**Chapter 7. Patterns for Strategy and Risk Reduction**

What is a chapter about strategy and business risk reduction doing in a book about cloud native patterns?

The biggest risk factor enterprises now face is not being able to respond fast enough to a changing environment. No matter what your sector, disruptor companies can show up at any time, and your existing competitors are also seeking the ability to shift direction and choose new markets very, very quickly. If you have a competitor that can shift in 6 months and you can only manage to shift in 12 months, then you are in trouble. Risk reduction today is the ability to respond to sudden or unexpected changes in market conditions when you don’t have much notice, in time to meet or beat the competition. And you achieve this ability through strategy.

How does this all play out in the real world? Well, we just witnessed WealthGrid making a smart move: realizing that the company’s long-term viability depends on adapting to changing environmental (market) conditions, the pragmatic implementation of which is to transform itself into a cloud native company.

In the effort to do just that, however, we also saw WealthGrid do some not so smart things during its two separate attempts at a cloud migration. So maybe now is actually a great time for a little strategy, with a nice side of risk reduction.

In this chapter we will introduce patterns that specifically shape and drive overall strategy in a cloud native organization. And, better yet, how to use them to reduce risk and build for long-term success, both during a transformation and then on into the future. We will examine patterns for:

* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Value Hierarchy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#value_hierarchy)
* [Business Case](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#business_case)
* [Executive Commitment](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#executive_commitment)
* [Transformation Champion](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#transformation_champion)
* [Vision First](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#vision_first)
* [Objective Setting](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#objective_setting)
* [Involve the Business](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#involve_the_business)
* [Periodic Checkups](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#periodic_checkups)
* [Data-Driven Decision Making](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#data_driven_decision_making)
* [Learning Loop](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_loop)
* [Learning Organization](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_organization)
* [Measure What Matters](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#measure_what_matters)
* [Research Through Action](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#research_through_action)
* [Gradually Raising the Stakes](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes)
* [No Regret Moves](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#no_regret_moves)
* [Options and Hedges](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#options_and_hedges)
* [Big Bet](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#big_bet)
* [Reduce Cost of Experimentation](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reduce_cost_of_experimentation)
* [Exit Strategy Over Vendor Lock-in](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#exit_strategy_over_vendor_lock_in)
* [Three Horizons](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#three_horizons_pattern)
* [Reflective Breaks](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reflective_breaks)
* [Designated Strategist](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#designated_strategist)

But first, let’s set the scene for how and when these patterns come into play. We will do this by getting acquainted with the leader of strategy and decision making at WealthGrid and taking a quick look at the challenges he faces in leading the company through an uncertain environment.

**Meet Steve**

Steve is WealthGrid’s CEO. He has always been a numbers/business kind of guy, never an engineer. Nonetheless, he recognizes the importance of investing in new technology to keep WealthGrid relevant in the market and healthy as a company overall.

Steve has spent his entire career in traditional companies with hierarchical structures that followed the Waterfall delivery approach. Working this way is now second nature to him. He has deep-rooted assumptions about how an organization works: there is of course a hierarchy. Certain people handle certain jobs, and everyone stays in their lane unless explicitly reassigned. Experimentation is bad, because novelty means risk. This kind of company is all about doing one thing, and doing that one thing predictably, reliably, and well.

So Steve’s one thing is running a bank: he is a banker, with deep expertise in how banks work. But banks are not just banks anymore; they are tech companies—or at least they need to act like one, if they want to stay in business. This puts a lot of pressure on Steve, and also WealthGrid’s other uppermost managers. They are all intelligent, well-educated executives who see the trend of IT becoming of central importance to any company like theirs. All the publications for senior execs are advising investment in machine learning, in automation, in cloud everything. But what the heck does all of that even mean? Steve has always had an IT department to handle the technical side of things; he has never needed to understand anything about it (in fact, Steve personally is so tech-shy that he just had his kids set up his new iPhone for him).

The good news for Steve is that what the company needs from him now is not deep tech knowledge. Or any technical knowledge at all, for that matter. What WealthGrid needs from Steve at this point is strategy.

**Strategize, Then Delegate**

Steve doesn’t know it yet, but being a banker actually gives him all the strategic skills needed to help WealthGrid navigate its way onto the cloud. The core business of a bank, after all, is to buy and sell money while taking a percentage in the middle, in the form of loan rates. What is really at work? Simple risk management: knowing the right time to borrow money and choosing the right customers to lend it to. It turns out Steve knows all about analyzing current state, establishing a desired goal, and identifying and evaluating procedures to avoid or minimize getting lost along the way.

There is one thing, however, that will get in Steve’s way: Steve.

Steve brings a strong risk management skill set to any cloud native initiative WealthGrid might undertake. But he is also a product of the environment where he built that experience: the traditional hierarchy using Waterfall delivery methods. In that environment, the executives make all the decisions—not just strategy, but many execution details as well. They bring in both inside and outside experts and architects, create many reports and documentation, and in general take a long time to think things through. Once the plan is set, it is handed off to middle managers to oversee the plan’s execution, exactly as created with no diverging allowed, at the team level.

Waterfall and predictability and planning and documentation: we don’t mean to make it sound like these are bad things. Far from it. They are tools for creating a consistent, stable, and efficient delivery environment and, used at the right time and in the right way, are good and important things. The right time for them, however, is not during the creative and innovative process that a company must undergo to transform itself into a cloud native entity. But it doesn’t stay in creative mode forever. Once the new system is in place and cloud native knowledge has been infused into the company, things will rebalance to focus mainly once again on proficient delivery of core business value.

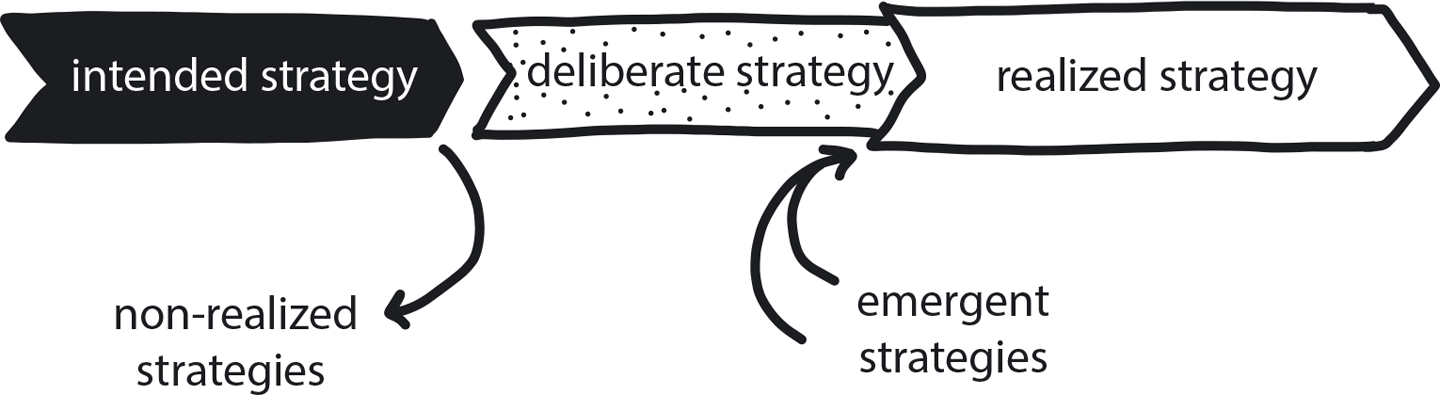
Companies like WealthGrid, with their highly proficient and historically profitable way of working, are accustomed to operating in a stable environment. Their market had long been steady and reasonably predictable, their competition known and manageable. They were good at deciding a smart course of action, doing a lot of research to create the best implementation, and then producing a detailed plan of action to make it happen on time—usually, a two-year span—and on budget. During that time span the focus was on execution, not re-evaluating the original goals, so the plan never changed.

We call this type of long-term, predictive, and above all fixed way of working *static strategy.* This was how business worked, and worked very well, for a very long time…until the steady and predictable world became today’s reality of constant rapid change.

Now a strategic leader’s job is to create dynamic strategy. This means watching the company’s market, competitors, and other environmental factors both current and emergent in order to make comparatively quick and short-term decisions about how to respond and which direction to go next. Quick and short-term moves that are frequently re-evaluated, and adjusted if necessary, are what’s needed now in order to respond to ever-faster change conditions. The Dynamic Strategy pattern, inspired by Henry Mintzberg’s theories on emergent strategy,[**1**](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#ch07fn1) describes the practical application of these concepts.

**Pattern: Dynamic Strategy**

*Today’s tech-driven marketplace is a constantly shifting environment, no matter what business you are in—so your game plan needs to shift right along with it (*[*Figure 7-1*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy03)*).*



**Figure 7-1. Dynamic Strategy**

A company is commencing a cloud native transformation, has completely and successfully migrated its platform, or is at any point along the way. Traditionally, companies could set strategy, translate to objectives, and then move comfortably into long-term execution without ever looking back to re-evaluate. Today’s environment is uncertain and constantly changing. Cloud native evolved in response to this reality as a nimble, scalable, and fast framework for handling constant variability.

**In This Context**

**Not responding quickly enough to market changes or new information may lead the company to continue building products according to an old strategy that is no longer fully relevant. The original strategy could be realized in its totality, but in the meanwhile competitors could come up with better products, technology could change, and much better opportunities could be missed.**

**Ultimately, the company may end up with exactly what was planned in the beginning of the project—only to find that this is not what they actually need when they finally go to market.**

* New technologies are constantly introduced, bringing unexpected new competitors to every sector.
* Low-certainty projects carry extremely high risk. Any decision made early in the project is highly likely to be uncovered as a wrong decision later.

**Therefore**

**Continually re-evaluate circumstances as the initiative moves forward.**

Check if the relevant products are still in demand and if the chosen technologies, organizational structure, and development processes are still the best for the most successful delivery. Always monitor the competition to adjust delivery planning and releasing optimal functionality to maximize market impact.

* If adjustment is required, use the Gradually Raising the Stakes pattern to reduce risk and make better decisions.
* Dynamic Strategy is essential for doing cloud native right, from inception to completion. This pattern is so essential that it is built into our transformation design from the beginning.
* Use Reflective Breaks for the executive team to review and reassess strategy.

**Consequently**

**The executive leaders are aware when the environment changes and adjust strategic goals to keep the company heading in the right direction.**

* + Digital transformation is a constant balancing act between innovation and pragmatism—delivering what earns the company’s money right now.
* + If a disruptive new competitor enters the market unexpectedly, the CEO and board can shift the company toward increased creativity to keep up.
* + If quality is suffering, the leaders can redirect strategy to refocus on delivery until the situation improves.

**Common Pitfalls**

Being overly reactive and changing direction *too* often, which is distracting, wastes time, causes instability, and can siphon resources away from delivery. The company’s Designated Strategist can provide helpful perspective when leaders are contemplating a potential strategy adjustment.

**Related Biases**

*Irrational escalation/sunk-cost fallacy*

The urge to keep investing in a project, despite new evidence that the decision was probably wrong, because you’ve already sunk so much into it.

*Illusion of control*

The tendency to overestimate one’s degree of influence over external events. Many complex and emergent processes are difficult to steer, much less control. Sometimes we need to embrace some uncertainty to ultimately get results.

**Related Patterns**

* [Designated Strategist](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#designated_strategist)
* [Reflective Breaks](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reflective_breaks)
* [Three Horizons](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#three_horizons_pattern)
* [Executive Commitment](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#executive_commitment)
* [Value Hierarchy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#value_hierarchy)

Today’s tech-driven marketplace, no matter what business you are in, is the ultimate uncertain environment. If you have a strategy, it needs to change all the time. Dynamic Strategy is the transformation pattern that teaches us to observe how the world is changing and to continually evolve and adjust strategy to match. And this responsibility sits squarely with the company’s executive leadership.

WealthGrid’s first transformation attempt doesn’t really count, since it was Jenny and her team trying to do it as a side project. There was not any strategy at all and upper management, other than perhaps granting permission, was barely involved. But they should have been. Ideally, Jenny would present a problem to the board, demonstrate the need for a strategic decision, ask for the decision, and then get authorization to execute on that strategic decision. Obviously this didn’t happen, so instead Jenny and her team did their best but ultimately stumbled around for six months.

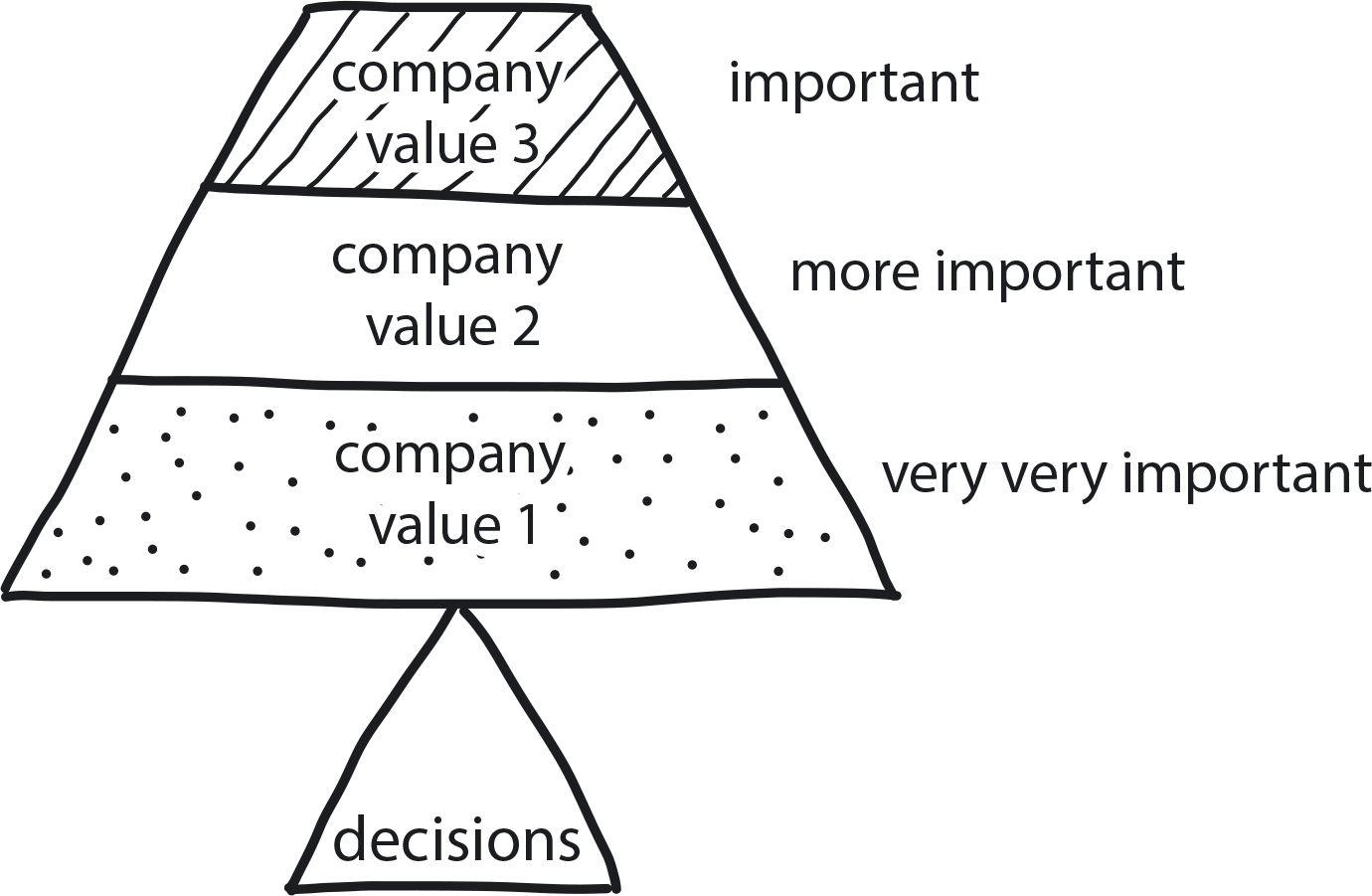
For the second attempt, though, everyone got on board and things still went sideways. The problem? WealthGrid’s leadership was involved, true, but they fell victim to the very common pitfall of static strategy: give scope, allocate budget, set deadline, trust team to deliver…and never look back to reassess the original goals and plans to see if they are still needed and still working.

Once a working strategy is defined, the middle managers take over to translate the strategy into specific objectives and deliverable goals. Their job is to explain these to their teams and then step back and let the teams decide exactly how to execute them. The delivery schedule emerges bottom up: the teams report the timeline of when they can achieve what. The teams have the freedom to choose their own tools and approach (within a set of company standards, since too much freedom leads to chaos and conflicting technologies) so long as they deliver functional results. This is all a single chain that has to function in coordination. If any of the links in the chains go too fast or too slow, the overall throughput suffers. (This is described more deeply in the Decide Closest to the Action pattern, which is described in [Chapter 8](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#patterns_for_organization_and_culture).)

Since decisions regarding strategy now need to be made quickly and since the power—and responsibility for making all other kinds of decisions—is being distributed to middle management and execution teams, it’s essential to share a consistent set of values and priorities that guide decision making across the organization.

**Pattern: Value Hierarchy**

*When an organization’s values are clearly stated and prioritized, as well as fully internalized across the company, people have the basis for making day-to-day decisions without needing to seek consent or permission/approval (*[*Figure 7-2*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#value_hierarchy01)*).*



**Figure 7-2. Value Hierarchy**

The company’s cloud native transformation has just begun, and there is a lot of uncertainty. People are still learning about the tech, and the organizational structure is evolving. Teams are becoming independent, and there are many moving pieces.

