Best practices for lean development governance, Part 1: Principles and organization

Scott W. Ambler
Per Kroll

from The Rational Edge: The first in a series of articles covering IBM Rational's recommended approach to governing modern software development efforts, this article explores the mission and principles of lean governance, along with the organization and stakeholder collaboration required for project-by-project success.

The goal of an IT governance program is to establish chains of responsibility, authority, and communication to empower people in support of the overall enterprise goals and strategy. You do this by balancing risk versus return on IT investments, setting in place effective processes and practices, defining the direction and goals for the department, and defining the roles which people play within and with the department. Development governance is an important subset of IT governance, the scope of which covers the steering of software and system development projects. The focus of this three-part article series is on practices which support lean development governance.

Traditional governance often focuses on command-and-control strategies which strive to manage and direct development project teams in an explicit manner. Although this is a valid and effective strategy in some situations, for many organizations this approach is akin to herding cats -- you'll put a lot of work into the governance effort but achieve very little in practice. Lean governance focuses on collaborative strategies that strive to enable and motivate team members implicitly. For example, the traditional approach to coding guidelines would be to create them and then enforce their usage through formal inspections. The lean approach would be to write the guidelines collaboratively with your programmers, explain why it is important for everyone to adopt the guidelines, and then provide tooling and support to make it as easy as possible for developers to
follow those guidelines. This lean governance approach is akin to leading cats; if you grab a piece of raw fish, cats will follow you wherever you want to go.

This is the first in a series of three articles that will explain our recommended approach to governing modern software development efforts. We are describing this approach for several reasons:

1. Our experiences with existing approaches to IT governance -- such as Control Objectives for Information-Related Technology (CobiT), and the Project Management Institute (PMI)’s Organizational Project Management Maturity Model (OPM-3) -- are that these traditional approaches are often too heavy in practice for many organizations, although they provide a wealth of advice.

2. More and more project teams are following an agile approach to software development,¹ and these project teams must be governed effectively.

3. We believe that many non-agile teams could benefit from the more collaborative approach described in this paper.

4. We believe that organizations that have adopted the traditional frameworks listed above, or similar frameworks, can still benefit from "loosening the reins" on their approach to governing development projects.

This first article, Part 1, will explain the mission and principles of lean governance, along with the organization and stakeholder collaboration required for project-by-project success. Parts II and III will cover, respectively, processes and metrics; and roles/responsibilities, and policies/standards.

Introducing the best practices

Figure 1 categorizes and illustrates the relationships of the practices of lean governance, and Table 1 overviews each of them in alphabetical order. (The three articles in this series will explore each practice in detail.)

Figure 1: The practices of lean software development governance.
Table 1. Overview of the Lean Development Governance Practices, listed alphabetically here.

