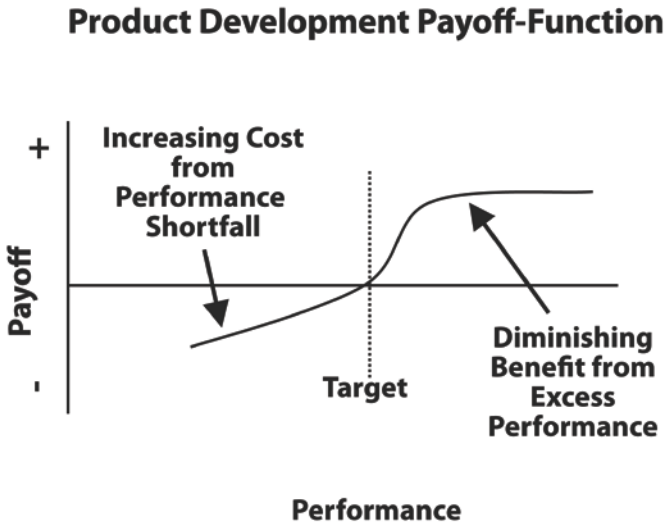
Product development queues are more insidious because they tend to be invisible. In a manufacturing environment, work in process (WIP) inventory build up is easy to see. But, more importantly, it is easy to quantify. We often think of product development backlogs as free. They are not.

Queues affect capacity utilization. As queue size increases, we tend to apply more of our capacity to alleviate the situation. This increased capacity utilization reduces our flexibility, which is mandatory in the unpredictable world of product development. Reinertsen uses a series of mathematical formulas to illustrate how high levels of capacity utilization on an ongoing basis increases queue size and derails product development efforts. He prescribes controlling capacity utilization as the best way to manage queue size.

Exploiting Variability

In the manufacturing world, variability is almost always bad. Six Sigma and Lean thinking encourage us to stamp out variability. In product development, variability is not always bad. Because Agile development draws from these disciplines, we must be very careful not to take too much with us.

What becomes clear from the pages of this book is that we must remain vigilant for unexpected variations from the plan that benefit the product.



From "The Principles of Product Development Flow," by Donald G. Reinertsen. Celeritas Publishing: 2009. Copyright 2009, Donald G. Reinertsen

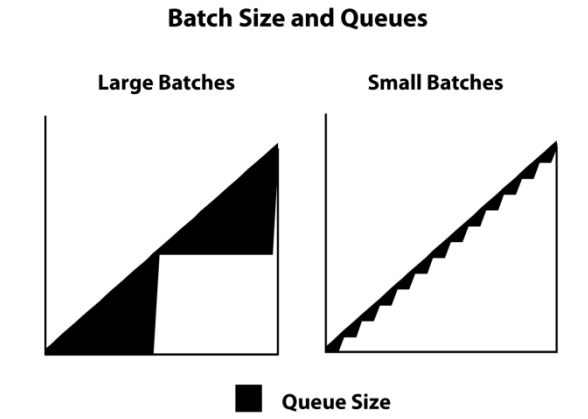
The payoff/performance chart above illustrates a typical product development feature payoff curve. Clearly, there comes a point at which additional features cost more than the benefit that is derived from them. If we can apply economic principles to the value of features, we can quantify the benefit of continued development and properly assess whether additional features make economic sense.

If variability has both positive and negative consequences, it stands to reason that we should only try to eliminate bad variability. Reinertsen discusses ways in which to do this. He also introduces this idea: when it is imprudent to eliminate variability, the better choice is to minimize its cost. Lowering capacity utilization (and thereby queues) and reducing batch size (iteration cycle time) have such an effect.

Reducing Batch Size

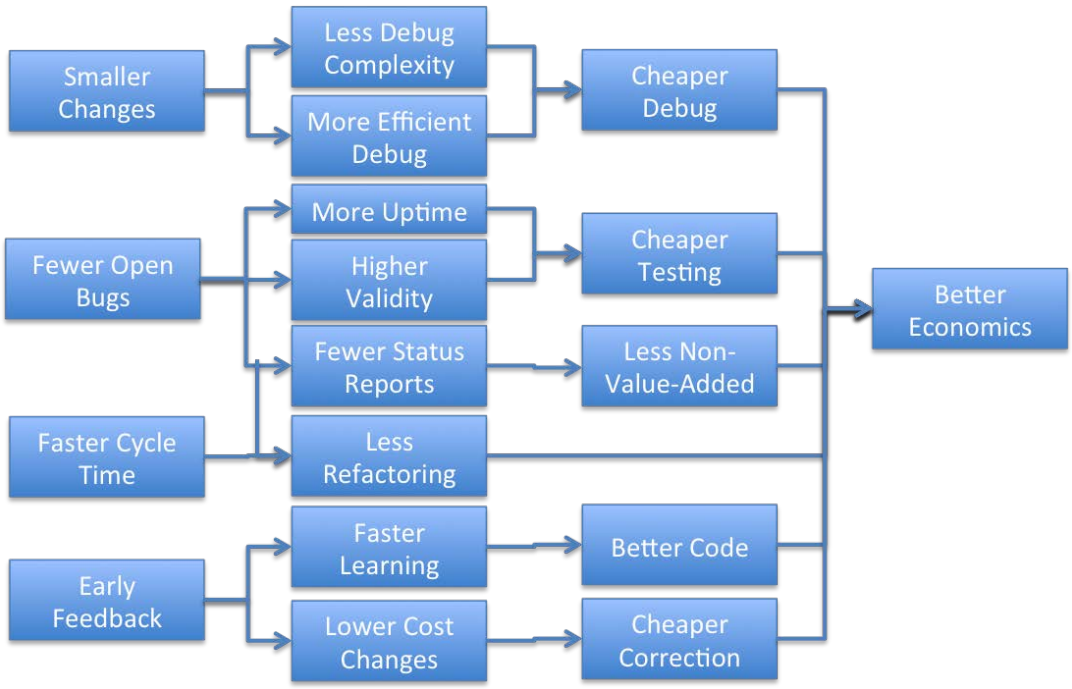
No self-respecting Lean approach to product development would be complete without discussing batch size. In product development, batches contain one or more functional elements. Batches can be more elusive in an uncertain environment. Often functionality that seems like it will require minimal design and development effort ends up being anything but. The opposite can also be true. There are no easy answers to this conundrum, but adaptability to a changing situation is paramount.