**In This Context**

**Without a clear understanding of the company values and the priorities, people have no easy way to connect their daily work to the company strategy. In such a situation, different teams may make conflicting decisions or waste a lot of effort on low-priority tasks.**

* Gaining consensus from all stakeholders is time-consuming.
* In traditional organizations managers make all the decisions.
* The market is changing frequently, with new competitors appearing unexpectedly.
* Cloud native technology is constantly and rapidly evolving.

**Therefore**

**Create an ordered list of clearly stated values to simplify decision making and guide behavior in an uncertain environment.**

* Identify the company’s values: what is important to us?
* Formulate these simply and clearly.
* Rank these values in order of importance.
* Incorporate this value hierarchy into the company’s culture and identity by broadcasting it across the organization.

**Consequently**

**Teams and individuals in an organization are able to make decisions with confidence.**

* + Organizational principles and priorities are clear and commonly understood by all.
* + People can easily make informed choices that reflect both the company’s values and best interest.
* + In a constantly changing environment, there is a stable and constant point of orientation.
* − Too-frequent changes to the value hierarchy can cause confusion and chaos.

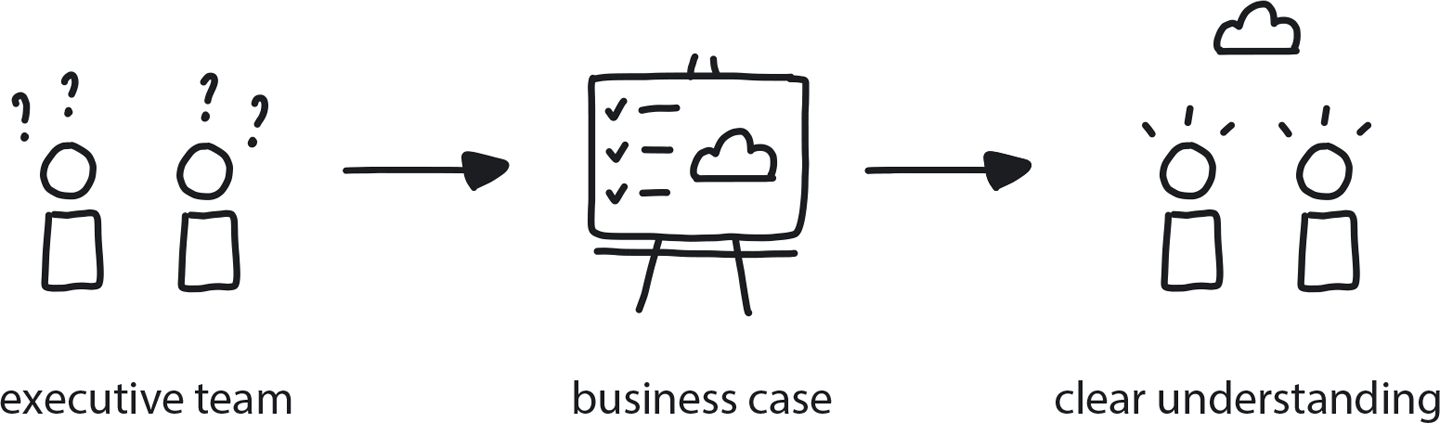
**Related Patterns**

* [Decide Closest to the Action](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#decide_closest_to_the_action)
* [Objective Setting](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#objective_setting)
* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Vision First](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#vision_first)

Once a Value Hierarchy is in place, the next step is to make sure the company will truly benefit from investing in a cloud native transformation.

**Pattern: Business Case**

*Before launching a cloud native transformation, an enterprise’s leadership must make sure the initiative is needed and that the benefits will justify the investment (*[*Figure 7-3*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#business_case03)*).*



**Figure 7-3. Business Case**

A company is experiencing pressure, whether from the market or internally, to move to cloud native. The executive team is contemplating making the move to cloud native, but there is little internal knowledge of cloud native tech and culture or understanding of its benefits.

**In This Context**

**Cloud native transformations are a big commitment, requiring significant investment of budget, time, and team talent. Too many organizations, though, get caught up in the hype of the cloud conversation and make decisions without understanding how exactly a transformation fits their business needs and goals. The risk is especially high for organizations that have already established rapid and significant internal momentum toward making this move.**

* The traditional model is for organizations to be massively risk-averse to minimize uncertainty at all costs.
* Change-averse culture avoids new technologies or experimental approaches. Cloud native architectures are conceptually different from traditional approaches, merging careful upfront planning with flexible and mutable, experimentation-based implementation.
* Tech teams are eager to get started with the transformation, even before a business case is established.
* Cloud native is complex, and the benefits are not easily visible.
* Companies often don’t have good baselines for how much it actually costs them to do development in their current approach, so trying to evaluate cost savings or time savings in a new approach is difficult.

**Therefore**

**Create a formal business case to help educate the organization’s executive team, to evaluate how the transformation will serve the company’s goals, and to create a clear vision for where the organization is headed.**

* Discussion should involve business stakeholders, as well as information-gathering interviews with internal and external resources.
* Explain key cloud native advantages, including acceleration of business velocity, resilience, scalability, potential cost savings, and the ability to quickly act on feedback from customers to improve products.
* Evaluate the realistic scope and cost of the transformation, including technical and organizational changes.

**Consequently**

**The business case for a cloud native transformation is clear. The company’s decision makers have a clear understanding of the initiative and the advantages it will confer when complete. They are ready to move forward.**

* + They are prepared to allocate the necessary budget and resources that such a large project will require.
* + Enhanced recruitment and retention of tech staff eager to work with modern systems.

**Common Pitfalls**

If the company has recently invested in new (non-cloud-native) infrastructure or other modernization efforts, there can be significant resistance to now investing in cloud native.

Executive team underestimates the cost, time, and resource demands of the transformation as they perceive it a purely technological change with a reasonably predictable scope. They may have a long history of collaboration and trust in their IT team, but in the case of the cloud native transformation, the IT team is typically not in a position to evaluate the full impact that an initiative will have upon the entire business.

**Related Biases**

*Bandwagon bias*

Everyone is talking about doing Kubernetes, so we better do it too!

*Status quo bias*

Everything is fine right now, there is no need to rock the boat by introducing new tech or changing how we do things. That is risky!

*Ostrich effect*

Competition is stable in our sector; we don’t need to worry about disruption.

*Ambiguity effect*

Cloud native is complex and the benefits are not easily visible, so better just to keep things the way they are.

*Sunk-cost effect*

If a company has invested significant budgets into existing infra or other modernization projects, the tendency will be to continue investing there instead of switching to a new architecture—even if it is demonstrably superior.

*Pro-innovation bias*

The belief that technology can fix all problems can be used as a nudge toward positive action: increase the chance of approval by showing innovation as an explicit benefit

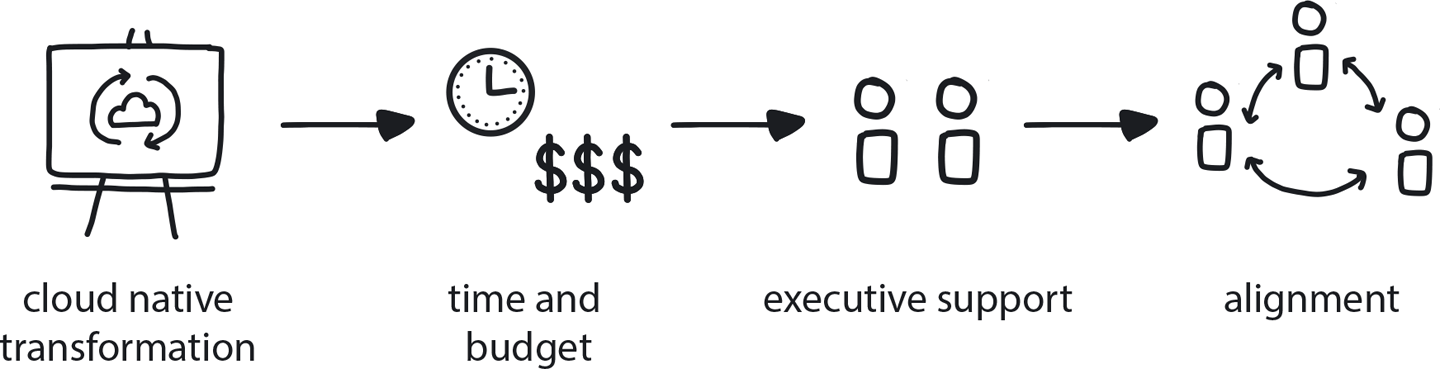
**Related Patterns**

* [Executive Commitment](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#executive_commitment)

Once a genuine business case has been established, it’s a quick next step to our next pattern: Executive Commitment.

**Pattern: Executive Commitment**

*To ensure allocation of sufficient resources and reasonable delivery time frames, large-scale projects such as cloud native transformation require strong commitment from the top executive team (*[*Figure 7-4*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#executive_commitment03)*).*



**Figure 7-4. Executive Commitment**

An enterprise that is using Waterfall or Agile software development practices has made a clear decision to adopt cloud native, with a business case supporting the transformation.

**In This Context**

**Cloud native transformations require significant changes in all areas of an organization, from infrastructure to processes to culture. These changes place large demands on the organization’s budget and time.**

* Clients continue demanding fast delivery of new functionality, leaving no slack for structural changes.
* Executive performance is measured by P&L (profit and loss statement), reducing incentives to invest in long-term structural improvement such as cloud native transformation.
* Executives likely don’t understand the full scope of the cloud native transformation.
* Successful adoption of cloud native may significantly speed up feature development and increase tech teams’ job satisfaction.

**Therefore**

**Announce the cloud native transformation as a high-priority strategic initiative that is important to the company and that has explicit support from the executive management**.

* + Prepare a transformation strategy and the allocation of adequate resources and budget.
* + Publicly announce the cloud native transformation as a strategic initiative. This creates company-wide alignment and awareness, while setting the expectation of collaboration from all departments.

**Consequently**

**The company is aligned around common goals and everyone understands priorities for the transformation.**

* + All departments follow a unified vision, thus avoiding independent silos that lead to inconsistent, or even conflicting, implementation.
* − Personal public commitment by execs makes it difficult for them to reverse the decision later, due to public embarrassment related to changing their minds.

**Common Pitfalls**

The focus is on technical changes only, without including the organizational changes that are also essential for a cloud native transformation to succeed. Or the initiative gets treated like just another tech/infrastructure upgrade, instead of a true paradigm shift. In such cases Executive Commitment exists, only for a wrong scope of the initiative.

**Related Biases**

*Authority bias*

The tendency to attribute greater accuracy to the opinion of an authority figure and be more influenced by that opinion can be used as a nudge for the team to act. Even if those leading the decision to go cloud native are not deeply technical, their status in the company grants them authority to drive the change.

*Bandwagon effect*

The tendency to do (or believe) things because many other people do (or believe) the same thing, related to groupthink and herd behavior, can be used in this case as a nudge to enlist wide support for the initiative.

**Related Patterns**

* [Business Case](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#business_case)
* [Vision First](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#vision_first)
* [Core Team](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#core_team)
* [Objective Setting](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#objective_setting)
* [Transformation Champion](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#transformation_champion)
* [Internal Evangelism](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#internal_evangelism)

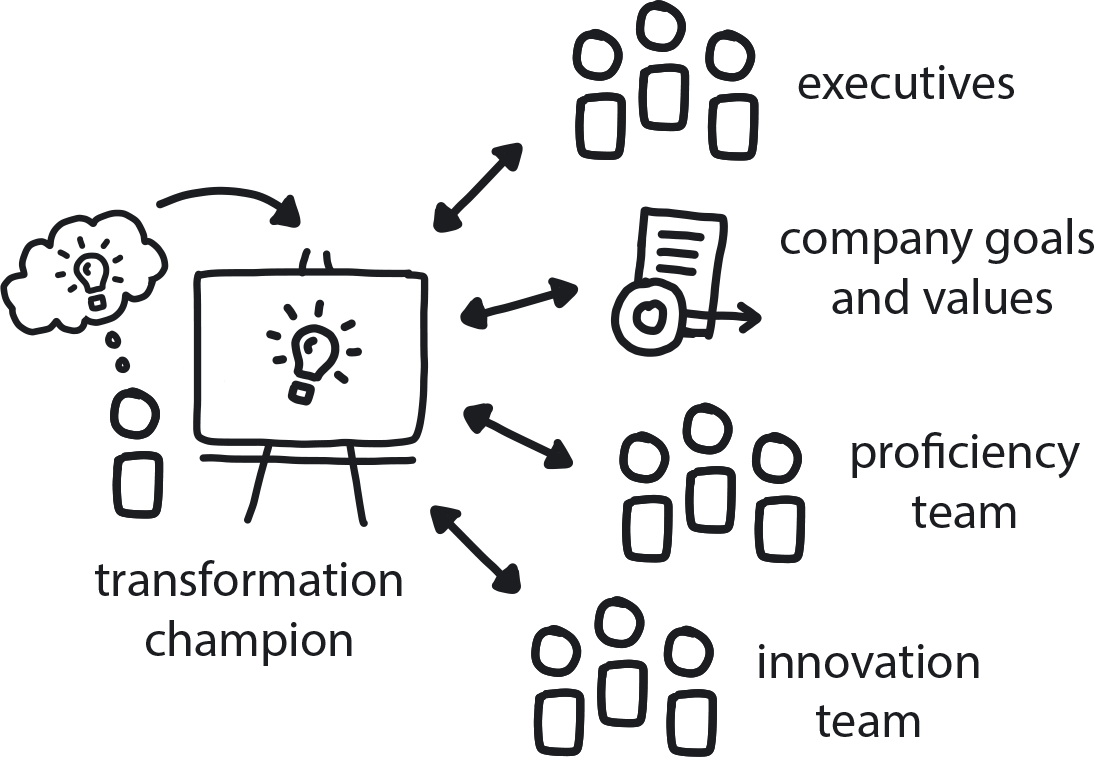
Establishing executive commitment to the initiative is essential, but you’re still going to need a designated hands-on person to lead it. That person is the transformation champion.

**From Theory to Execution**

Choosing a transformation champion is an important early milestone in the transformation’s forward progress. We talk much more about this role later in [Chapter 11](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch11.html#applying_the_patterns), but for now the short version is: this is where things start getting real! Now come the first functional steps for getting the transformation underway.

**Pattern: Transformation Champion**

*When a person is promoting a good new idea that can take the company’s goals and values into the future, recognize and empower them to lead the action (*[*Figure 7-5*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#transformation_champion01)*).*



**Figure 7-5. Transformation Champion**

Market conditions are changing fast, and the company needs to evolve to keep up.

**In This Context**

**Successful established enterprises focus on proficient delivery of their core products or services and often forget how to be innovative. When a disruptive competitor appears, it is difficult for them to respond quickly and effectively. There are always a few people within the organization who see the future better than others. An even smaller subset of these are willing and able to take organized action, but many organizations ignore them and waste the opportunity to encourage healthy leadership.**

**Without such motivational leaders, the initiative often falls flat and keeps going only after management exerts some bureaucratic pressure to push it forward.**

* Change creates anxiety, so people tend to avoid it.
* There is little knowledge or experience within the organization around cloud native architecture/tech.
* An established enterprise accustomed to delivering a highly proficient product/service will likely struggle to innovate.

**Therefore**

**Recognize the person (or group) who has triggered the movement and name them transformation champion. Authorize them as designated advocate for the initiative. Name a different person to this role only if there is a very compelling reason to do so.**

There are conditions under which the person who started things is not the right transformation champion. It could be that they are not able to see the wider business perspective, as sometimes happens with technical people, or it may be that they are abrasive (as sometimes also happens with technical people). In that case, you want to let them lead the technical portion of the transformation and then find a partner for them who can share the lead on the cultural and organizational aspects of the transformation.

The transformation champion is a person or a small team who understands both the transformation and the company objectives, is well-connected within the organization, and is highly motivated to promote the transformation. Unless they are given authority, however, they will be unable to stimulate effective change across the organization.

* Don’t create a champion: discover the one who is already there.
* Empower this person/group to lead the initiative.
* Publicize their role as authorized champion.

**Consequently**

**The transition has a focal point for organizing the transformation initiative and a champion in charge of driving it forward. The transformation champion is connected with both the proficient and innovative branches of the transformation and can act as a bridge between them.**

* + The transformation champion evangelizes the initiative across the organization.
* + The organization’s leaders publicly empower the champion with the authority to lead.
* + The company now has a channel for reintroducing innovation while maintaining proficiency.

**Common Pitfalls**

The transformation champion can come from anywhere within a company, but in an organization with a strong internal hierarchy there is always pressure to name higher-up managers to leadership positions. The wrong person in this role—and they can be wrong for a variety of reasons—can seriously hinder the migration.

**Related Biases**

*Status quo bias*

Overcome the natural resistance/anxiety around organizational change with internal marketing about the cloud native initiative and how it will help everyone.

*Hostile attribution bias*

Internal transformation leaders run into this form of resistance where people fear that a new cloud native will take their jobs away or threaten their position in the company. We shouldn’t think that this is their real opinion, as it is a normal human bias, but those acting under this bias can do real damage by passively resisting, or even actively thwarting, a transformation.