<table>
<thead>
<tr>
<th>Practice</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapt the Process</td>
<td>Because teams vary in size, distribution, purpose, criticality, need for oversight, and member skillset, one process size does not fit all. You must tailor the process to meet a team’s exact needs. Furthermore, processes must be evaluated and allowed to evolve over time to meet the changing needs of your organization.</td>
</tr>
<tr>
<td>Align HR Policies With IT Values</td>
<td>Hiring, retaining, and promoting technical staff requires different strategies compared to non-technical staff. You need to establish specific incentives/rewards appropriate for the mindset of your technical staff to ensure timely delivery as well as other key accomplishments, such as retraining on a new technology.</td>
</tr>
<tr>
<td>Align Team Structure With Architecture</td>
<td>The organization of your project team should reflect the desired architectural structure of the system you are building to streamline the activities of the team. For example, distributed teams are well suited for building partitioned architectures; centralized teams are often more adept at building centralized architectures.</td>
</tr>
<tr>
<td>Business-Driven Project Pipeline</td>
<td>You should invest in the projects that are well-aligned to the business direction, return definable value, and match well with the priorities of the enterprise. This approach implies less focus on “cool technology” and greater focus on supporting the business.</td>
</tr>
<tr>
<td>Continuous Improvement</td>
<td>You should strive to identify and act on lessons learned throughout the project, not just at the end. For example, a quick retrospective at the end of each iteration and a more detailed retrospective at critical milestones is a straightforward way to improve as a project progresses.</td>
</tr>
<tr>
<td>Continuous Project Monitoring</td>
<td>Automated metrics gathering enables you to monitor projects and thereby identify potential issues so that you can collaborate closely with the project team to resolve problems early. You will also initially need to identify a minimal set of measurements to collect.</td>
</tr>
<tr>
<td>Embedded Compliance</td>
<td>It is better to build compliance into your day-to-day process, instead of having a separate compliance process that often results in unnecessary overhead. Automation is critical.</td>
</tr>
<tr>
<td>Flexible Architectures</td>
<td>Architectures that are service-oriented, component-based, or object-oriented and implement common architectural and design patterns lend themselves to greater levels of consistency, reuse, enhanceability, and adaptability.</td>
</tr>
<tr>
<td>Integrated Lifecycle Environment</td>
<td>You should strive to automate as much of the “drudge work” -- such as metrics gathering and system build -- as possible. Your tools and processes should fit together effectively throughout the lifecycle. An initial investment at the beginning of a project to set up your toolset can pay significant dividends in reduced work effort throughout a project.</td>
</tr>
<tr>
<td>Iterative Development</td>
<td>An iterative approach to software delivery allows progressive development and disclosure of software components, with a reduction of overall failure risk, and provides an ability to make fine-grained adjustment and correction with minimal lost time for rework.</td>
</tr>
<tr>
<td>Pragmatic Governance Body</td>
<td>Effective governance bodies focus on enabling development teams in a cost-effective and timely manner. They typically have a small core staff with a majority of members being representatives from the governed organizations.</td>
</tr>
<tr>
<td>Risk-Based Milestones</td>
<td>You want to mitigate the risks of your project, in particular business and technical risks, early in the lifecycle. You do this by having throughout your project several milestones that teams work toward. The goal of each milestone is to address one or more risks, such as the Lifecycle Architecture (LCA) milestone in the Rational Unified Process (RUP)</td>
</tr>
</tbody>
</table>
that requires your architecture be proven through working code before
construction begins.

**Scenario-Driven Development**

The whole cannot be defined without understanding the parts, and the parts cannot be defined in detail without understanding the whole. By taking a scenario-driven approach, you can understand how people will actually use your system, thereby enabling you to build something that meets their actual needs.

**Self-Organizing Teams**

The best people for planning work are the ones who are going to do it. IT professionals should be respected as intelligent people who can determine their own strategies for working together. When given a bit of coaching and guidance, they can plan their work within established parameters, such as iteration boundaries.

**Simple and Relevant Metrics**

You should automate metrics collection as much as possible, minimize the number of metrics collected, and know why you're collecting them.

**Staged Program Delivery**

Programs, which are collections of related projects, should be rolled out in increments over time. Instead of holding back a release to wait for a subproject, each individual subproject must sign up to predetermined release dates. If the subproject misses, it skips to the next release, minimizing the impact to the customers of the program.

**Valued IT Assets**

Guidance, such as programming guidelines or database design conventions, and reusable assets such as frameworks and components, will be adopted if they are perceived to add value to developers. You want to make it as easy as possible for developers to comply with, and more importantly take advantage of, your corporate IT infrastructure.

The rest of this article will describe a) the mission and principles of a Pragmatic Governance Body as well as Staged Program Delivery; and b) the organization and meetings referred to as Business-Driven Project Pipeline and Scenario-Driven Development. (Parts II and III of this article series will cover the remaining best practices presented above.)

For each best practice presented, the discussion will cover its benefits, trade-offs involved (i.e., what requirements must be met within the organization in order to maintain the practice going forward), antipatterns (i.e., what behaviors are opposite to the practice and will threaten its implementation or negate its benefits), and default recommendations (i.e., the minimum steps to be taken in order to establish this practice in an organization).

**Mission and principles**

The practices in this category provide clear direction and underlying principles to encourage the right behavior. These practices are:

- Pragmatic Governance Body
- Staged Program Delivery

**Pragmatic Governance Body**

A governance program does not run itself; a group of people called a governance body runs it. The manner in which the governance body organizes and conducts itself is a key determinant of the overall effectiveness of your governance program. A pragmatic governance body focuses on enabling IT professionals first, and on controlling and managing them second. It does this by:

- Creating an environment where people can be effective.
• Promoting situational-specific strategies, procedures, and practices.
• Providing teams with access to the resources they need, including ready access to business stakeholders.
• Providing guidance, support, and mentoring to teams that have deviated from expected norms.