Anyone working in an Agile software development environment is familiar with the benefits of small batch size. Small batches (or short iterations) provide fast feedback (more on this later), but they also have the effect of reducing queue size. The simple diagram below illustrates this perfectly.



From "The Principles of Product Development Flow," by Donald G. Reinertsen. Celeritas Publishing: 2009. Copyright 2009, Donald G. Reinertsen

Clearly, big iterations require big queues. Reducing batches can have many benefits in a software development environment. Here is an example from the book for the testing portion of software development.



Lean principles: colocation, short iterations, low hanging fruit, and modular design are all discussed. While none of these ideas are new, it is valuable to read about them in the context of Much of this chapter is a rehash of concepts that are familiar to anyone who has used Agile or maximizing economic value.

Applying WIP Constraints

Continuing the theme of ascribing economic value to costs of delay, the chapter on WIP discusses how said cost can be minimized by controlling WIP. A number of tactics for controlling WIP are put forth. Goldratt's Theory of Constraints is invoked along with rate-matching between adjacent processes. The following chart shows compelling evidence of the efficacy of WIP constraints.

Effect of WIP Constraint

	No WIP Constraint	WIP Constrained at 2x Avg	Extra Cost / (Savings)
Annual Cost of Capacity	$2,000,000	$2,022,472	$22,472
Opportunity Cost of Blocked Jobs	$0	$25,000	$25,000
Cost of Delay	$1,643,836	$1,183,562	($460,274)

Note: Cost-of-delay savings is 9.7 times the added blocking and capacity costs.

The usual suspects are rounded up for slashing WIP: cutting low-value features, flexible resource allocation and blocking demand. Reinertsen also provides a number of ways to visualize and monitor WIP to know when to deploy WIP control measures.

Controlling Flow under Uncertainty

One of the hallmarks of this book is the use of some unexpected sources for models of behavior. In this chapter, Reinertsen uses traffic control systems because they must adapt to constantly changing and uncertain conditions -- accidents happen. Weather happens. Street repair happens. Some of these things happen on a schedule that can be anticipated and others cannot. Sound familiar?

What we learn from this example is that flow is a product of speed and density. Furthermore, small changes in either can have a profound effect on flow. When either goes up significantly, congestion quickly ensues. As we already know, congestion for product developers means bigger queues, higher capacity utilization, delays and higher costs. How to manage this?

One of the best examples of what Reinertsen calls "controlled occupancy" is familiar to Californians. It is what they refer to as the "ramp meter."

Ramp meters control the volume of vehicles on the freeway and thereby increase flow without actually limiting access. We can use similar mechanisms to manage WIP level. Ideas such as cadence, forecasting and synchronization are discussed. As is the case throughout the book, concepts introduced elsewhere are interwoven. Capacity utilization is revisited with an eye towards determining the correct margin to leave available to ensure higher flow rates.

There is also a very interesting section of this chapter that discusses how to sequence work. All things being equal, it is best to do the smallest jobs first in order to reduce the cost of delay. However, different jobs will often have vastly different costs-of-delay. In this case, it is usually better to do high cost-of-delay jobs first. There is a lot of great material here and the usual mathematical treatment of the topic, which will help the practitioner with better tools to evaluate how work can be optimally sequenced.

The chapter concludes with a discussion of resource management. I would like to provide some additional details here, but I will refrain in order to keep this review to a manageable length. Again, you will find some powerful tools to think about how to optimize the use of human and system resources.