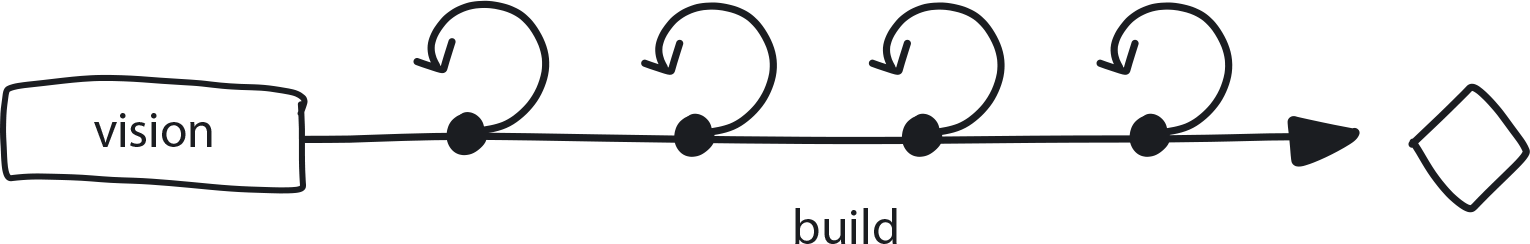
**Related Patterns**

* [Business Case](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#business_case)
* [Executive Commitment](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#executive_commitment)
* [Core Team](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#core_team)
* [Vision First](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#vision_first)
* [Objective Setting](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#objective_setting)
* [Internal Evangelism](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#internal_evangelism)
* [Gradually Raising the Stakes](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes)

The transformation champion’s very first task is to create a big-picture vision for the transformation, defining its initial shape and direction. This doesn’t have to be exact or detailed, and it’s important to not let this vision become set in stone. Dynamic Strategy means we will refine and adjust as necessary as we go along.

**Pattern: Vision First**

*Defining a high-level transformation path as the very first step helps set the right course through an uncertain environment (*[*Figure 7-6*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#vision_first01)*).*



**Figure 7-6. Vision First**

The company has established a Business Case for going cloud native, achieved Executive Commitment, and is ready to move forward with a transformation initiative.

**In This Context**

**The company needs to define a clear and achievable vision that can be translated into specific executable steps.**

* Without an overall consistent vision, different teams will make independent and, frequently, conflicting architectural decisions.
* The combination of limited experience and lack of extra time and flexibility for research leads to pursuing cloud native implementation using “well-known ways.”
* In many companies, enterprise architects are responsible for creating a detailed architecture. Many enterprise architects lack sufficient theoretical or practical experience in the cloud native approach.
* Agile methodologies, widely adopted in the contemporary business world, create pressure to produce results early and onboard teams to new systems very quickly.

**Therefore**

**Define and visualize the organizational structure and architecture of the whole system upfront.**

* This can be either requested from external sources or uncovered by a series of small research and prototyping projects ranging from a few hours to a few days each.
* Keep the vision high-level to allow freedom of choice during implementation (not dictating specific tools or approaches) yet also specific enough to provide clear goals and desired outcomes.
* Revisit the vision regularly to evaluate status and adjust the plan if necessary.
* Executive Commitment paired with leadership by the Transformation Champion are essential for successful vision creation—this indicates the path for the teams to follow.

**Consequently**

**All teams have a clear guiding principle for the implementation phase.**

* + The teams can start producing the lower-level architecture and translate it to the backlog of tasks.

**Common Pitfalls**

Attempting to move to cloud native by using old techniques/processes, because that is what you know how to do.

**Related Biases**

*Law of the instrument*

An overreliance on a familiar tool or methods, ignoring or undervaluing alternative approaches. “If all you have is a hammer, everything looks like a nail.”

**Related Patterns**

* [Executive Commitment](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#executive_commitment)
* [Transformation Champion](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#transformation_champion)
* [Core Team](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#core_team)
* [Objective Setting](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#objective_setting)

From Vision First we move directly into Objective Setting. This is the stage where responsibility begins to shift from the organization’s executive leadership into the hands of middle managers/project leaders. Now it is their turn to begin converting high-level vision into explicit and visible goals and actions.

**Pattern: Objective Setting**

*After establishing a transformation vision, the next step is to translate it into pragmatic goals and actions for moving the initiative ahead (*[*Figure 7-7*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#objective_setting03)*).*



**Figure 7-7. Objective Setting**

A company is committed to adopting cloud native technology and transforming its culture to suit. The Business Case is established; the executive team is fully committed and has devised the initial Dynamic Strategy. Middle management and the rest of the company are ready to take the next steps.

There is still an overall low level of cloud native knowledge on all levels of the company, and the transformation plan is still very high-level.

**In This Context**

**There is commitment to and a vision for transformation, but concrete steps for getting there still need to be defined.**

* There is little knowledge or understanding of cloud native tech and culture within the organization.
* Change creates anxiety and uncertainty.
* Change also creates opportunities: if you keep doing the same thing, what is the chance that you are going to get new results?
* When scope changes significantly, new executive commitment must be made to the altered project.

**Therefore**

**Executives need to hand over the high-level strategy to middle managers to translate it into specific and tangible objectives for their teams. Keep redefining the strategy and the objectives based on known information, not guesswork.**

Objective-setting is done by mid-level managers. It is essentially the art of taking high-level vision, making it explicit and visible, and then explaining the priorities to those involved in execution so they can make better decisions on their own.

For example, a company may establish the high-level strategy of being able to run a full experimental cycle within days (define, build, deliver, and collect feedback). The specific and tangible objectives for this strategy would be: use cloud native tech (containers, microservices, dynamic scheduling, etc.) and a Continuous Integration/Continuous Delivery build approach. These objectives in turn can be divided into even more specific sub-areas, with more detailed objectives created for each one.

* Keep learning about the market.
* Keep adjusting the strategy in response to new conditions in order to uncover new information.
* Make the strategy explicit and visible, and explain the priorities to all involved so they can make better decisions on their own.

**Consequently**

**The initial strategy is continually improved, adjusted, and translated into clear and tangible objectives. The relevant teams in the company know what they need to achieve and are constantly providing new information to upper management.**

* + The team is aware of the latest direction and priorities.
* − There is cost to monitoring the market and current conditions; requires time and effort.
* − Continuous changes in the strategy lead to confusion and possibly frustration in the teams.
* − Difficult to change direction, especially once you have momentum in a certain direction.

**Common Pitfalls**

* The executive team itself does continuous, detailed planning and objective-setting for the rest of the company without significant input from the teams, or alternatively seeks input from an enterprise architect with almost no knowledge or experience in cloud native. Objectives defined this way and with no experimentation will have no chance of being correct and will provide little motivation for the rest of the company.
* The executives will just give an order to middle management to execute the cloud native transformation without closely observing the progress and providing executive support. In such cases, the commitment is less than complete, as the full responsibility gets rolled onto middle managers, who will have limited time, budget, and other resources to allocate to a full-scale transformation.
* Adjusting strategy only every two to three years is the most common mistake. It is a relic of the long-range predictive Waterfall planning process, and executives will stick to a strategy for a very long time—regardless of the actual state of the market and the company—because they are accustomed to following a “set it and forget it” static strategy.

**Related Biases**

(Most of them, really—the majority of biases are against making any changes, and here we are saying “Make all the changes!”)

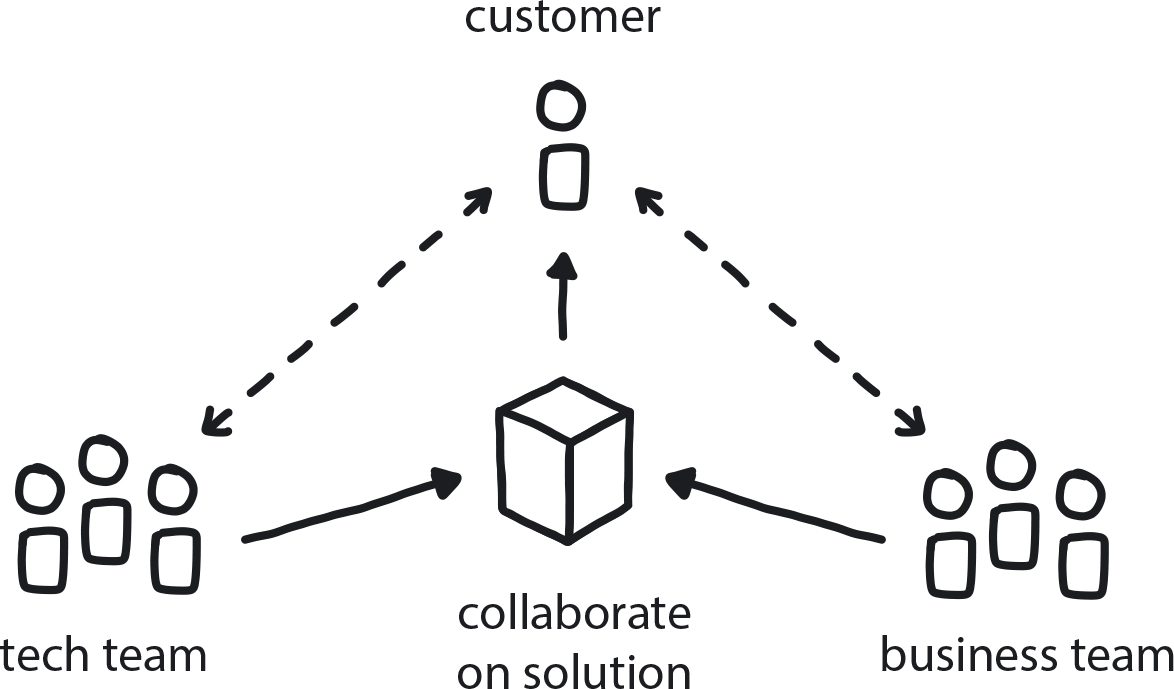
**Related Patterns**

* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Executive Commitment](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#executive_commitment)
* [Vision First](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#vision_first)
* [Gradually Raising the Stakes](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes)
* [No Regret Moves](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#no_regret_moves)
* [Big Bet](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#big_bet)
* [Options and Hedges](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#options_and_hedges)
* [Value Hierarchy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#value_hierarchy)

How do we know what the most valuable, and useful, goals for this transformation might be? As the engineering teams start to get hands on with executing the technical objectives, business teams need to be involved as well to help target the right enterprise outcomes.

**Pattern: Involve the Business**

*The business teams and the tech teams need to collaborate to create an effective customer-feedback loop that drives product improvement (see*[*Figure 7-8*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#involve_the_business03)*).*



**Figure 7-8. Involve the Business**

The company is aiming to deliver value to customers as fast as possible while learning from customer feedback and steadily improving the product.

**In This Context**

**When developers are running quick iterations without involving customer-facing people, the value could be limited to tech solutions only. Businesspeople, however, can’t run full-tech experiments.**

* Departments in large enterprises tend to focus on their own tasks.
* Long development cycles can make it difficult to incorporate customer feedback in a meaningful way.

**Therefore**

**Create close collaboration between dev teams and the business to define experiments for testing new customer value and quickly executing them.**

To close the learning loop, experiments and changes should include everyone from developers to the business teams to the customer and back, and the results should then be used to define and drive the next change.

* Customer feedback drives new ideas, and fast turnaround on a smaller scale (shorter delivery cycle) helps both dev and business teams learn more quickly.
* Development teams need to embed measurement of customer feedback into products.
* The business teams need to work with developers to define what is valuable for customers.

**Consequently**

**Your products (or services) can change quickly in response to actual customer needs and desires.**

* + Teams are learning from each other.
* − Effort is required to build this communication and collaboration into the delivery cycle as a regular event, and it creates one more step in the process.

**Related Biases**

*Curse of knowledge*

When better-informed people find it extremely difficult to think about problems from the perspective of less-informed people—in this case, tech teams needing to explain technical things to the businesspeople, and the businesspeople needing to explain customer-facing needs/issues as executable deliverables.

*Dunning-Kruger effect*

The technical people make decisions because they think they know what the business needs. Not only is this not their area of expertise, but tech teams are generally inward-facing. They are not engaged in the kinds of customer-facing roles that would grant them the perspective to also deeply understand business needs.

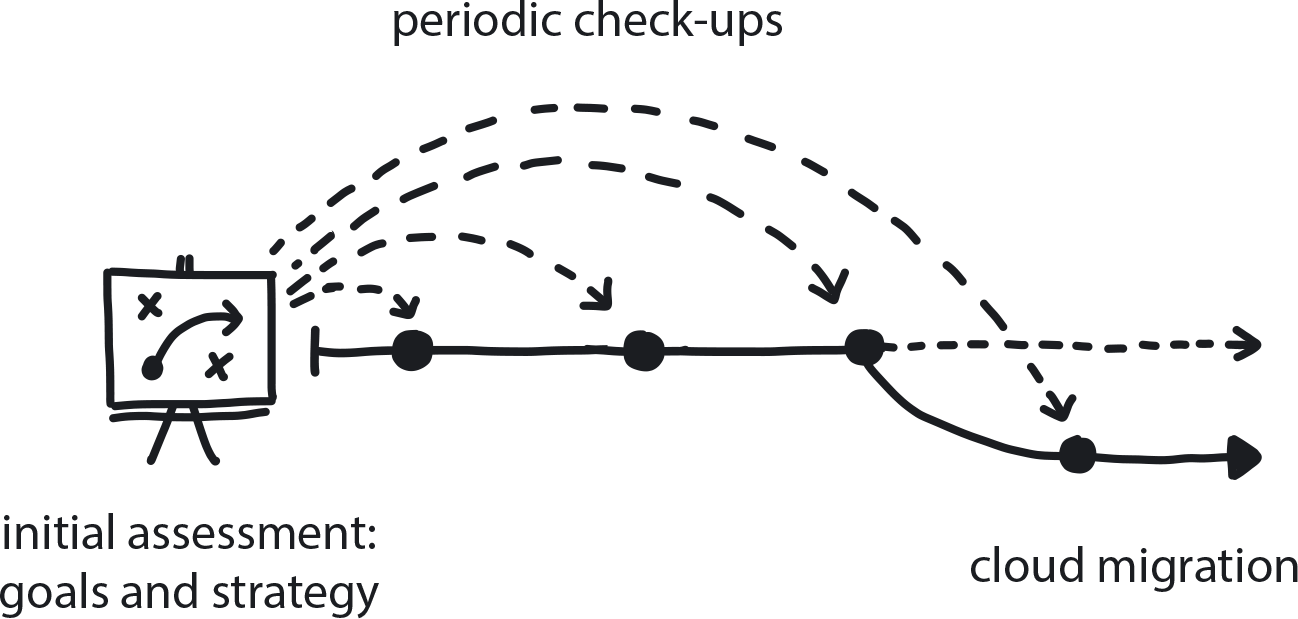
**Related Patterns**

* [Data-Driven Decision Making](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#data_driven_decision_making)
* [Personalized Relationships for Co-Creation](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#personalized_relationships_for_co_creat)
* [Learning Loop](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_loop)
* [Measure What Matters](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#measure_what_matters)

As the transformation moves ahead, it’s important to continue re-assessing the objectives—the business environment can change without warning, and the transformation goals need to change to match it.

**Pattern: Periodic Checkups**

*Frequently reassess vision and objectives to ensure these remain the correct direction to proceed as the business environment shifts (*[*Figure 7-9*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#periodic_checkups01)*).*



**Figure 7-9. Periodic Checkups**

The company is immersed in the new and changing environment of cloud native. Goals and strategy are set, the Core Team is underway, and the transformation is going full speed ahead. But the business goals might be changing as the environment changes.

**In This Context**

**Teams focused solely on execution without pausing to assess and reappraise the direction they’re going in might achieve what they originally planned—but not what they ultimately needed, because circumstances changed along the way.**

* Teams tend to work by creating strategy, choosing a direction, and then continuing on without reviewing changes in environment or goals.
* Visions and plans require adjustment to meet real-world conditions. In the words of German military strategist Helmuth von Moltke, “No battle plan survives contact with the enemy.”
* Cloud native ecosystem changes quickly, with new tech emerging all the time.
* It is impossible to predict what the cloud native world will look like in a year.
* Under delivery pressure, teams don’t have a chance to look around them, only straight ahead.
* The process itself may require adjustments as the team matures.

**Therefore**

**Make sure that you and your team are still on the right path. Assess the current situation with regard to initial strategic decisions.**

* Run a gap analysis assessment every month during the transformation.
* Have a standard assessment template.
* Invite independent experts to provide a second opinion.

**Consequently**

**The Core Team meets regularly to assess current conditions and can adjust direction as circumstances require.**

* + Time to reflect and celebrate progress.
* − Frequent adjustments may distract from overall delivery.

**Related Biases**

*Ostrich effect*

The tendency to ignore an obvious problem because you don’t have time to investigate it, or perhaps simply don’t understand it well enough to see any solution.

*Pro-innovation bias*

The tendency to be excessively optimistic about a new technology, assuming it will solve all your problems while failing to recognize its limitations and weaknesses.

