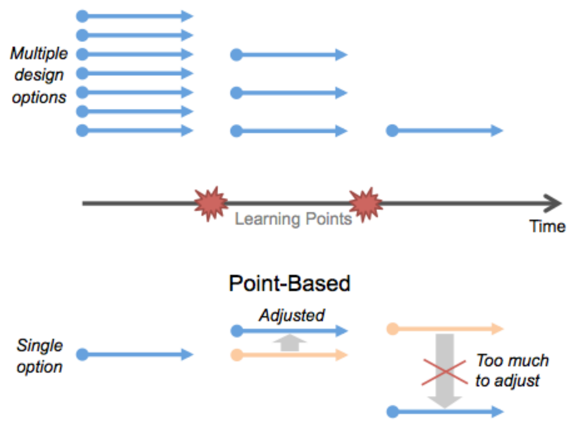
# Principle #3 – Assume variability; preserve options

*Generate alternative system-level designs and subsystem concepts. Rather than try to pick an early winner, aggressively eliminate alternatives. The designs that survive are your most robust alternatives.*
*—Allen C. Ward, Lean Product and Process Development*

Systems builders tend to have a natural inclination to try to reduce variability. It just seems that the more you think you know and have already decided, the further along you are. But this is often not the case.

While it is true that variability can lead to bad outcomes, the opposite case can also be true. Variability is not inherently bad or good. Rather, it is the economics associated with the timing and type of variability that determines the outcomes. A focus on eliminating variability too soon perpetuates a risk avoidance culture wherein people can't make mistakes and gain experience by learning what works and what doesn't.

Other than a general understanding of system intent, Lean systems builders recognize that very little is actually known at the beginning of the project. If it was, they would have already built it. However, traditional design practices tend to drive developers to quickly converge on a single option—a point in the potential solution space—and then modify that design until it eventually meets the system intent. This can be an effective approach, unless of course one picks the wrong starting point; then subsequent iterations to refine that solution can be very time consuming and lead to a suboptimal design [1]. And the bigger and more technically innovative the system is, the higher the odds are that your starting point was not the optimal one. A better approach, referred to as Set-based Design or Set-based Concurrent Engineering [2], is illustrated in the figure below.



In this approach, the systems builder initially casts a wide net by considering multiple design choices at the start. Thereafter, they continuously evaluate economic and technical tradeoffs—typically as exhibited by objective evidence presented at integration learning points. They then eliminate the weaker options over time; and finally, converge on a final design, based on the knowledge that has been gained to that point.

This process leaves design options open as long as possible, converges as and when necessary, and produces more optimal technical and economic outcomes.

## Learn More

[1] Iansiti, Marco. “Shooting the Rapids: Managing Product Development in Turbulent Environments.” California Management Review 38 (1995): 37–58.
[2] Ward, Allan C. and Durward Sobek. Lean Product and Process Development. Lean Enterprise Institute Inc., 2014.

# Principle #4 – Build incrementally with fast, integrated learning cycles

*The epiphany of integration points is that they control product development and are the leverage points to improve the system. When timing of integration points slip, the project is in trouble.*
*—Dantar P. Oosterwal*

## Building Systems Incrementally

In traditional, stage-gated development, investment cost begins immediately and accumulates until a solution is delivered. Often, there is little to no actual value delivered until all of the committed features are available, or the program runs out of time or money. During development, it is difficult to get any meaningful feedback because the process isn't designed for it, and the system isn't designed or implemented in such a way that incremental capabilities can be evaluated by the customer. The risk remains in the program until the deadline, and even into deployment and initial use. This process is error prone and problematic, and typically results in loss of trust between the system builder and the customer. In an attempt to adjust for this, customers and systems builders try even harder to define the requirements and select "the best" design up-front. They also typically implement even more rigorous stage gates. Each of these solutions actually compounds the underlying problem. This is a systems-level problem in the development process, and it must be addressed systemically.