Benefits

There are two primary benefits to this practice:

1. **It promotes the spread of effective governance.** IT teams will be much more likely to conform to your organization’s governance program if you make it easy and desirable for them to do so.
2. **It makes governance actionable.** Governance is just shelf-ware unless people step up and make the processes and polices come alive in the organization to help the organization reach its goals.

Trade-offs

There are several trade-offs associated with this practice:

• **Requires support of the business.** Your governance program must reflect the needs of the business. To ensure this, business stakeholders must be actively involved on the governance body. A pragmatic governance body cannot be made up of IT people only.
• **Requires continued investment.** A governance body must be staffed, and since governance is typically a long-term commitment, there is a need for ongoing investment.
• **Control is relinquished to executing body.** Although the focus of a pragmatic governance body is on enabling teams, they still have the responsibility to direct them and assure that they are operating within allowable norms.

Anti-patterns

The following anti-patterns are related to governance bodies:

• **Governance for governance sake.** Also known as "Bureaucrats Run Amok," this is a governance body whose focus is on ensuring that the appropriate documentation is created by development teams at the appropriate times. This approach is not an enabler of better business execution.
• **Control through governance.** This is a governance body whose focus is controlling and directing development teams instead of enabling them. You know that this is occurring when the governance body spends most of its time crafting edicts, procedures, and/or policies for teams to follow.

Recommended default

Create a small central team, often referred to as a governance competency center, which has a dotted line reporting structure to both IT and to the appropriate business units.
Staged Program Delivery

The average IT project is becoming smaller and smaller. Smaller projects have been shown on average to have more successful project outcomes,\textsuperscript{2} see Figure 1, and are proven to be more productive. At the same time, comprehensive business processes require the support of an increasingly complex set of IT systems. This means that delivering major business benefits requires the coordinated execution of a set of projects centered on a common mission and frequently requiring coordination of resources, technologies, architectures, and timelines. This is done through program management.

Effective program management delivers incremental value to the business by being clever about how to stage/schedule the program so that a subset of all projects within the program are coordinated around a core set of business goal. This typically requires breakdown of larger goals to subgoals, and mapping projects to deliver upon these subgoals. The program will then drive resource allocation and management of the relevant set of projects to deliver upon the relevant business goals or subgoals as early as possible, what is called staged program delivery, as shown in Figure 2.

![The Program Lifecycle - Stages](image)

\textbf{Figure 2:} Staged program delivery means that projects within a program are divided into groups so each group of projects can deliver incremental value as early as possible. This figure is borrowed from Kurt Bittner and Ian Spence's book \textit{Managing Iterative Software Development Projects}, Addison-Wesley Professional: 2006.

Once a program stage has been defined, the projects within that stage are managed through a control project, with the same phases as a normal RUP project, as shown in Figure 3.

The control project manages integration and provides management milestones centered on risk and variance reduction. The control project is assessed by doing an aggregate assessment of all projects. To assess the risk profile for the entire program, however, you cannot simply
aggregate the risk profile for each project; you also need to ensure that risks related to cross-project integration have been addressed. Usage of a control project enables coordinated execution of the program while providing flexibility in execution of individual projects within the project. Iterations play a fundamental role here, since they provide stable anchoring points to allow projects within the program to drive and validate meaningful cross project integration.

**Figure 3. By managing a program stage through a control project, risk reduction and value creation can be effectively managed.** This figure is borrowed from Kurt Bittner and Ian Spence's book *Managing Iterative Software Development Projects*, Addison-Wesley Professional: 2006.

**Benefits**

Staged program delivery, especially when leveraging a control project, provides a number of benefits:

- **Effective execution centered on business objectives.** By grouping projects according to business objectives, and managing them as a program, we can more effectively deliver on major business objectives.
- **Coordinated release of many related parts.** Usage of a control project provides us with well-defined governance milestones for the program through a set of milestones that focus on risk and variance reduction as well as value creation.
- **Incremental value delivery.** Dividing a potentially very large program delivers value incrementally around business subgoals. By managing projects within the program via iterative development techniques, evolving capabilities can be deployed as necessary.
- **Improved efficiency.** Semi-independent execution of projects enables efficient development within the various projects, because this provides them with as much tactical flexibility as possible, thus enabling higher productivity. We do, however, need visible control points to allow assessment of individual project performance as well as the program as a whole. The
performance of individual projects can be assessed at the end of each iteration using fact-based assessments, assuming these projects use iterative development.