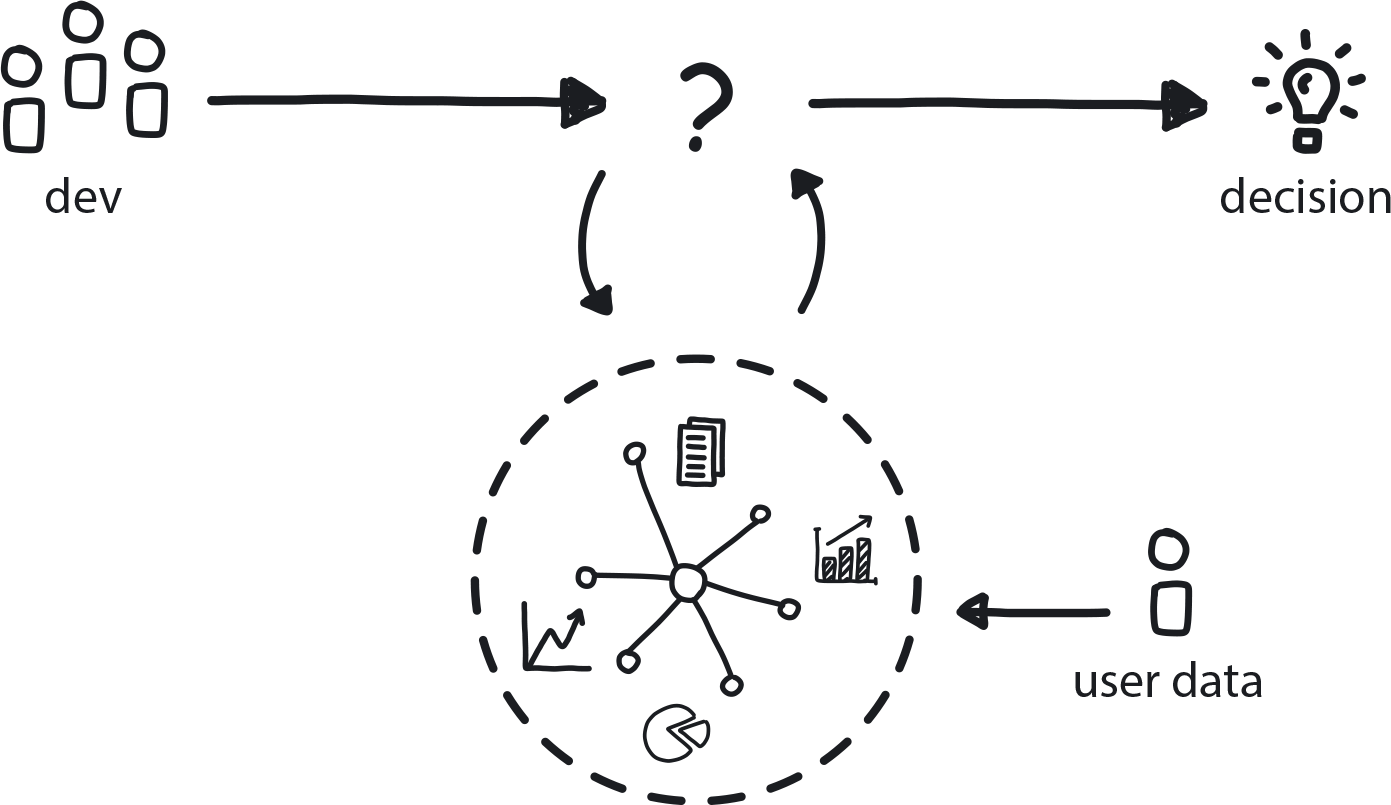
**Related Patterns**

* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Reflective Breaks](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reflective_breaks)
* [Designated Strategist](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#designated_strategist)
* [Three Horizons](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#three_horizons_pattern)

The best decision is an informed decision. In cloud native, we collect information and draw insights to guide strategy and next steps. The Data-Driven Decision Making and Learning Loop patterns describe how to incorporate this into your company in a practical way.

**Pattern: Data-Driven Decision Making**

*Collect data, extract patterns and facts, and use them to make inferences to drive objective decision making (*[*Figure 7-10*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#data_driven_decision_making03)*).*



**Figure 7-10. Data-Driven Decision Making**

A company is moving to cloud native and sees the complexity and number of components growing exponentially. Each component is built separately. Competitors are changing products on a daily basis.

**In This Context**

**Managers make decisions based on their expectations from previous experience, which might not apply in the new and unknown environment of a cloud native system.**

* Long development and release cycles (six months to a year, or even longer) that are typical in the Waterfall development approach result in products that are no longer relevant by the time they reach the customer.
* Software systems and users are complex systems. It’s impossible to predict behavior.
* In hierarchical organizations managers protect their status by knowing everything and deciding everything.
* Technically it is easy to collect data.
* Measuring the wrong things may lead to bad results.

**Therefore**

**Make product decisions based on data collected from actual users (observability, measure what matters).**

Data can be collected automatically by embedding the data collection tools into the platform and all applications before they are released to customers. However, developers need to be careful not to be overly reliant on, or trusting of, user feedback/data. Users often don’t know what they want, especially in regards to radical innovation.

* Apply the A/B Testing pattern to evaluate different options by exposing them to the real customers.
* Similarly, the Learning Loop pattern builds on Data-Driven Decision Making.
* Collect feedback and clicks/business measurements following the release of changes.

**Consequently**

**The team can quickly make decisions based on objective measurements.**

* + Instead of arguing on direction, team can set up experiments and measure.
* − It is easier to follow the herd.
* − It is not always easy to measure and interpret the right things.
* − Sometimes data and users are wrong, and the team needs to trust their instinct and pivot to a radical change.

**Common Pitfalls**

The first pitfall is collecting insufficient data to make real decisions. The team has a feeling that they have everything they need while really they have plenty of blind spots and the full picture is different.

On the other side, teams become overly reliant on the data collected from users and forget that *users themselves don’t always know what they really want*. Human intuition is a valuable resource based on thousands of years of evolution and is especially valuable to the process of radical innovation. If back in 2012, a year before Twitter was founded, you had asked people how much they would want or need an SMS service over the internet, the answer would probably have led Twitter’s founders to drop their idea to create such a service.

**Related Biases**

*Law of the instrument*

Overreliance on a familiar tool or methods, ignoring or undervaluing new or alternative approaches. In this context we would see managers attempting to lead a move to cloud native using old techniques/processes and cultural assumptions. For example, micromanaging work done in a highly distributed team building microservices or insisting that component services all release together instead of independently.

*Status quo bias*

Why change things if they have always worked before?

*Hostile attribution bias*

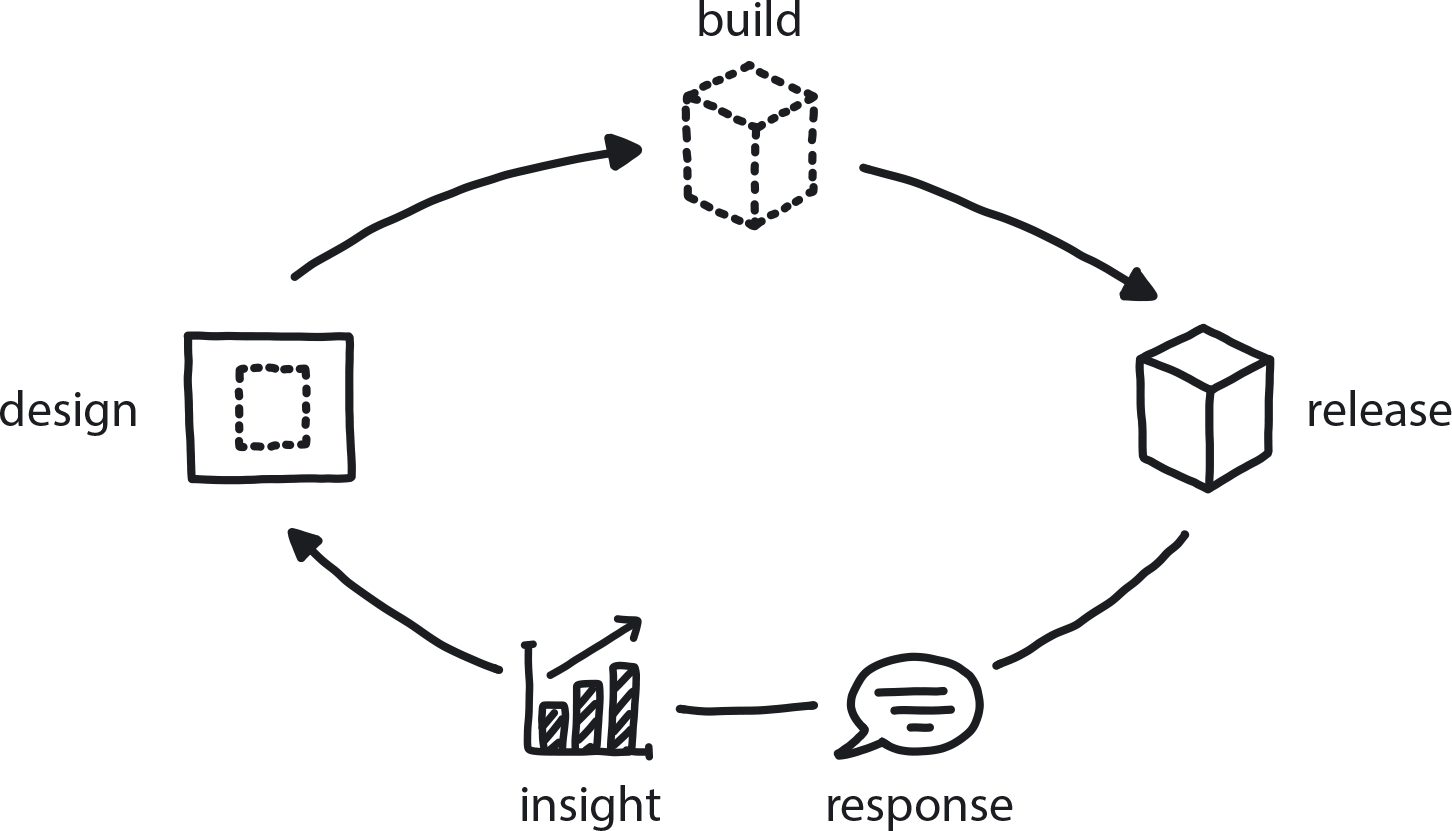
Assuming that others who are driving a change are doing so from hostile intent, trying to make the manager’s position ineffective or irrelevant.

**Related Patterns**

* [A/B Testing](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch09.html#a_solidus_b_testing)
* [Learning Loop](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_loop)
* [Measure What Matters](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#measure_what_matters)

**Pattern: Learning Loop**

*Building feedback collection into the delivery process closes the loop between engineers and the people who use their products, putting the customer at the center of the product development cycle (*[*Figure 7-11*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_loop01)*).*



**Figure 7-11. Learning Loop**

A company is working to transform its culture to align with the delivery approach required by cloud native’s distributed architecture of containerized microservices delivered via the cloud.

**In This Context**

**Learning happens in a three-part cycle: goal-setting, execution, and reflection. The first stage is identifying a challenge or problem and devising a likely solution. The second stage is carrying out the plan until it succeeds or fails. The third is studying the result—thinking back over what happened and how it worked out. In a very long delivery process this cycle is of limited use when lessons learned can be applied only months later, when the information is not fresh in the developer’s minds or perhaps no longer relevant.**

* The learning loop is useful in an organizational context only when it is closed (one stage leads directly to the next and the next, then the cycle repeats).
* Most companies adopting cloud native cite increasing product velocity as their main motive.
* Tightly coupled Waterfall-approach release cycles may be as long as six months to one year, or even two.

**Therefore**

**Build mechanisms for collecting user feedback and feeding it rapidly back into the delivery cycle, enabling responses to flow back from the customer so the business can make better-informed decisions.**

In cloud native, organizations can make extremely effective use of this three-part learning cycle as a feedback loop to developers that allows for fast iteration and adaptation to changing market conditions and customer needs. When a team is building its piece of an application and deploying it quickly and frequently to production, the team’s work gets in to users right away. If client feedback is collected at that point, it can be fed directly into the reflection stage for the team to take into consideration as the loop repeats and the team sets goals for its next work cycle.

* Use the feedback to create the changes customers are telling you they want.
* Not collecting feedback decreases the value of cloud native’s shorter time to market.

**Consequently**

**Apply Data-Driven Decision Making to cloud native’s rapid delivery cycle so that the output of the system continually goes back in to improve the system (you *can* go fast without breaking things).**

* + Developers capture the momentum of improvements, leading to more improvements.
* − Observability is complicated and needs to be carefully engineered to provide the right insights.
* − What customers want is not always feasible or cost-effective.

**Common Pitfalls**

Collecting feedback but not making it available or using it to gain meaningful insight. Engineers need to have observability into the outcome of their work so they can feed that knowledge into future work.

**Related Biases**

*Planning fallacy*

Make sure that the scope of the changes/improvements indicated by feedback actually fits into the delivery cycle.

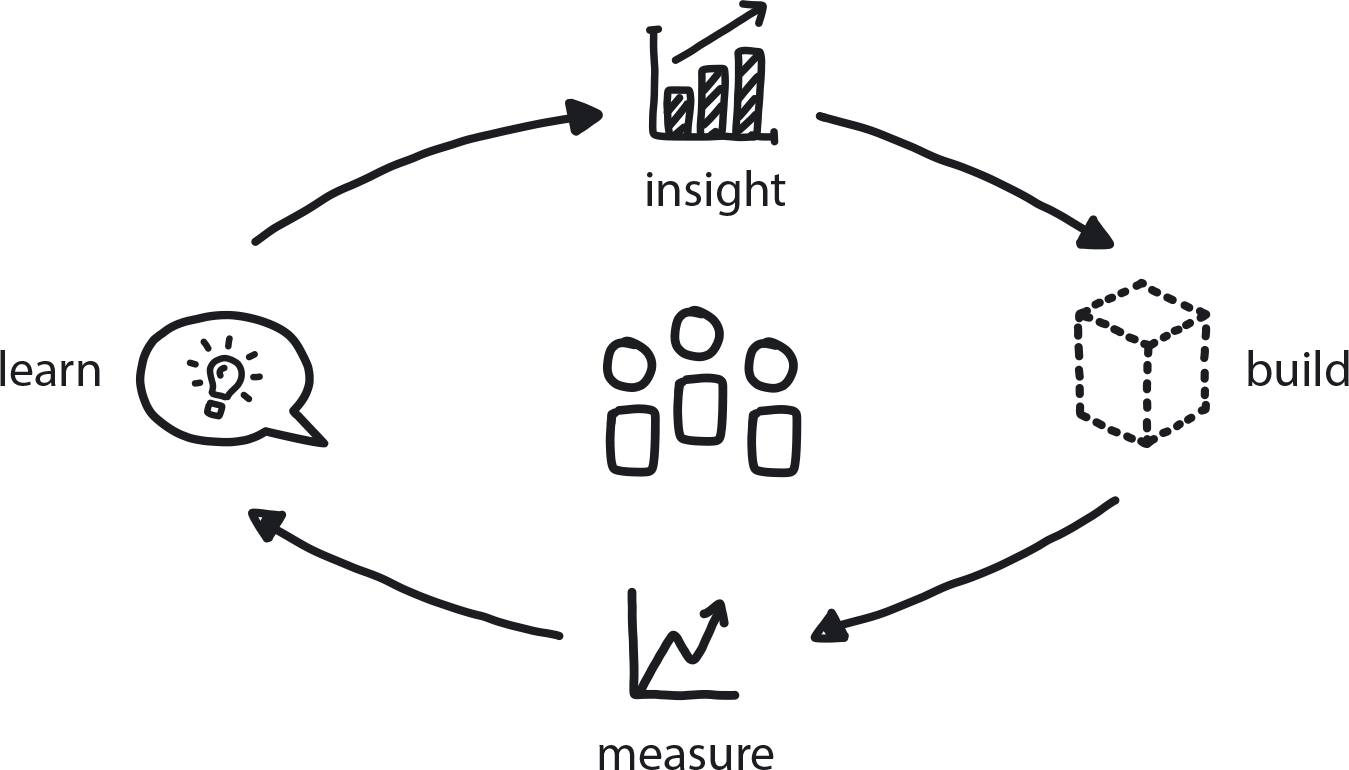
**Related Patterns**

* [Observability](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch10.html#observability)
* [Measure What Matters](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#measure_what_matters)
* [Reflective Breaks](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reflective_breaks)
* [Continuous Delivery](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch10.html#continuous_delivery)
* [Continuous Deployment](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch10.html#continuous_deployment)
* [Data-Driven Decision Making](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#data_driven_decision_making)

Events don’t occur in a vacuum. Effective decisions require not just situational awareness of events but also comprehending their context in order to interpret them and define the best response. Applying the Learning Organization pattern approach in your culture helps create and maintain a high level of understanding throughout the organization.

**Pattern: Learning Organization**

*An organization skilled at acquiring information, creating insight, and transferring knowledge can tolerate risk with confidence and solve difficult problems through experimentation and innovation (*[*Figure 7-12*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_organization01)*).*



**Figure 7-12. Learning Organization**

A company is launching a cloud native transformation and needs to create a new culture in the organization that supports innovation and accepts uncertainty. In the new and complex world of cloud native technology, there is no “right” way to go about a cloud native transformation.

**In This Context**

**Organizations migrating from Waterfall or Agile paradigms to cloud native don’t typically have the skill set for working in a highly uncertain and ambiguous environment: open-mindedness, a willingness to experiment and tolerate risk, and above all the ability to enter into the transformation process without a detailed map.**

Traditionally, before starting large projects managers would require all the answers and full work estimations. Then once the project was approved there would be no real possibility to adjust course. Such bureaucratic habits reinforce stability and reduce risks by eliminating change, but they conflict with the need to innovate and lead to re-creating the same old monolithic systems and organization, only now with newer tools.

* Teams are coming in with proficient, rather than creative, skills.
* Innovation requires the ability to tolerate risk and ambiguity.
* The process of innovation depends on failures and improvements over time.
* Few people are trained to think or work in a creative, open-ended way.

**Therefore**

**Take an honest look at your current culture. Build in the willingness to accept ambiguity and risk as part of your daily organizational process.**

Rather than demanding a full detailed plan with clear estimation upfront, embrace Dynamic Strategy and help teams to structure effective experiments. Make sure there is enough Psychological Safety to allow people to take risks.

* Leaders need to lead by example and show everyone that it’s safe to try new things even if they fail.
* Consider your company’s current relationship to risk and change. Does experimentation require permission? Is failure considered automatically “bad”?
* Treat change as an opportunity rather than a cause for anxiety.

**Consequently**

**Teams are co-creating solutions and challenging each other with Productive Feedback as they experiment their way toward the right answers.**

* + People learn how to act autonomously while being willing to fail—and then mine what went wrong so they can learn from it.
* − Training for cognitive resilience and creativity is possible, but it requires investment and work.