## Integration Points Create Knowledge from Uncertainty

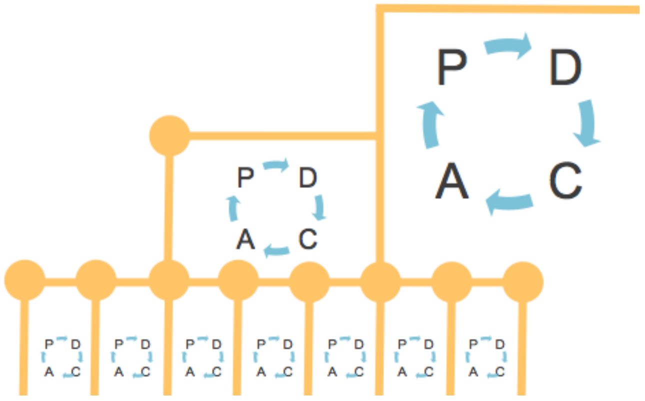
Lean systems builders approach the problem differently. Instead of picking a single requirements and design choice early—assuming that it is both feasible and will provide fitness for purpose—systems builders work within a range of requirements and design options (Principle 3) and build the solution incrementally in a series of short time-boxes. Each time-box results in an increment of a working system that can be evaluated by the system builder and the customer. Subsequent time-boxes build upon the previous increments and the solution evolves until it is released. The knowledge gained from integration points is not solely for the purpose of establishing technical viability. Many integration points can serve as minimum viable solutions or prototypes for testing the market, establishing usability, and gaining objective customer feedback. Where necessary, these fast feedback points allow the systems builder to "pivot" to an alternate course of action, one that should better serve the needs of the intended customers.

## Integration Points Occur by Intent

Cadence-based *integration points* become the primary focus of the systems builder, via a development process and a solution architecture that is designed in part for that specific purpose. Each integration point creates a “pull event” that *pulls* the various solution elements into an integrated whole, even though it addresses only a portion of the system intent. Integration points *pull* the stakeholders together as well, a routine synchronization that help assure that the evolving solution addresses the real and current business needs, as opposed to the assumptions that were established at the beginning. Each integration point delivers its own value by *converting uncertainty into knowledge*—knowledge of the technical viability of the current design choice, and knowledge of the potential viability of the solution, all based on objective measures (Principle #5).

## Faster Learning Through Faster Cycles

Integration points are an instantiation of Shewart’s basic Plan-Do-Check-Adjust cycle [3], and thereby serve as primary mechanism for controlling the variability of solution development.



The more frequent the points, the faster the learning. In complex systems development, local integration points are used to assure that each element or capability for the system is meeting its responsibilities in contributing to the overall solution intent. These local points must be further integrated at the next higher system level. The larger the system, the more such integration levels exist. Systems builders understand that the *top-level, least-frequent integration point* provides for the only true measure of system progress, and they endeavor throughout to achieve those points as frequently as possible. All stakeholders understand that *when timing of integration points slip, the project is in trouble.* But even then, this timely knowledge helps facilitate the necessary adjustments to scope, technical approach, cost or delivery timing needed to get the project tracking to revised expectations.

## Learn More

[1] Oosterwal, Dantar P. *The Lean Machine: How Harley-Davidson Drove Top-Line Growth and Profitability with Revolutionary Lean Product Development.* Amacom, 2010.
[2] Ward, Allan C. and Durward Sobek. *Lean Product and Process Development.* Lean Enterprise Institute Inc., 2014.
[3] Deming, W. Edwards. *Out of the Crisis.* MIT Press, 2000.

## Principle #5 – Base milestones on objective evaluation of working systems

*There was in fact no correlation between exiting phase gates on time and project success … the data suggested the inverse might be true.*
*—Dantar P. Oosterwal, The Lean Machine*

### The Problem with Phase-Gate Milestones

The development of today's large systems requires substantial investment—an investment that can reach millions, tens of millions, and even hundreds of millions of dollars. Together, systems builders and customers have a fiduciary responsibility to ensure that the investment in new solutions will deliver the necessary economic benefit. Otherwise, there is no reason to make the investment. Clearly, stakeholders must collaborate in such a way as to help ensure the prospective economic benefit *throughout* the development process and not just engage in "wishful thinking" that all will be well at the end. To address this challenge, the industry has generally applied a sequential phase-gated (waterfall) development process, whereby progress is measured—and control is exercised—via a series of specific milestones.