**Trade-offs**

There are two important trade-offs for effective staged program delivery.

- **Increased coordination.** Executing a program requires coordination, which requires resources and knowledge.
- **Risk of delay.** When not managed well, a program may cause release to be determined by the slowest moving project. In the worst case, this will cause a majority of project teams to be held up for the "greater good" of the company, but the reality is that it's often to the detriment of the company. The trick is to have good oversight, and redefine your program scope when a given project does not keep up, so you still can get some business value delivered early.

**Anti-patterns**

The following anti-pattern is associated with managing project pipeline by business value:

- **Slowest drum beat.** The slowest project within a program determines the speed of the overall effort. An effective program is seen as a train -- the train is leaving the station at a certain time, and either a project is on it or not.

**Recommended default**

We recommend smaller programs managed by a control project run according to RUP’s four phases, with loosely coupled individual projects, each following RUP’s iterative approach.

**Organization and Meetings**

The practices in this category provide guidance for defining the appropriate organizational structure and formal reporting mechanisms to focus at the right level with the right stakeholders. These practices are:

- Business-Driven Project Pipeline
- Scenario-Driven Development

**Business-driven project pipeline**

The needs for IT projects are always greater than available resources, which forces us to prioritize among potential projects in our pipeline and optimize the selected projects to align with our strategy. A good governance solution should maximize business value of development investments made, and ensure strategic alignment with business goals and objectives, and we need a mechanism for accomplishing this. This can be done effectively by leveraging scorecards and other portfolio management strategies, where each project is assessed against a defined set of parameters reflecting strategic alignment and business value.
It should be noted that scorecards are a model of business value, and all models have their limitations. This means that you may occasionally find that a project with lower score should be prioritized over one with a higher score, forcing manual overrides, but also in this case, scorecards force a discussion about strategy and business value, revealing why the model was not perfect in some rare cases.

Benefits

Managing the project pipeline by business value carries the following benefits:

- **Convergence on business value.** Pipeline management forces the organization to agree on the business strategy and what parameters drive the most business value. As an example, consider these questions: Should your IT organization prioritize supporting new business opportunities, improving cost efficiency of ongoing business processes, or enabling more rapid organizational growth? These questions force discussions around funding of projects centered on strategic alignment and business value.
- **Improved ROI.** Projects with the best mix of risk, return on value, and strategy alignment are selected, driving higher overall ROI. Scorecards also provide a powerful communication and motivational mechanism for the entire IT department to align with strategy and maximize business value. If you want your project to be funded, you need to ensure that it aligns well with what is most important to the business.
- **Improved transparency.** In many organizations it is a mystery how funding decisions are made. Pipeline management by business value can be done in an open and transparent manner, which improves visibility and builds trust.

Trade-offs

Trade-offs to consider when managing project pipeline by business value include:

- **Strategy and value articulation.** Pipeline management forces you to define what should drive your investments. This is difficult, and forces you to deal with what your strategy "needs to be" versus going by "gut feel."
- **Relinquished control.** The process becomes more objective and open, requiring each project to articulate its business value. This may not appeal to everybody, particularly people who are happy with the status quo.
- **Frequency.** Pipeline management needs to be done on a regular basis, such as weekly or monthly, so that projects are not delayed. This requires a substantial investment to make work effectively.

Anti-patterns

The following anti-patterns are associated with managing project pipelines:

- **Subjective project score.** If project selection becomes subjective, different from one project to the next, and if decisions are not traceable or defendable, you have a poorly functioning scoring processes. (Maybe your scorecard is hard to use, forcing too much subjectivity, or scoring is done without any proper oversight and transparency.)
• **Gamed scorecard.** We have seen many cases where scorecards sound good on paper, but in reality, they prioritize the wrong projects. This is done by over emphasizing issues so that "pet projects" come out on top, or by simply miss-scoring projects to get the political answer that you desire. You will typically need to calibrate your scorecard until you get it right.