**Common Pitfalls**

Expecting teams to learn to work innovatively, experiment, and fail willingly, etc., while trying to use a proficient management approach that demands steady results.

Long-term precise and predictive planning is the norm in most enterprises, and it is also the biggest obstacle to meaningful change.

**Related Biases**

*Law of the instrument*

An overreliance on a familiar tool or methods, ignoring or undervaluing alternative approaches. “If all you have is a hammer, everything looks like a nail.”

*Shared information bias*

Group members are likely to spend more time and energy discussing information that all members are already familiar with (i.e., shared information). For example, the whole team did Docker training, so they spend a lot of time talking about Docker—and no time at all about Kubernetes, which is equally necessary but they don’t know much about it. To counter this bias, especially in new and complex environments, teams should learn new things all the time.

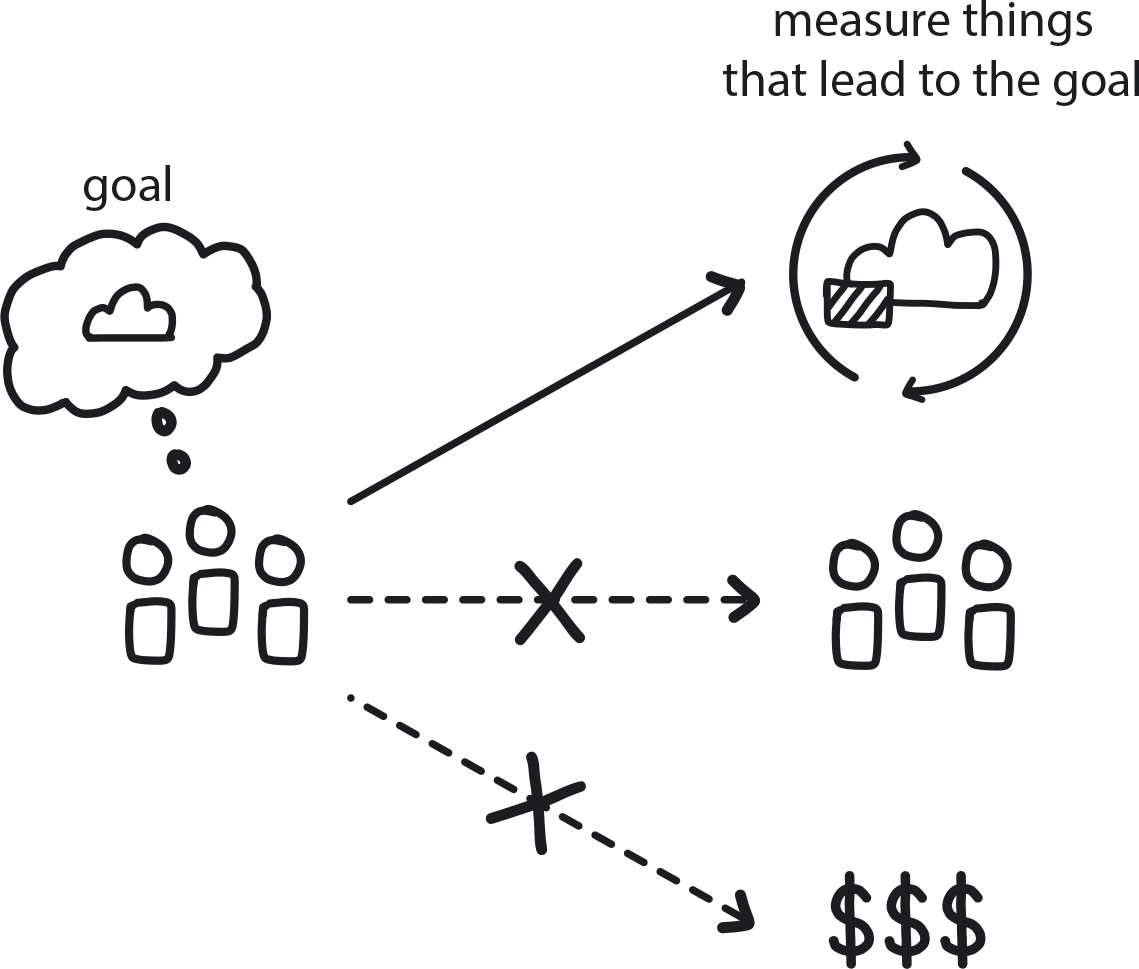
**Related Patterns**

* [Psychological Safety](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#psychological_safety)
* [Personal Relationships for Co-Creation](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#personalized_relationships_for_co_creat)
* [Manage for Creativity](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_creativity)
* [Manage for Proficiency](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_proficiency)
* [Blameless Inquiry](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#blameless_inquiry)
* [Learning Loop](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_loop)
* [Productive Feedback](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#productive_feedback)

Of course, in order to understand what’s really happening and the best way to respond, you need to be looking at the right things.

**Pattern: Measure What Matters**

*People optimize their actions based on how their work is measured. Assessing the wrong things leads people to optimize for the wrong goals (*[*Figure 7-13*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#measure_what_matters01)*).*



**Figure 7-13. Measure What Matters**

The company has strategic and tactical goals, and the management team wants to establish company performance metrics to measure and monitor progress.

**In This Context**

**People tend to optimize their work output based on what is measured. Incorrect measurements will result in flawed deliveries (delivering the wrong things) and suboptimal performance.**

For example, measuring velocity in Scrum will lead to inflation of story points. If the key performance indicator (KPI) doesn’t reflect the company’s actual needs, the result will be poor. This is an example of Goodhart’s law, which states “When a measure becomes a target, it ceases to be a good measure.” What happens is that which this can occur is individuals trying to anticipate the effect of a policy and then taking actions that alter its outcome.

* People tend to optimize for best measurements.
* Most methodologies bring their own metrics.
* Different teams may have different goals.

**Therefore**

**Always adjust performance measurements to fit the organization’s strategic and tactical needs. Keep measuring the most important KPI and stop when specific behavior becomes routine. Only measure a few KPIs at a time, choosing ones related to the current worst bottlenecks. Prioritizing customer value as the main metric helps to focus on customer needs.**

* Keep KPIs measurable and achievable but also stretched for delivery progress.
* Measure differently for creativity versus proficiency.
* Avoid getting stuck in vanity metrics like number of site visitors and focus on measuring customer value.

**Consequently**

**Managers set up KPIs in conjunction with goals and adjust them as the goals are changing.**

* + Higher awareness from all involved.
* + All stakeholders have more clarity about priorities and goals.
* − It is difficult—and confusing to the teams—if metrics change frequently.

**Common Pitfalls**

Having too many measurements is a common pitfall. People can’t be effective when they try to achieve too many simultaneous goals. Although a company may have a variety of strategic and tactical goals, it must choose only the most critical ones—or risk achieving none at all.

**Related Biases**

*Information bias*

The tendency to seek information even when it cannot affect the outcome; in A/B testing this would lead to choosing meaningless variables to test.

*Confirmation bias*

Picking test variables that will “prove” your existing opinion, rather than genuinely exploring alternatives.

*Congruence bias*

If you have a preconceived outcome in mind and test results that confirm this, you stop testing instead of seeking further information.

*Parkinson’s law of triviality/”Bikeshedding.”*

Choosing something easy but unimportant to test and evaluate instead of something that is complex and difficult but meaningful.

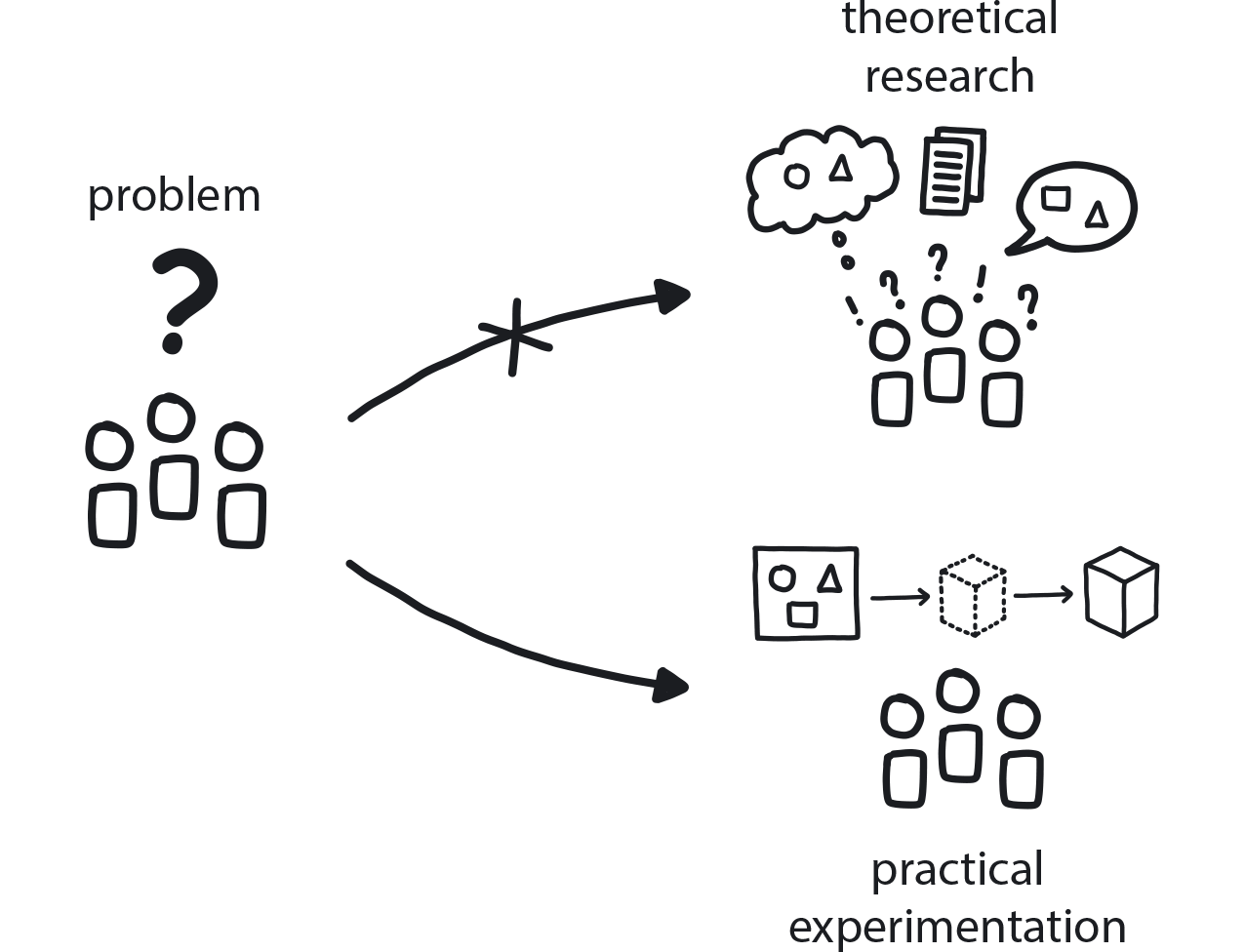
**Related Patterns**

* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Involve the Business](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#involve_the_business)
* [A/B Testing](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch09.html#a_solidus_b_testing)
* [Manage for Creativity](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_creativity)
* [Manage for Proficiency](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_proficiency)
* [Learning Loop](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_loop)

It’s a mistake to think of measurement as merely the passive collection of data. It’s the same with research. Figuring things out through trial and error keeps teams engaged and builds skills by doing.

**Pattern: Research Through Action**

*People can sometimes use research as a way to avoid making decisions, so hands-on learning through small experiments builds confidence and jumpstarts progress (*[*Figure 7-14*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#research_through_action01)*).*



**Figure 7-14. Research Through Action**

The challenges are new and complex, information is scarce, and the company is pushing to go fast.

**In This Context**

**In a new or unfamiliar environment, people can analyze things too much and fail to make progress: analysis paralysis.**

The tendency is to spend a lot of time on research, because making an actual decision is daunting—particularly in an environment where failure historically resulted in punishment. The effort to take in so much new information can also be overwhelming. As a result, people will often skip from one idea to the next before getting enough understanding to guide informed decision making. All of this adds up to procrastination because it is not effective data gathering related to actually moving ahead with a plan—it is research for research’s sake.

* When people are not allowed to fail, they avoid taking risks.
* Quick actions carry a lot of risk.
* If the cost of action is high, people tend to delay and wait for more info.
* In a world with many dependencies, taking action requires involving others.

**Therefore**

**Run small experiments instead of full analysis and research; choose action over extensive contemplation and exhaustive research.**

We all find ourselves from time to time in front of a massive pile of work without the faintest idea where to start. Doing nothing is the easiest choice, of course, but this won’t lead very far. So then we try some thorough planning to “make sense” of all these tasks, doing lots of reading and Googling. In an uncertain environment, however, we won’t be much smarter after all that work—the best course of action is to simply pick the first task from the pile and do it! Then another one and another one. Keep doing this until you gather enough information about what’s going on, and at that point a bit of planning could be appropriate.

* Experiments and PoCs instead of in-depth architectural documents.
* Limit the risk of action with short deadlines and low cost.

**Consequently**

**You are making minor yet tangible progress through taking small, iterative steps.**

* + Uncover unknown-unknowns through experimentation.
* + Increase team motivation and joint learning.
* + Many experiments fail—this is OK.
* + A solution that is not fully baked can still be a valid choice.

**Related Biases**

*Ambiguity effect*

The tendency to avoid options where information is missing or difficult to gather.

*Information bias*

The tendency to seek information even when it cannot affect action. “Analysis paralysis” is very common—continuing to find more answers for more questions, even though there are only two or three possible actions to take and the benefits of one of them are very clear.

*Parkinson’s law of triviality/”Bikeshedding.”*

Choosing something easy but unimportant to test and evaluate instead of something that is complex and difficult but meaningful.

**Related Patterns**

* [Exploratory Experiments](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#exploratory_experiments)
* [PoCs](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#proof_of_concept)
* [Avoid Reinventing the Wheel](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch09.html#avoid_reinventing_the_wheel)
* [Reduce Cost of Experimentation](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reduce_cost_of_experimentation)
* [Gradually Raising the Stakes](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes)

**Practical Patterns for Managing Any Kind of Risk**

Whenever you need to create strategy around any big change for your organization because there is no clear next step, much less an obvious path to success, Gradually Raising the Stakes is the pattern that will take you there. It gives us the tools to address uncertainty and gradually reduce risk by moving step by logical next step. The way it works is through three graduated levels of risk taking: little risk, moderate risk, high risk. In terms of pure strategy formation we are guided through the process by unpacking Gradually Raising the Stakes’ related next-steps patterns: No Regret Moves, Options and Hedges, and Big Bet.[**2**](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#ch07fn2)

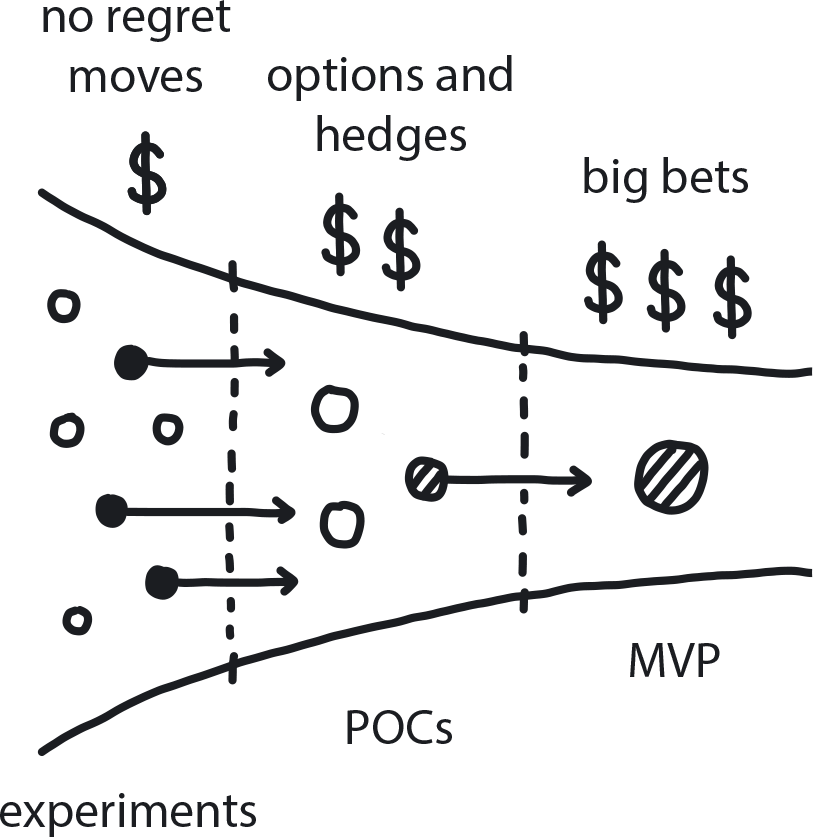
When it comes to setting the right technical path in a transformation design, however, there is a different set of sub-patterns correlating with this three incremental stages de-risking approach: Exploratory Experiments, PoCs (Proof of Concepts), and MVP Platform. You can find these in [Chapter 9](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch09.html#patterns_for_development_and_process).

Low-certainty projects carry extremely high risk. Any decision you make early in the project is highly likely to be uncovered later as wrong. In a predictable environment like Waterfall, you can start a big project without doing experiments/PoCs because, although the project itself is new, it is still very similar to previous projects. In an uncertain environment where the next move is not clear (and the one after that a complete mystery) this strategy may lead you to make major technical and organizational decisions too early—with little chance of reversing them later, due to the sunk-cost fallacy. This often leads to a wrong solution remaining in place—sticking with the original choice because so much is already invested—and possibly the punishment of key people involved in choosing the solution.