These phase-gate milestones are not arbitrary. They follow the apparently logical and sequential process of discovery, requirements, design, development, test, and delivery. Of course, it hasn't worked out all that well for many, as Figure 1 shows.

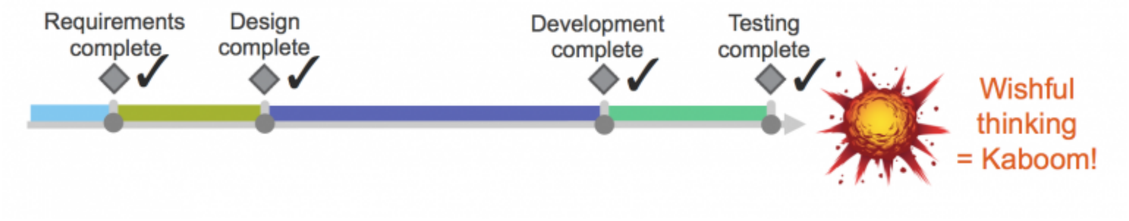


*Figure 1. The problem with phase-gate milestones*

The root causes of this problem is the failure to recognize four critical errors within the basic assumption that phase gates reveal real progress and thereby mitigate risk:
Centralizing requirements and design decisions in siloed functions that may not be integrally involved in the solution building.

Forcing too-early design decisions and "false positive feasibility" [1]: An early choice is made for the best known option at that time; development proceeds under the assumption that everything is on track; only later comes the discovery that the path chosen is not actually feasible. (Principle #3)

Assuming a “point” solution exists and can be built right the first time. This ignores the variability inherent in the process and provides no legitimate outlet for it. Variability will find a way to express itself.

Taking up-front decisions creates large batches of requirements, code and tests, and long queues. This leads to large batch handoffs and delayed feedback. (Principle #6)

## Base Milestones on Objective Evidence

Clearly, the phase gate model does not mitigate risk as intended, and a different approach is needed. Principle #4 – Build incrementally with fast, integrated learning cycles provides elements of a solution to this dilemma.

Throughout development, the system is built in increments, each of which is an integration point that demonstrates some evidence as to the viability of the current in-process solution. Unlike phase-gate development, every milestone involves a portion of each step—requirements, design, development, testing—together producing an increment of value (see Figure 2). Further, this is done routinely, on a cadence (Principle #7), which provides the discipline needed to ensure periodic availability and evaluation, as well as predetermined time boundaries that can be used to collapse the field of less desirable options.

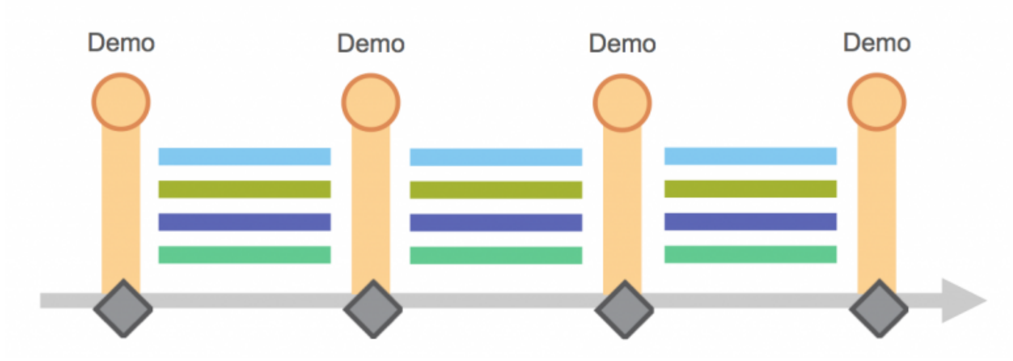


*Figure 2. Milestones based on objective evaluation of working systems*

What is actually measured at these critical integration points is subject to the nature and type of the system being built. But the system can be measured and assessed, and evaluated by the relevant stakeholders *frequently, and throughout the solution development life cycle*. This provides the financial, technical, and fitness-for-purpose governance needed to ensure that the continuing investment will produce a commensurate return.

## Learn More

[1] Oosterwal, Dantar P. The Lean Machine: How Harley-Davidson Drove Top-Line Growth and Profitability with Revolutionary Lean Product Development. Amacom, 2010.