• **Portfolio burden.** If the process for managing the pipeline is highly bureaucratic and/or adds a significant time lag into your project initiation efforts, the value of the scoring may not motivate the drawbacks.

**Recommended default**

We recommend that you focus on less than five parameters in your scorecard, and that you use scoring as an aid in a business discussion about prioritization rather than as a strict measure.

**Scenario-Driven Development**

Under the practice for *Staged program delivery* through control projects, we discussed that comprehensive business processes requires the support of an increasingly complex set of IT systems, and that smaller projects are proven to be more effective and have higher success rates than larger projects. We can view the IT System as one system, which is broken down into a set of smaller parts, where each part is developed as a system of its own, as shown in Figure 4. This configuration is called a "system of systems," where the parts are the individual applications, developed by individual projects.

The whole cannot be defined without understanding the parts, and the parts cannot be defined in detail without understanding the whole. The risk is that we will not understand how the parts impact the overall solution, and we will get bogged down in building parts that don't fit together. We hence need to apply a technique that allows us to keep the whole and parts tightly together. We do this by applying use cases at all levels:

• Requirements are defined and traded through use-case scenarios -- that is, by showing how the parts needs to collaborate to accomplish system-wide use cases.
• Architectures and interfaces are defined and traded in the same manner.
• Use case scenarios also drive system-level integration and testing scenarios, ensuring that the parts deliver the key end-to-end use cases of the overall system.
We are today applying this technique very successfully within IBM Software Group. We develop hundreds of applications by hundreds of teams, and customers are using dozens of our products together. Over the last few years we have focused more and more on these end-to-end business scenarios (sometimes called green threads) for an overall solution that drives better integration of the parts. We have found that this allows us to prioritize our resources so they add the most value to our customers.

Benefits

Scenario-driven development using system-of-systems thinking carries many benefits:

- **Focus on delivering business capabilities.** It forces each project to focus on how their part delivers business value to the organization.
- **Better integration between many moving parts.** It forces projects developing different applications to focus on good integration between applications. Focus on the integration is crucial, since business value is primarily delivered through end-to-end business scenarios, rather than through individual parts by themselves.
- **Provides control mechanisms.** Control mechanisms ensure that teams are delivering business value. Furthermore, this sort of scenario-driven approach helps eliminate gaps in capability and can identify inconsistencies and areas where progress is lacking.

Trade-offs

There are two important trade-offs associated with scenario-driven development:

- **Business and technical vision.** This practice requires overall business and technical vision, and not all organizations are at a sufficient level of maturity to provide that.
• **Enterprise level coordination.** This practice is based on the ability to coordinate the development of the different parts, and not all organizations have the ability to deal with coordination at this grand scale.

Anti-patterns

The following anti-patterns are associated with scenario-driven development:

• **Private visions.** Business scenarios are only loosely captured in peoples' heads, but never shared or documented. End-to-end business scenarios are never articulated, thus they are never delivered upon.

• **IT-driven business.** Although the role of IT is to support the business, in this anti-pattern the opposite occurs. This happens when no business processes are defined or are used to drive incremental IT capabilities.

• **Ivory-tower business model.** Business processes are developed, but they are not effectively leveraged by IT projects to drive incremental IT capabilities.

• **Silo projects.** Each project is doing its best to deliver business value, but there is no articulation of end-to-end business processes, nor is their coordination between projects.

Recommended default

Capture key business processes and use them to drive and coordinate IT investments.

Coming next month...

Next month, Part 2 of this article series will cover Processes -- including Iterative development, Risk-Based Milestones, and Continuous Improvement -- as well as measurements and monitoring methods. Stay tuned!

Notes

1 Scott Ambler ran an Agile Adoption Survey for Dr. Dobb's Journal (www.ddj.com) in March 2007 which found that 69% of respondents worked in organizations that had adopted one or more agile techniques and that another 7% felt their organizations would do so within a year. The survey is summarized in the August 2007 edition of DDJ.

2 The Standish Group's Chaos Report v 3 (2003) demonstrated that failure rate increases with project size.

Acknowledgments

We want to thank Michael Hanford and Victor Ramos for their reviews of this article series.