So how *do* you undertake a major new initiative—and avoid making major and costly mistakes—when you have no previous knowledge or experience in this area? Patterns help us move through uncertain situations, and Gradually Raising the Stakes opens the door to this process.

**Pattern: Gradually Raising the Stakes**

*In an uncertain environment, slowly increase investment in learning and information-gathering; eventually you uncover enough information to reduce risk and make better-informed decisions (*[*Figure 7-15*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes01)*).*



**Figure 7-15. Gradually Raising the Stakes**

You are in the beginning of a transformation, and the executive team is committed to moving forward but needs to map how it will proceed. The company is new to cloud native, and the team has little knowledge or experience in the field.

**In This Context**

**Making major decisions before having enough information to understand the parameters carries a great deal of risk. However, in the uncertain environment of an early cloud native transformation when there is not yet much knowledge or a clear path, grabbing right away for a “big bang” solution is very tempting.**

* In traditional environments budgets are allocated based on full estimation of scope and a clear execution plan.
* In the cloud native environment high uncertainty makes estimation difficult; next steps are not predefined but uncovered one at a time.
* Executives still expect problems to have a well-defined and reasonably predictable solution, because that is how they have always worked before.

**Therefore**

**Avoid making big decisions early; do a series of small projects, always growing slowly in size, until you have enough information to make a big bet.**

Begin the initiative with small no-risk actions that benefit the organization in any circumstances (see: *No Regret Moves*). Once a baseline understanding is in place, move to actions that help deepen understanding in areas that seem especially relevant to the organization’s transformation goals, and begin narrowing down the field of options (see: Options and Hedges). Eventually a reasonably clear best path will emerge, and the company can then make a reasonably confident Big Bet commitment. This, in the cloud native context, means building an MVP version of the most likely large-scale solution.

* Move incrementally through three stages: first, learn the basics. Then deepen knowledge through more detailed investigation and experimentation, which helps eliminate wrong choices. Finally, gather the information revealed and use it to decide the best likely path.
* Once the big bet is taken, stay on the path until the circumstances are stable.
* If circumstances change due to new information uncovered by experiments, or if the market changes, then remember to refactor the strategy using Dynamic Strategy.

**Consequently**

**The project has been gradually refined/decided without taking disproportionately high risks, and appropriate budget and resources have been allocated to each stage based on its level of uncertainty.**

* + You know how much resource each project will require as you uncover next steps; you are allocating just that, no more, no less.
* + The level of detail of the project, and the understanding of the scope and the context, gradually increases with each step—which proportionally lowers the embedded risk.
* − There is great ambiguity in the project, which prevents thorough financial and resource planning. Ambiguity typically feeds anxiety.

**Common Pitfalls**

Making big decisions before having all the information—committing to a solution without doing enough experimenting to uncover risks (or simply not experimenting at all); conversely, endlessly experimenting with everything to minimize risk never quite reaching a decision or committing to a plan.

**Related Biases**

*Irrational escalation/Sunk-cost fallacy*

The urge to keep investing in a project, despite new evidence that the decision was probably wrong, because you’ve already sunk so much into it.

*Confirmation bias*

Picking test variables that will “prove” your existing opinion, rather than genuinely exploring alternatives.

*Ambiguity effect*

Cloud native is complex and the benefits are not easily visible, so better just to keep things the way they are.

*Zero-risk bias*

Preference for reducing a small risk to zero over a greater reduction in a larger risk. People keep experimenting thinking they can eventually get something perfect, and so never stop doing research and experimentation because there is always a better solution if you wait long enough and keep looking.

**Related Patterns**

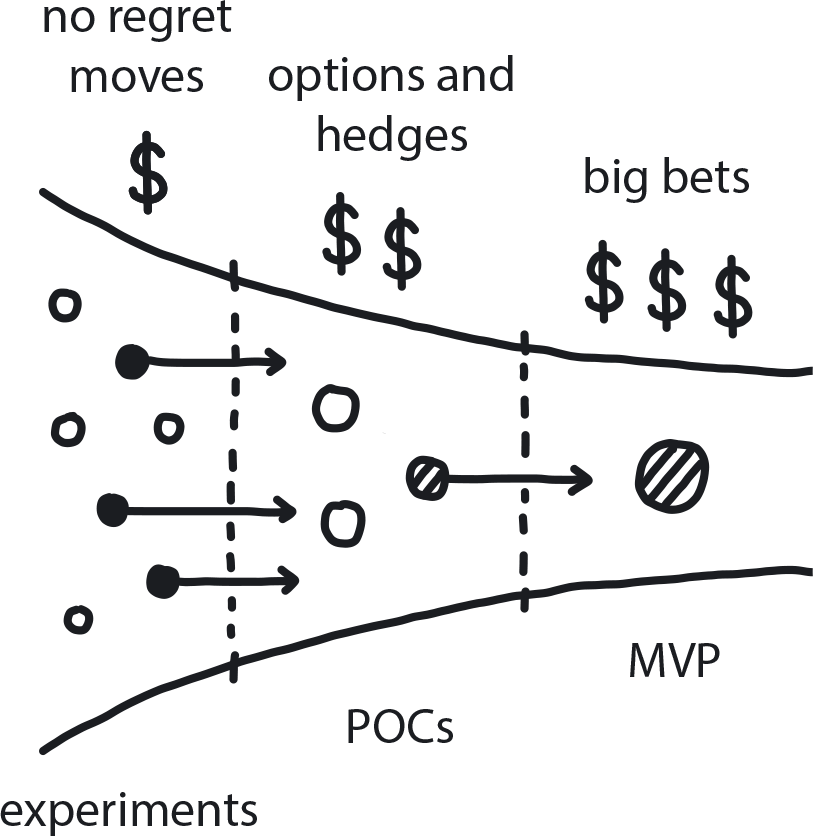
* [No Regret Moves](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#no_regret_moves)
* [Options and Hedges](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#options_and_hedges)
* [Big Bet](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#big_bet)
* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)

Cloud native transformation is a massive project with a variety of risky moves embedded in it. Making a totally wrong technical or organizational turn early on might be much worse than just a small loss of time and money. In the past five years observing cloud native transformations, we’ve seen companies struggling in a transformation for two to three years without ever reaching satisfactory results. Instead, they stay stuck fighting with the wrong tools and an organization that doesn’t allow innovation.

In a situation with high embedded risk and little knowledge, how do you identify, classify, and investigate possible solutions in such an unknown environment? The No Regret Moves pattern show us the place to begin.

**Pattern: No Regret Moves**

*Small, quick actions that require little investment of time and money but increase knowledge, reduce risk, and benefit the entire organization—inside or outside of a transformation scenario (*[*Figure 7-16*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#no_regret_moves01)*).*



**Figure 7-16. No Regret Moves**

A company is in the beginning of a cloud native transformation initiative or facing any other difficult technical or organizational question with no obvious or readily available answer.

**In This Context**

**Lacking adequate information, the team has no practical way to make an educated decision—and essentially will have to gamble on a semi-random solution and hope for the best.**

Unfortunately for traditional Waterfall organizations, managers are measured based on their success in leading large initiatives. Thus, many ignore initial small exploratory steps in favor of jumping into a game-changing project that will earn them a fat bonus and a nice promotion. In Waterfall organizations, which typically operate with known technologies in reasonably stable markets, that strategy often works. But in a highly volatile cloud native environment, it can easily lead to disaster or, worse, wasted years without either clear success or failure.

* Change creates anxiety, so people tend to avoid it.
* In the early stages of a cloud native transformation there is little knowledge or experience within the organization around cloud native architecture/tech/culture.
* The situation is highly uncertain, and making big commitments early carries a lot of risk.

**Therefore**

**Take first-stage risk-reduction actions that are quick, low-cost, and benefit the company no matter what.**

Some improvements to operational effectiveness—including training and coaching, and, in a project context, running small experiments and technical exercises—would benefit any business in just about any circumstance. These small but beneficial and practically no-risk moves are especially valuable during the first, highly uncertain days of a cloud migration and form the first of the three stages of graduated transformation risk-management strategy.

* An in-depth quality assessment like the cloud native Maturity Matrix creates situational awareness that is universally valuable to the organization now, but also serves as a gap analysis for future moves.
* Trainings and educational offerings keep employees engaged and constantly refresh their knowledge.
* Hackathons and small experiments uncover technical answers via hands-on actions.

**Consequently**

**The organization has gained self-awareness and knowledge without investing huge amounts of time or money. Risk has been incrementally lowered, and the company’s leaders are ready to take the next step in setting the transformation path.**

* + When people know more, they feel more confident, which increases their ability to meet challenges creatively and proactively.
* − Since No Risk Moves are so affordable and beneficial, the company could stay stuck in this stage for too long and delay getting the transformation underway.

**Common Pitfalls**

Doing things like training and education for only some, but not all, teams. Likewise, having an assessment done but not sharing the results across the organization. The more people know, the more risk is reduced—so don’t limit information sharing to only certain groups

**Related Patterns**

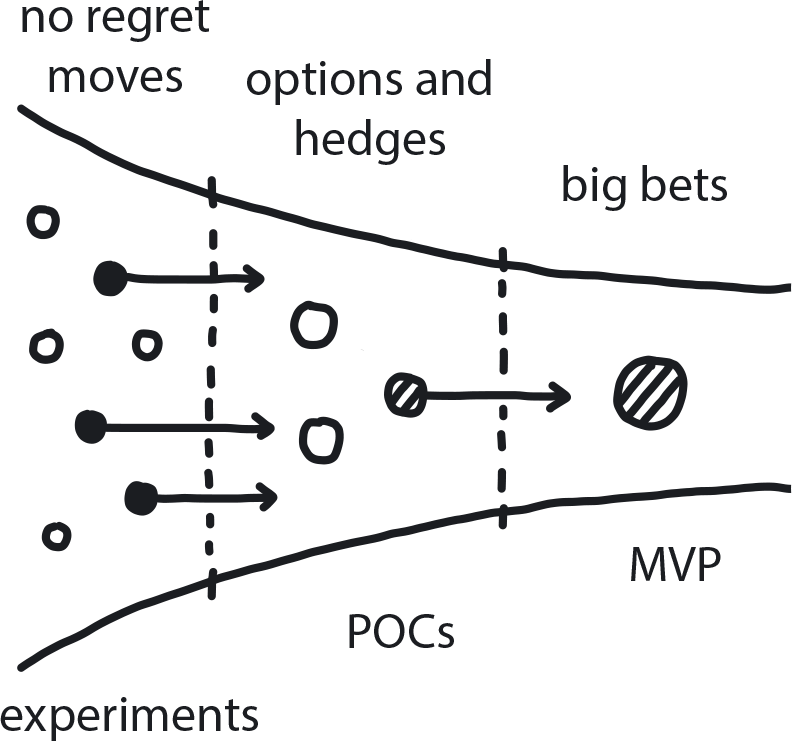
* [Gradually Raising the Stakes](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes)
* [Options and Hedges](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#options_and_hedges)
* [Big Bet](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#big_bet)
* [Learning Organization](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_organization)
* [Research Through Action](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#research_through_action)
* [Exploratory Experiments](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#exploratory_experiments)
* [Reduce Cost of Experiments](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reduce_cost_of_experimentation)

At the beginning of a transformation risk is high and knowledge is low, so it’s easy to just keep running experiments (especially when they are quick, low-cost, and generally beneficial No Regret Moves). Your research has given you a better understanding of what is going on, but there is still fairly high risk involved. Making a big decision at this point can still lead to unnecessarily high expense, while running additional tiny experiments that uncover no new information is just a waste of time.

Once some promising potential solutions have been identified, how do you further test their durability and fitness? The Options and Hedges pattern shows us the next logical step to follow No Regret Moves.

**Pattern: Options and Hedges**

*Research has created deeper understanding, and a few potentially promising transformation paths have begun to emerge. Continue reducing the risk by focusing on the most promising options and developing them further (*[*Figure 7-17*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#options_and_hedges01)*).*



**Figure 7-17. Options and Hedges**

You have achieved moderate certainty by running a series of small experiments, research projects, and trainings, but the team is not yet confident enough to make major decisions about the transformation.

**In This Context**

**Your research has given you a better understanding of what is going on, but major decisions are still not obvious. Commitment to a large solution at this point still carries serious high risk of choosing the wrong solution, while running additional tiny experiments that uncover no new information is just a waste of time.**

* Pressure is rising to make commitments.
* Too easy to keep running experiments forever.
* There will always be some uncertainty, no matter how much research you do.

**Therefore**

**Make small tactical decisions aimed at creating and understanding a new path forward. They can be rolled back or forward, ramped up or down, and will at least eliminate some options while you create new plans.**

The goal is to validate the results of any successful experiments performed so far.

* Do mid-size proof-of-concept projects that take a few weeks to a couple of months.
* Stay aware of biases like the IKEA effect and sunk-cost fallacy that could lead to sticking with a solution even when it’s not really working.
* No commitments yet! Maintaining the ability to change direction reasonably easily is still important here.
* Take a vendor-neutral approach: companies need the best tools and techniques for their cloud native platform, and these are not necessarily the ones offered in bundled solutions.

**Consequently**

**You have uncovered the majority of the important information required and are reasonably certain where you are going next.**

* + Risk declines because, as you experiment, you eliminate things that don’t work and reveal the ones that do.
* + Next steps come into stark relief. The results of all these little experiments actually shape the architecture of the new system.
* + You are ready to take the next big steps (Big Bet) because you have gained knowledge and experience.

**Common Pitfalls**

Jumping too soon by committing to a Big Bet move prematurely. Or the opposite, never jumping because you are doing endless research in the form of No Regret Moves.

**Related Biases**

*Information bias*

The tendency to seek information even when it cannot affect action. “Analysis paralysis” is common—continuing to find more answers for more questions, even though there are only two or three possible actions to take and the benefits of one of them are clear.

*Confirmation bias*

Picking test variables that will “prove” your existing opinion, rather than genuinely exploring alternatives.

*Congruence bias*

The tendency to test hypotheses exclusively through direct single testing, instead of testing possible alternative hypotheses.

*Ambiguity effect*

The tendency to avoid options for which missing information makes the probability of the outcome seem “unknown.”

**Related Patterns**

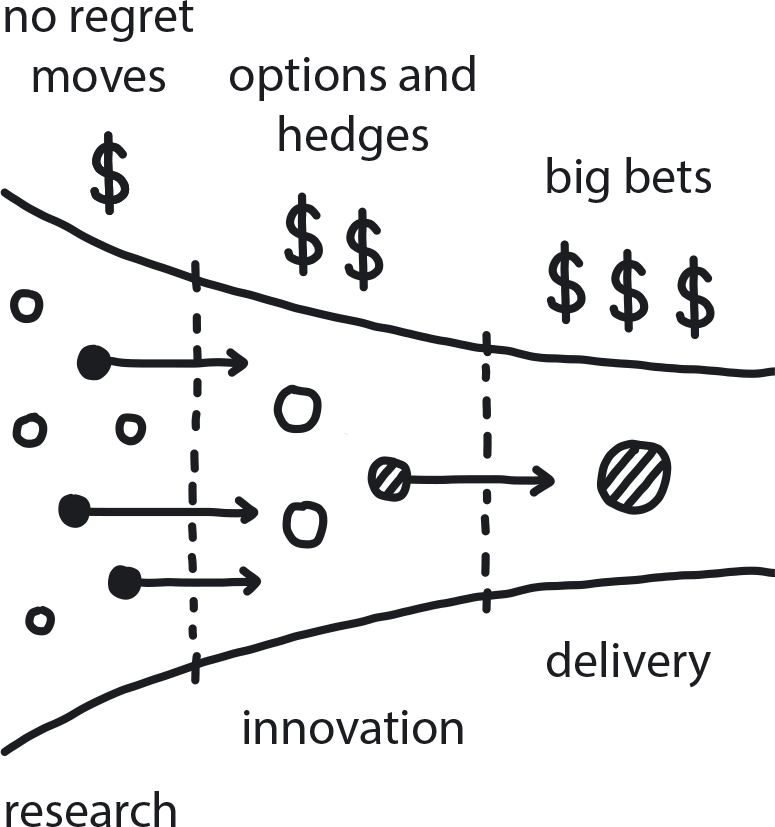
* [Gradually Raising the Stakes](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes)
* [PoCs](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#proof_of_concept)
* [MVP](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#mvp_platform)
* [Big Bet](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#big_bet)
* [No Regret Moves](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#no_regret_moves)

When decisions are made too early, before enough information was uncovered by experiments and research, it’s easy to commit to a solution that turns out to be less than ideal. This means risking reversal at high cost in order to change to a better one—or sticking with a wrong solution anyway due to sunk costs.

Conversely, sometimes the team is hesitant to make a big decision. This often leads to wasting time and resources on research that adds only marginally useful information or, worse, on parallel investment into multiple solutions for the same problem. The answer to both these conundrums is to make the Big Bet, but at the right time. But how do you know when it’s the right time to finally choose a single solution, even if there is still some uncertainty—when is the right time to finally end the research? How do you convince the organization that the approach you are outlining can solve real problems at real scale? The next step after Options and Hedges, the Big Bet pattern, shows us how.

**Pattern: Big Bet**

*When enough information is available, commit to a significant solution for moving the cloud migration forward. Focus on execution rather than research (*[*Figure 7-18*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#big_bet01)*).*



**Figure 7-18. Big Bet**

A company is facing a big technical or organizational decision. Experiments were performed, research done, major points validated, and the team has a good understanding of the company’s needs and the problem domain. Multiple major directions are still open.