Using Fast Feedback

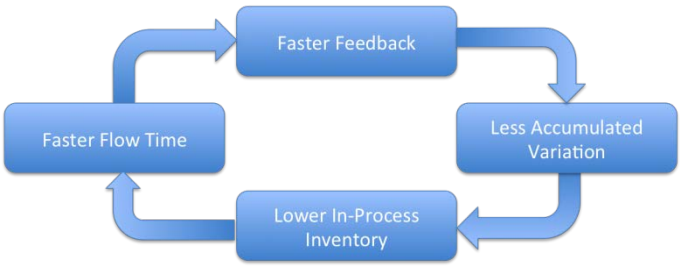
Creating short feedback loops is nothing new to the Agile practitioner. It is a central principle and the entire philosophy tends to revolve around arranging work to get fast feedback. Here again, we see the use of a more quantitative approach to evaluating this feedback and defining the right metrics to target economic indicators of performance.

Balanced Set Points

		Sensitivity Factor	Economic Limit	Control Set Point	
Project A	1% Expense	$50,000	$50.000	1.00	Percent
	1% Unit Cost	$250,000	$50,000	0.20	Percent
	1% Product Performance	$200,000	$50,000	0.25	Percent
	1-Month Delay	$500,000	$50,000	0.10	Months
Project B	1% Expense	$100,000	$50.000	0.50	Percent
	1% Unit Cost	$50,000	$50,000	1.00	Percent
	1% Product Performance	$200,000	$50,000	0.25	Percent
	1-Month Delay	$50,000	$50,000	1.00	Months

The chart above shows how different control variables have different economic impacts and should therefore have different tolerances for variation (i.e. Set Points). Here we see two different projects in which the same variables have different economic impacts. So, the set points have been adjusted to match the sensitivity of each variable. This has the effect of equalizing economic deviations.

Reinertsen cautions against mistaking dynamic goals for static goals. This can be particularly true with software development projects because we often establish goals for the project in the early stages and become locked into our believe that deviations from those goals are categorically bad. Design goals are almost always dynamic. As more is learned, we must be prepared to throw out our most fundamental beliefs about why we are doing the project and what we need to accomplish. It is only through this type of thinking that we can exploit positive deviations. Fast feedback allows us to remain vigilant for these opportunities.



Fast feedback provides a reinforcing cycle that keeps our inventory of design variations low. Because we are constantly correcting our course, refactoring prior efforts that turn out to be bad deviations are kept small enough to be much less costly than the deviations that come from longer feedback cycles.

There is a lot here about not only selecting the right metrics, but how to establish the right control mechanisms so that the metrics have their intended consequences. One example offered is that of the bathroom scale purchased to measure weight-loss performance. Many who have had this experience will report the ultimate avoidance of the dreaded device. Only with the proper controls in place does the scale become an effective tool within the system.

Achieving Decentralized Control

In my own experience, companies that implement decentralized control often exhibit a mild form of schizophrenia. Leadership feels obligated to assert their authority, while at the same time sending messages that people at all levels should feel free to make "appropriate" decisions. Without clearly defining what "appropriate" means, staff are left questioning what decisions are available to them and thus act as though they have no decision-making power unless specifically granted on a case-by-case basis.

In this chapter, Reinertsen uses his military experience and training with the Marines to illustrate how an unlikely organization provides a shining example of the principles of decentralized control. He points out that in the military, field personnel are given a mission, but since conditions on the ground can change rapidly, advantage goes to the fighting force that can adapt the quickest. Relying on extensive central command slows things down, puts decisions in the hands of those with the least knowledge of the situation on the ground, and risks miscommunication when orders are returned.

People who work together develop tight communications based on common context and language. When leadership defines the mission clearly, teams can interpret their situation against that mission to make good choices. They can also assess when it is appropriate to engage leadership.

My favorite part of this chapter was the discussion around alignment. The discussion of creating clear roles and boundaries resonates because too often neither of these is well-defined and clearly communicated in a manner that is actionable by team members at all levels. Reinertsen, in typical fashion, offers lots of ways to evaluate the characteristics of managing decentralized control by integrating ideas from all the preceding chapters.

Conclusion

Throughout this book, ideas are integrated from different chapters. Because each principle is labeled with the type and a sequence number, the book is easy to use as a reference. When one principle calls for integration with another, we get a clear picture of how all these principles fit together to form a coherent strategy for building effective product development organizations.

Despite the fact that much of the content has been covered by other texts on Lean and Agile development, Reinertsen brings a rigor to these practices that is often missing from other offerings. By focusing on real economic value, we learn to manage the elements of a project that matter most. I came looking for a few nuggets and found an entire gold mine.

Tom Bellinson

Mr.Bellinson has been working in information technology positions for 30 years. His diverse background has allowed him to gain intimate working knowledge in technical, marketing, sales and executive roles.
Most recently, Mr. Bellinson finds himself serving as a Project Manager for Agile software development at the University of Michigan Medical School. From 2008 to 2011 Bellinson worked with at risk businesses in Michigan through a State funded program which was administered by the University of Michigan.
Prior to working for the University of Michigan, Mr. Bellinson served as Vice President of an ERP software company, an independent business and IT consultant, as chief information officer of an automotive engineering services company and as founder and President of a systems integration firm that was a pioneer in Internet services marketplace.

Bellinson holds a degree in Communications with a Minor in Management from Oakland University in Rochester, MI and has a variety of technical certifications including APICS CPIM and CSCP.

He can be reached at tom@itmethods.net.