**In This Context**

**Continuing research and experimentation without ever making any big decision leads to significant waste of resources as the teams are not focused on solving the problem and the direction is not chosen yet. It means that there is no clear alignment across teams regarding a solution, and no stable and focused delivery process has been established.**

* In Waterfall, teams tend to do too much research and try to find all the answers for all questions.
* Many times any decision is better than none.
* Large corporations hate risk and therefore push for more research.
* In Waterfall organizations, managers’ success is measured by their ability to decide on and execute major changes.

**Therefore**

**Make a commitment to a large-scale solution, like a large rebuild, architectural change, migration, purchase of new products, etc., bearing in mind that it might require organizational change.**

After exploring the options with No Regret Moves and increasing the chance of success even more with Options and Hedges, we now can make a big commitment toward the right longer-term solution. Once the commitment is made, the teams switch from research to execution mode, provided there are no significant changes in the market or other game-changing information. This creates alignment among teams and allows quick product delivery, without endless discussions about the direction.

* Make the commitment clear to the team.
* Stop doing competing projects.
* Consider an exit strategy (evaluate related costs should it be necessary to reverse the decision).

**Consequently**

**There is full commitment to the chosen direction. It is clear to everyone that this is a commitment moment: at this time we stop experimenting and move forward. Unless there is a significant change in market or strategy conditions, teams stay committed to the chosen path.**

* + A single solution is chosen, and the team is focused on making it work.
* + No effort is wasted on work that doesn’t improve the decision process.
* − There is always a chance that the decision can turn out to not be the right one.

**Common Pitfalls**

Almost always: people make the Big Bet too early. In some cases the decision proves to be successful, which reinforces the confidence of the managers even if that success was a result of pure luck. Other decisions are not so good, and in those cases, managers will work hard to either hide the failure or present it as a success. Sunk-cost fallacy is a very strong factor in such situations.

Forgetting to get back to experimenting/No Regret Moves when conditions shift and change is in order.

**Related Biases**

*Irrational escalation/Sunk-cost fallacy*

The urge to keep investing in a project, despite new evidence that the decision was probably wrong, because you’ve already sunk so much into it.

*Zero-risk bias*

Preference for reducing a small risk to zero over a greater reduction in a larger risk.

*Status quo*

Once you commit to something, you stop looking around—but you still need to keep situational awareness to be able to adjust to market/environment changes with Dynamic Strategy.

**Related Patterns**

* [Gradually Raising the Stakes](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#gradually_raising_the_stakes)
* [No Regret Moves](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#no_regret_moves)
* [Options and Hedges](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#options_and_hedges)
* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Three Horizons](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#three_horizons_pattern)
* [Exit Strategy Over Vendor Lock-in](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#exit_strategy_over_vendor_lock_in)

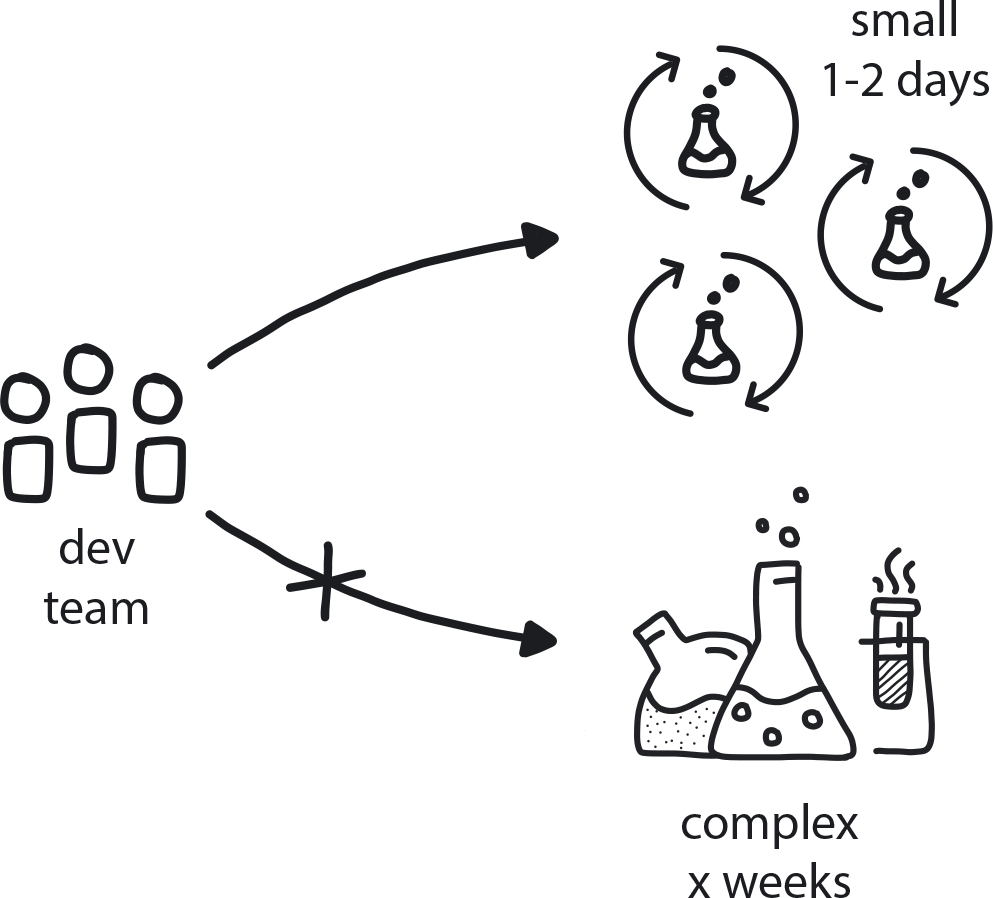
Cloud native processes are iterative; they incorporate rapid feedback and data from a change to then make more changes. Thus these patterns operate as loops, and the essential process can be summed up as “investigate, iterate, incorporate, repeat.” That methodology is at the heart of the Gradually Raising the Stakes pattern: No Regret Moves is about investigating many possible solutions/paths to narrow down to the most promising ones. Options and Hedges then takes a deeper look at the best options, iterating them forward to see how they hold up. This provides more data to make a best-fit Big Bet decision. This set of patterns is shown here as a way to de-risk major situations like a transformation initiative, but the loop is equally applicable to smaller scale and more minor decisions as well.

Since experimentation is the main tool for working our way through the incremental stages of curating solutions through investigation and elimination—no matter how large or small the problem at hand—it is imperative that your company removes any barriers that prevent or bottlenecks that slow these small but impactful investigations.

Running an experiment in a traditional organization is difficult. Trying something new in a complex monolithic software system requires extensive planning and complex setup to order and install servers running the experiment: it may be months before you get results. If each experiment costs $100,000 and takes several months, not to mention a lot of meetings and paperwork to get permission in the first place, then you won’t run many. In most cases the team will simply skip experimentation altogether and move directly to full execution, ignoring the massive risks that this approach entails.

**Pattern: Reduce Cost of Experimentation**

*When someone has an idea that requires validation, the cost of doing experiments around it needs to be as low as possible (*[*Figure 7-19*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#Reduce_Cost_of_Experimentation01)*).*



**Figure 7-19. Reduce Cost of Experimentation**

A company is moving into cloud native with little understanding of tooling and technologies. Uncertainty is very high. New ideas and solutions come up all the time, and the team has to run a variety of validation experiments to select the best ones.

**In This Context**

**There are significant barriers to experimentation in the organization: permission is required, and the related planning, documentation, and coordination meetings take a lot of time. Then actually getting the results afterward typically requires a significant wait. As a result, engineers will often skip experimentation and move directly to execution.**

* Very little chance that any experiment will lead to canceling a project because there is already high commitment to that path.
* Higher cost means an experiment is less likely to be done.
* If cost is low enough, people can experiment even with things that lead to failure.
* The higher the cost, the lower the willingness to abandon the experiment (sunk-cost effect).
* Traditional organizations require extensive documentation with any experimentation, which by definition makes costs high.

**Therefore**

**Put in place a simple, straightforward, and seamless process for doing experiments. When experimentation is central to an organization’s process and progress, it needs to be an inexpensive and easily accessible action.**

Outline and publicize a process, or provide periodic training. for how to create appropriate hypotheses with measures and then design small experiments to test them. Aim to remove bureaucratic roadblocks such as managerial approvals and extensive documentation for every action, while providing technical infrastructure that is light, fully automated, and requires only a tiny budget.

* Create a framework around experimentation: the tools, facilitation techniques, project-management structure, and guidelines for allocation.
* Such facilitation makes it faster to obtain and understand the results of an experiment.

**Consequently**

**More experiments take place. Instead of extensive research and guessing when a complex problem arises, a rapid process of hypothesis/results/analysis provides the solution.**

* + Teams can be more innovative when they know they can easily try out ideas.
* − There is cost to experiments, even if it is low.

**Common Pitfalls**

Potential for over-experimentation—people love experiments. If they are too cheap and easy, people will do nothing but experiment, and nothing ever moves forward (analysis paralysis).

Trying for a “sure thing”—If experiments require a long process of approval and documentation, then they typically become less like true explorations to test hypotheses and gather conclusions and more like PoCs to reassure that the proposed plan is working as described or intended.

**Related Biases**

*Irrational escalation/Sunk-cost fallacy*

The urge to keep investing in a project despite new evidence that the decision was probably wrong, because you’ve already sunk so much into it.

*Information bias*

The tendency to seek information even when it cannot affect action. Also known as “analysis paralysis,” this is very common—continuing to find more answers to more questions, even though there are only two or three possible actions to take and the benefits of one of them are clear.

*Parkinson’s law of triviality*

Choosing something easy but unimportant to test and evaluate, instead of something that is complex and difficult but meaningful.

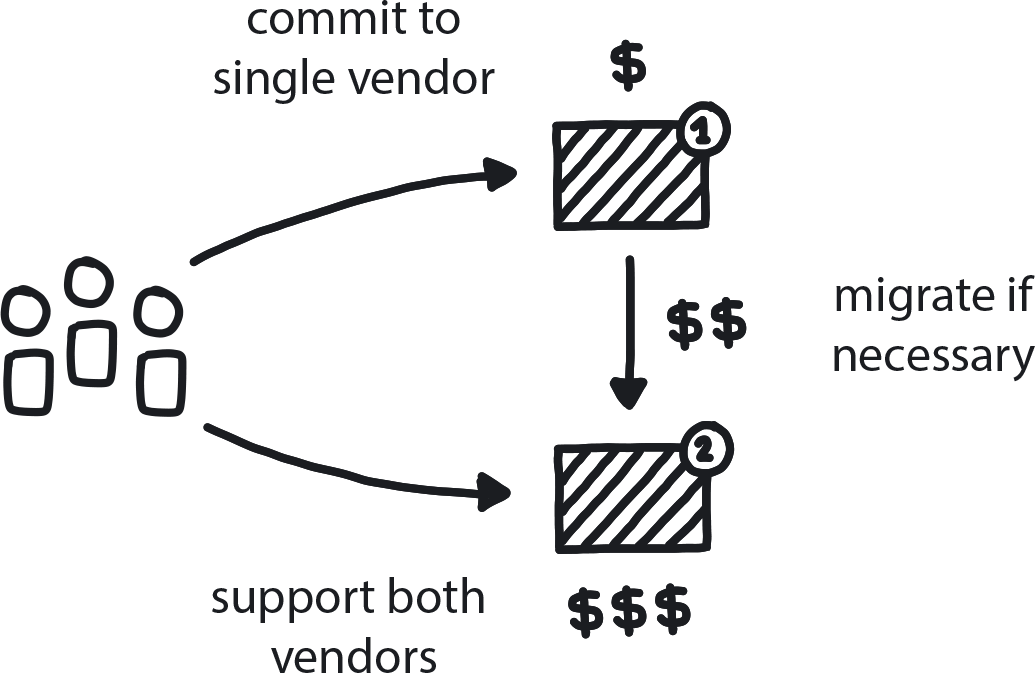
**Related Patterns**

* [Exploratory Experiments](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#exploratory_experiments)
* [PoCs](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#proof_of_concept)
* [Measure What Matters](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#measure_what_matters)
* [Psychological Safety](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#psychological_safety)
* [Productive Feedback](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#productive_feedback)
* [Manage for Creativity](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_creativity)
* [Blameless Inquiry](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#blameless_inquiry)

Even as the company is working to identify the best transformation path, it’s important to keep an eye out for second-best options too. This information comes in handy when evaluating backup plans, and even more so for deciding whether to commit to a particularly global set of solutions.

**Pattern: Exit Strategy Over Vendor Lock-in**

*Public cloud or other major product vendors can handle all aspects of building and operating a cloud native platform, and their tools are often excellent—but when committing to a vendor/technology/platform, it’s important to identify an alternate solution and any costs associated with switching over (*[*Figure 7-20*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#exit_strategy_over_vendor_lock)*).*



**Figure 7-20. Exit Strategy Over Vendor Lock-in**

Companies need to use tools/products provided by vendors (commercial or open source communities) but are afraid to get locked into a single vendor. Some industries require supporting multiple choices for some key platforms, tools, or technologies used to develop company products. Being fully committed to a single tool or technology can drive costs up or hinder tech progress down the line.

**In This Context**

**Committing to a single vendor (or simply a single large solution) creates reliance upon their ongoing stability and availability and pricing, but the cost of maintaining active alternative/backup options is prohibitive.**

This leads to full dependency on the vendor. If the vendor is in trouble or when a better option becomes available on the market, the company cannot afford to switch—which leads to using unmaintained or inferior technology.

* There is risk in any choice.
* Every decision can be reversed; it’s just a matter of cost.
* Some industries require multiple vendors for each solution.

**Therefore**

**Instead of blindly refusing to commit to a single vendor, explore the options for a second migration, if necessary, and what they would cost. Then make an educated decision based on the tradeoff between short-term gains from a vendor with the best tool and the long-term risk of migrating out of it if needed. Often lower costs and higher productivity outweigh the risk.**

Consider implementing architectural changes that will reduce the cost of migration in case it will be required. Such changes can be inexpensive if done in the earliest stages of the project, and they can significantly reduce the risk of the future migration.

* Prepare a migration plan just in case.
* Consider reducing dependencies that significantly increase risk.
* AWS could be a big help now.
* Closed ecosystems often offer an excellent set of tools/options.
* Invest in commonly used tools and technologies that may become industry standards.

**Consequently**

**The team can focus on getting the maximum performance out of and benefit from each tool, and they are aware of what it would cost to migrate to alternative solutions should the need arise.**

* + There is a good understanding of the different backup options.
* + High-level plan is in place for a major migration scenario.
* − There is always risk when only one tool is used.

**Common Pitfalls**

Sub-prime solutions: A team may waste effort seeking, evaluating, or implementing solutions that will work with many different tools in order to optimize portability in the event of emergency. In many cases the resulting solution is far from optimal for the company’s actual needs, due to the need to get down to the lowest common denominator across chosen tools.

**Related Biases**

*Zero-risk bias*

Preference for reducing a small risk to zero over a greater reduction in a larger risk; in this case, building/buying an expensive backup solution in case of the unlikely need to suddenly and completely migrate between options.

**Related Patterns**

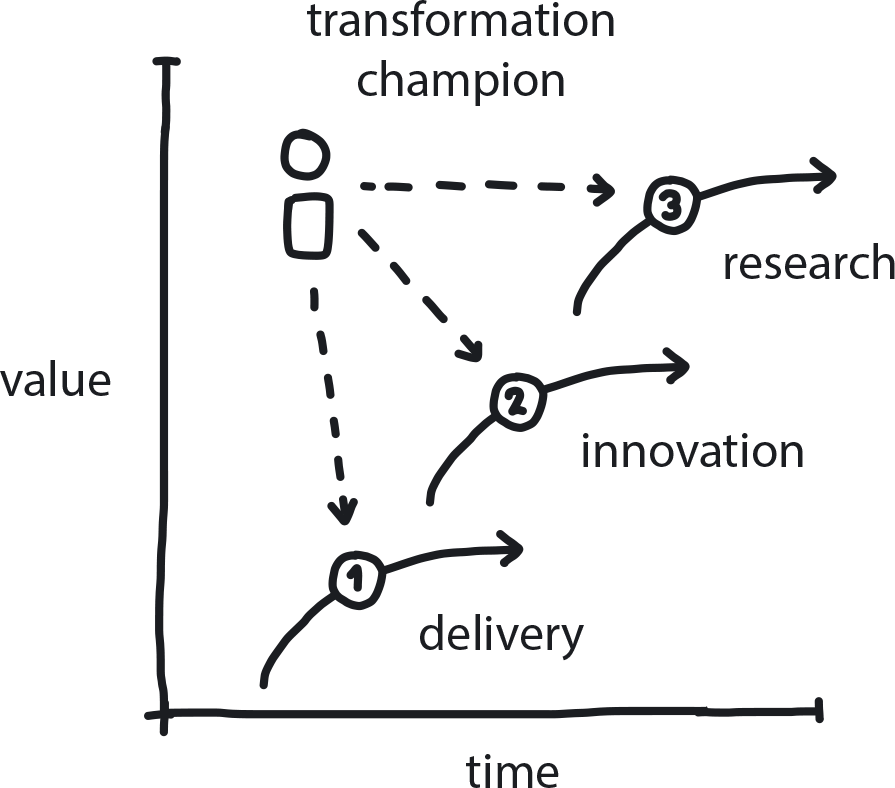
* [Public Cloud](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch10.html#public_cloud)
* [Private Cloud](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch10.html#private_cloud)
* [Avoid Reinventing the Wheel](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch09.html#avoid_reinventing_the_wheel)
* [Open Source Internal Projects](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch09.html#open_source_internal_projects)

**Maintaining Strategic Momentum**

One of the reasons that WealthGrid found it so very hard to find a successful path to cloud native was pretty ironic: they were so good at doing what they did that they forgot how to do anything else. In other words, they were beautifully effective at the proficient delivery process. In the process of perfecting this near-algorithmic state, however, the company moved away from exploration and experimentation, and so stopped investing in creativity and innovation. Three Horizons[**3**](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#ch07fn3) is the pattern that makes sure this never happens again and that any company is able to adapt to change when it comes even while operating at maximum proficiency for the situation.

**Pattern: Three Horizons**

*Proportional allocation of resources among delivery, innovation, and research makes an organization responsive to change while reliably delivering core business value (*[*Figure 7-21*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#three_horizons01)*).*



**Figure 7-21. Three Horizons**

An established enterprise is aiming to run a large innovation initiative while all current teams are dedicated to incremental (proficient) work on existing products. Or, a startup is building a new product and devotes all resources to research and innovation.

**In This Context**

**In general, companies seldom keep the right balance between delivery and innovation. Enterprises tend to allocate almost all resources to Horizon 1, proficient delivery of core business product/service, which eventually leads to stagnation. Startups tend to overcommit to innovation, Horizon 2, which leads to poor product quality and lack of focus on delivery value to customers. At the very far end lies Horizon 3, researching ideas that are promising but will not lead to any practical solutions in the foreseeable future.**

* Creative and proficient teams need to be managed differently.
* Startups tend to focus on innovation, enterprises on incremental improvements.
* Complex innovation many times looks like just another technology project.

**Therefore**

**Always allocate resources to delivery (current products or services), innovation (refining new products/services or significantly improving existing ones, relevant within 12**–**24 months), and research (long-term ideas and technologies). Champions are responsible for moving technology and knowledge across the teams.**

An adaptive business constantly evaluates and recalibrates the relationship among its three horizons, pursuing the optimal balance between proficiency and creativity. To manage the Three Horizons in the context of a cloud native transformation across an organization, it’s important to have a Transformation Champion who understands the different stages and helps the company’s teams and leaders distribute the focus, technology, and strategy as needed. Once the transformation is complete, a Designated Strategist can serve in the same role to keep the company in balance going forward. (For a deeper explanation of McKinsey’s Three Horizons model, see [Chapter 6](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch06.html#tools_for_understanding_and_using).)

* Give 70%–80% of resources to Horizon 1, proficiency; 15%–25% to Horizon 2, innovation; and 3%–5% to Horizon 3, research.
* Adjust investment in horizons when circumstances change (to improve quality or prod innovation).
* After significant investment in innovation, move people back to improve quality on H1 projects.

**Consequently**

**The company is always prepared for whatever the future brings while still delivering existing products frequently and at high quality.**

* + Continuous innovation.
* + Continuous improvement.
* + Stable delivery of core business value.
* − Some resources are not allocated to products that immediately generate revenue.

**Common Pitfalls**

Enterprises begin as startups but over time lose the capability to work on Horizons 2 and 3. This happens because they are fully optimized for delivery of their current products, which is a totally proficient task. When the market changes or a new big opportunity arises, they approach the challenge as any other proficient project to be delivered under constant pressure of frequent deadlines and with long and detailed documentation. Even if the company sets very innovative goals, such initiatives will fall short, since the teams are given no space for free thinking and no psychological safety to experiment and make mistakes.

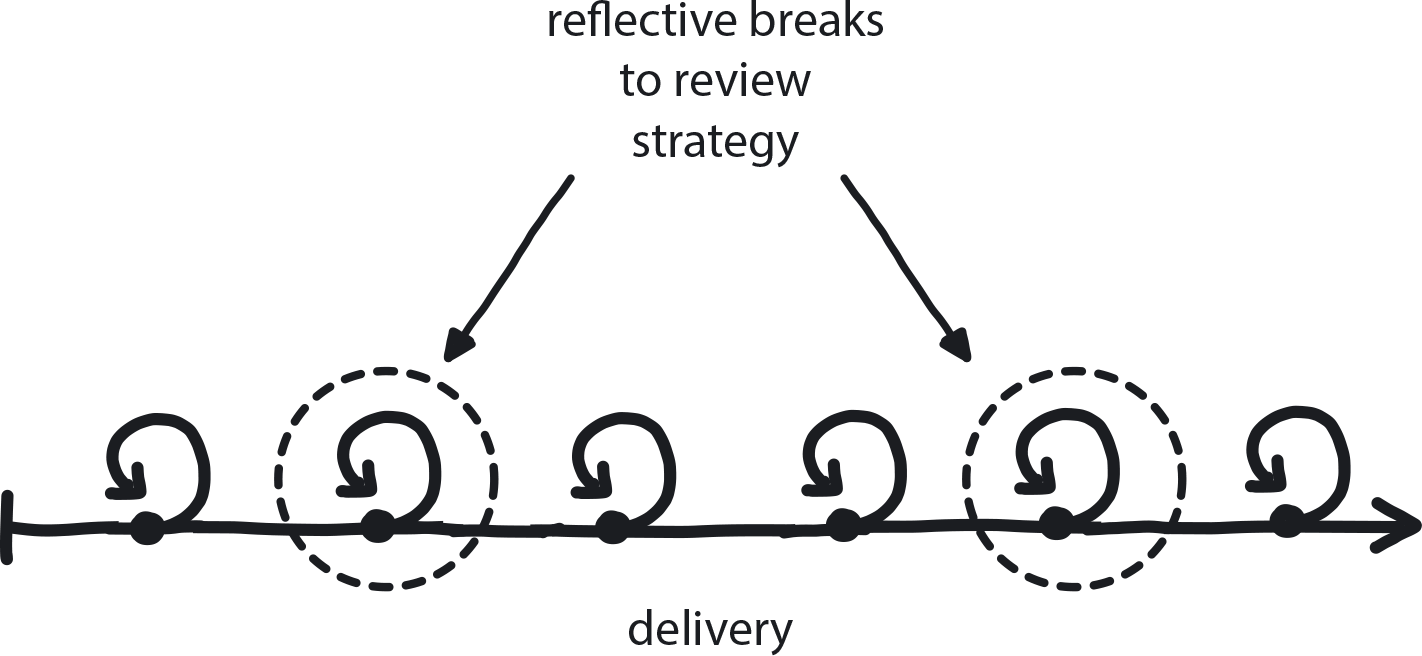
**Related Patterns**

* [Manage for Creativity](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_creativity)
* [Manage for Proficiency](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_proficiency)
* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Transformation Champion](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#transformation_champion)
* [Designated Strategist](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#designated_strategist)

Our last two patterns are essential for working in the Three Horizons model. Reflective Breaks make sure that everyone in the organization has a regular opportunity to take a step back and contemplate their current path and process. Naming a Designated Strategist means the company has someone specifically in charge of keeping an eye on the future.

**Pattern: Reflective Breaks**

*Build periodic times into the business delivery cycle dedicated to reviewing current strategy in light of shifting market conditions or other new information (*[*Figure 7-22*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#reflective_breaks01)*).*



**Figure 7-22. Reflective Breaks**

Strategy is in place, and the execution roadmap is clear. People are working under pressure to deliver according to the plan. The environment is still changing frequently and drastically.

**In This Context**

**When you run as fast as you can, you stop looking around to evaluate the situation and focus on only a single point**—**the finish line. Most modern delivery processes are designed to create stable pressure to help people focus on delivery. There is no planned and structured time set aside for periodic strategy reviews and for creative thinking.**

This is a good way to manage for proficiency and fast delivery of incremental, well-understood changes, but it leaves out the opportunity for radical change of direction that requires unstructured, open-ended time to think.

* People under pressure don’t reflect: to run fast, you look straight ahead—not side to side.
* It’s difficult to generate new ideas when focused on delivery. Innovation requires free time for thinking.
* The typical Agile environment creates continuous pressure to deliver new features in an unbroken cadence.

**Therefore**

**Build periodic planned “time-outs” into the business cycle across the entire organization.**

People think “taking a break” means that they get to stop working, but that is not the case in cloud native, or in any other big innovation initiative. The cloud native delivery cadence should include time for everyone in the organization to occasionally step back from their everyday responsibilities in order to do a different kind of work. Temporarily reducing the pressure to deliver your usual responsibilities makes room for thoughtful consideration of current conditions, both inside and outside the organization.

This periodic planned time-out applies across the entire organization. The cycle must reflect the enterprise’s specific circumstances, of course, but quarterly break periods across the calendar year are a logical approach that can be effective in many organizations. Other break examples could be an Agile org making every fourth (or fifth, sixth, or eighth) sprint a “rest” period before stepping back into focused delivery mode. Again, timing depends on factors specific to each company, such as development cycles and delivery schedule.

For the company’s leaders, Reflective Breaks are a pause from concentrating on forward-focused actions in order to review the existing environment—an opportunity to reflect on the strategy, review the architecture, inventory the backlogs, and basically re-evaluate the plan for the entire company.

For the development and business teams, it means taking time out from regular responsibilities like building new features and iterating improvements (or pursuing sales and marketing efforts, or administering internal company functions, etc.) in order to refocus on the strategic aspects of their work. The chance to step back and examine how everything you do every day as part of your job, from process to product, might be improved.

People don’t change strategy unless they have time to look at it. This is the chance to review architecture, adjust it if needed, and invest in nonfunctional improvements that will make everyone’s life better when the next sprint puts everybody back to busily executing their tasks.

* Make breaks on all levels.
* Use reflective breaks for executive team to review strategy, middle managers to review objectives, and execution teams to review backlogs and make nonfunctional improvements.

**Consequently**

**Teams are focused on execution but also have the regular opportunity to review and adjust on all levels of the company.**

* + Company spends the majority of time on focused delivery of profitable products/services but is still able to adjust direction.
* + Cyclical breaks are a crucible for innovation, which requires free time and bandwidth to think and create.
* − People tend to schedule their vacation time for these breaks and may not take full advantage of the opportunity to reflect on their day-to-day work.

**Common Pitfalls**

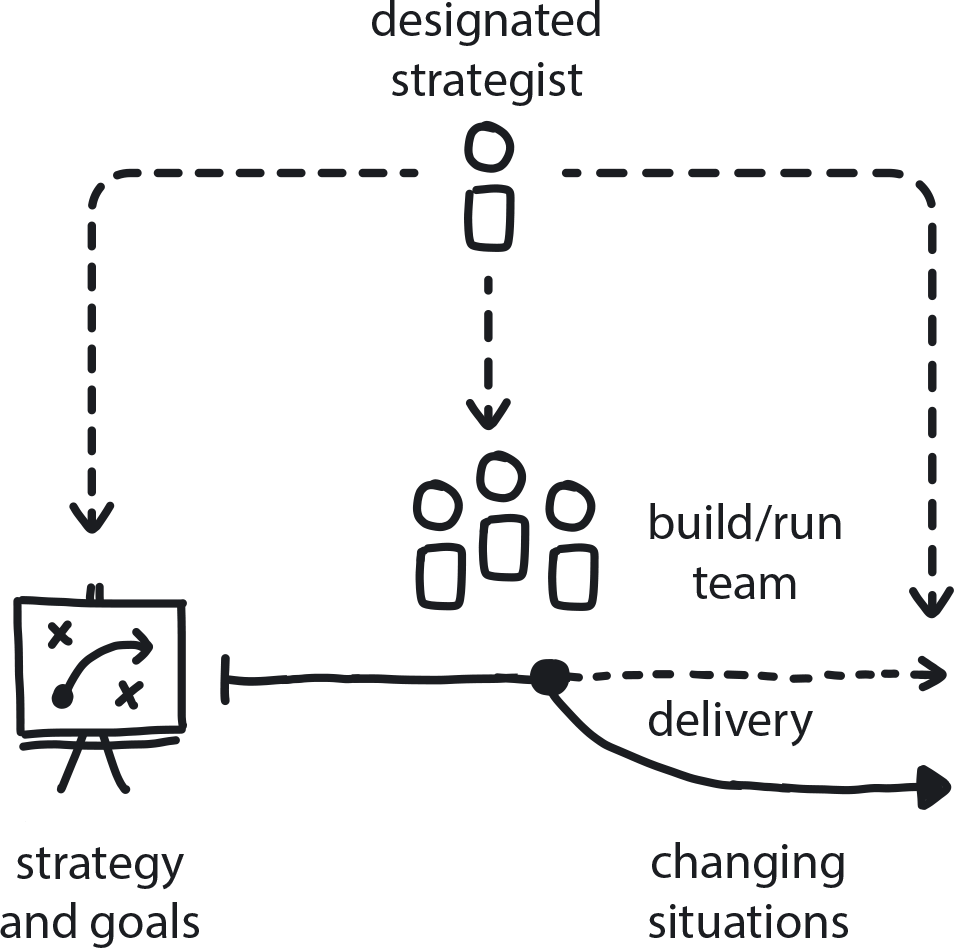
Trying to remain always 100% focused, which means people are overworked and don’t see anything around them. Sometimes a company claims to value reflective breaks but then embeds them as just another part of the delivery process. This means you are relying on individuals to take this time for reflection; some will do it and some will not. If there is a lot of time pressure, people will almost always choose what is urgent over what is merely important. This is why you need to build it in as a company-wide cycle.

**Related Patterns**

* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Learning Organization](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#learning_organization)
* [Decide Closest to the Action](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#decide_closest_to_the_action)
* [Manage for Proficiency](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_proficiency)
* [Manage for Creativity](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch08.html#manage_for_creativity)

**Pattern: Designated Strategist**

*When you’re running forward as fast as you can, it’s difficult to look around you—so make one person within the organization in charge of situational awareness (*[*Figure 7-23*](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#designated_strategist01)*).*



**Figure 7-23. Designated Strategist**

The company has a strategy and a clear execution backlog. The entire team is focused on proficiently delivering the best products. The market and the company situation are changing as the project progresses.

**In This Context**

**Once your transformation strategy has been defined, the team will tend to enter full execution mode—and stop refactoring the goals. This leads to the achievement of the original goals, but with no ongoing evaluation as circumstances evolve/change. People under stress of delivery can’t look around and re-evaluate the situation since they are pressured to focus solely on the set goals. The problem is that you might arrive at the finish line only to find that the problem has changed completely while you weren’t paying attention and your original solution no longer applies.**

* It’s difficult for people to do broad strategic and deep tactical tasks at the same time.
* Scrum, the most widely used software delivery methodology, is heavily focused on proficiency and creates pressure to deliver as much as possible as fast as possible. By default, it leaves no space for reflection.

**Therefore**

**Free one of the experienced architects or managers to focus solely on the future and evaluate all the scheduled tasks based on long-term goals.**

The Designated Strategist should have no short-term commitments in order to prevent them from getting drawn into “all hands on deck” delivery deadlines or firefighting mode. This person functions as the “lookout” (as in, the person with the spyglass on the crow’s nest of a sailing ship) while the executive team is at the wheel of the ship. The lookout warns of rocks ahead, or promising destinations to explore—and then the company’s strategic leadership decides how to turn the ship.

* Make sure strategist is not working on daily tasks.
* Strategist has no executive power.
* Can also do evangelizing.
* Connected well to both customer desires and business goals.
* Balance is important.

**Consequently**

**Teams can focus mainly on delivery while the company still maintains a strategic perspective.**

* + Teams are in charge of their work.
* + Strategist acts as advisor for adjusting goals and direction.
* − Strategist may create distractions and constant instability by suggesting too many direction changes.

**Common Pitfalls**

Making this only a part-time responsibility and giving the Designated Strategist other, unrelated tasks that distract from their focus on strategic analysis and planning. Urgent tasks almost always get prioritized higher than important tasks.

**Related Biases**

*Authority bias*

The tendency to attribute greater accuracy to the opinion of an authority figure (unrelated to its content) and to be more influenced by that opinion.

**Related Patterns**

* [Dynamic Strategy](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#dynamic_strategy)
* [Three Horizons](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#three_horizons_pattern)
* [Periodic Checkups](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#periodic_checkups)

**Ready for Next**

Our next three chapters introduce many more patterns for optimizing your organization and culture for cloud native, evolving your development processes, and building your new infrastructure. These are all things that you will do, actions you will take in the process of transforming your company.

These upcoming chapters deal with technologies and how to use them, and will need to be updated eventually. And sooner rather than later: technology changes fast, and how we use it also changes. The patterns and material in this chapter, by contrast, won’t really change. They exist to teach you *how* to do these things. Together, they form a roadmap and a cognitive tool set for managing change no matter where, and no matter when—whether in the middle of your transformation, when a sudden new competitor appears, or a few years from now when the next paradigm shift arrives to replace cloud native. These patterns for strategy and risk management aren’t simply tools to make you ready for cloud native: they help make you ready for *next.*

New cloud native patterns are emerging constantly. To continue sharing them and extending the cloud native pattern language we have established here, please visit [www.CNpatterns.org](http://www.cnpatterns.org/).

This is where to find the latest pattern developments, but also an online community for discussing and creating new patterns. We are inviting people from across the industry, thought leaders and influencers but most importantly everyday engineers and managers—those out there working elbows-deep in cloud native code and architecture—to contribute and participate. Hope to see you there!

[**1**](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#ch07fn1-marker) Henry Mintzberg’s original whitepaper, [“On Strategy, Deliberate and Emergent.”](http://bit.ly/2OkMSpJ)

[**2**](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#ch07fn2-marker) Gradually Raising the Stakes and its Related Patterns are drawn from multiple significant sources, including the book *Exploring Strategy: Text and Cases* by Duncan Angwin et al. (Pearson) and the Harvard Business Review article “Strategy Under Uncertainty,” by Hugh Courtney et al.

[**3**](https://learning.oreilly.com/library/view/cloud-native-transformation/9781492048893/ch07.html#ch07fn3-marker) This pattern is drawn from the concepts in the McKinsey Quarterly article [“Enduring Ideas: The Three Horizons of Growth,”](https://mck.co/35wZFLj).