Cutting IT Costs by Applying Lean Principles

Software Measurement for Lean Application Management

IT organizations are experimenting with Lean techniques to reduce application costs and delivery times with mixed results. To achieve these objectives, IT must eliminate the largest source of waste in application development and maintenance – defects and the enormous rework they cause. This paper shows how focusing on the Jidoka pillar of the Toyota Production System, which is the use of automation to detect and eliminate defects early, attacks and reduces defects and rework. By applying the waste-reducing principles of Jidoka to development, maintenance, and the design of applications, IT can both cut costs and shorten time-to-service.

Dr. Bill Curtis
Senior Vice President and Chief Scientist, CAST
Director, Consortium for IT Software Quality
Executive Summary

Dr. Bill Curtis, SVP and Chief Scientist at CAST, explains how the Lean practices pioneered in the Toyota Production System apply to the development and maintenance of application software, leading to decreased total cost of ownership and improved business responsiveness and operational dependability.

The Problem

- IT organizations have been applying Lean practices to the development and maintenance of application software, yet the powerful benefits have yet to be fully realized because they have not focused on the early detection and elimination of defects.

- Defects in source code and the rework they cause are the largest sources of waste in application development – accounting for 30% to 50% of development effort.

- As much as two thirds of all application maintenance effort can be classified as waste due to rework caused by defects and time spent trying to understand what is going on in poorly constructed applications.

- Over time, applications become less responsive and obese with code that is no longer needed, wasting machine resources.

- The fast pace and short release cycles of Agile truncates the time available for identifying and remediating structural problems in the source code of an application.

The Solution

- By focusing on Jidoka, the pillar of Lean involving the automated early detection and elimination of defects, defects can be identified when they are injected into the source code and fixed quickly, rather than later in the life cycle when their repair is often 10 times more time-consuming and expensive.
• Eliminating the causes of defects and continually training developers for defect-free work can dramatically improve productivity and shorten time-to-release.

• In application maintenance, it is vital to address the changeability of code before its release to reduce the time needed to understand the application for subsequent releases, and sustain the quality of the application over its lifetime.

• Over an application’s operational life, you must continuously tune the performance of the code and manage application size, especially when deploying it in a cloud environment.

Key Takeaways

• Through adherence to the Jidoka component of Lean, the combination of Lean Application Development, Lean Application Maintenance, and Lean Application Assets provides the foundation for Lean Application Management.

• Software analysis and measurement is key to supporting Lean Application Management. The automated analysis and measurement of structural quality of applications is critical to the overall cost of ownership, business responsiveness, and operational dependability required to run a Lean Enterprise.

• Lean processes work at their most efficient when performed on Lean products. The best way to reduce waste is to design it out of the application from the beginning by driving design decisions based on prioritized quality goals.
I. How Lean Applies to Software Applications

For the past several years, IT organizations have been exploring the application of Lean practices pioneered in the Toyota Production System\(^1\) for automobile assembly to the development and maintenance of application software. Its focus on empowered teams and process streamlining\(^3\) has popularized Lean in the Agile Methods community. However, some of the most powerful benefits of Lean have yet to be fully realized in application development since the Agile community has focused more on the just-in-time aspects of Lean rather than on Jidoka, the automated early detection and elimination of defects.

Jidoka is one of the two pillars of the Toyota Production System that provides the foundation for Lean practices (Figure 1). Jidoka uses automation to detect defects early in the production process, as well as their causes so they can be identified and eliminated. The fifth principle of what has become known as the Toyota Way is to “build a culture of stopping to fix problems and get quality right the first time.”\(^5\) This principle had a profound effect on the productivity of automobile manufacturing, and has the potential to transform the economics of application development as well.

![Figure 1 – The Toyota Production System](image-url)
Highlights

An IT organization can establish **Lean Application Management** by automating software measurement that supports key Lean practices.

Jidoka can be implemented in an application portfolio through the practices of Lean Application Management, which focuses on applying Lean methods in three areas: Application Development, Application Maintenance, and Application Assets. In the context of development and maintenance of business applications, Jidoka involves the use of technologies such as static analysis to identify defects when they are added to the source code, so they can be fixed before being passed on to a later life cycle phase where their repair is much more expensive. The sections that follow describe how an IT organization can establish Lean Application Management by automating software measurement that supports key Lean practices in each of these three areas.

II. Lean Application Development

The largest opportunity for improving quality and productivity during application development is by eliminating its largest sources of waste – defects and the rework they cause. In many organizations, 30% to 50% of development effort is devoted to rework. These staggering numbers are exacerbated by the fact that defects become 10 times more expensive to fix for each major phase of the software life cycle they slip past. Under these circumstances, productivity is largely determined by quality.

Before the Toyota Production System, 25% of the effort in traditional mass production automobile manufacturing was devoted to fixing defects in fully assembled automobiles at the end of the production line. Defects that would have been inexpensive to fix during assembly became expensive because large portions of the car had to be disassembled to make the needed repairs. Lean practices in the Toyota Production System virtually eliminated rework areas at the end of production lines, contributing heavily to Lean’s dramatic improvements in productivity, quality, and time-to-market over traditional mass production.
The level of rework in software applications is actually worse than that observed in traditional automobile assembly (Figure 2). As in mass production automobile manufacturing, rework in application development piles up at the end of the development cycle, causing delays and often throwing the project into chaotic thrashing to get the application sufficiently stable for release. This rework is manifest in the continuing cycle of testing, recoding defective components, and then retesting them to find even more defects.

Why doesn’t testing eliminate the rework problem? Modern testing practices are generally effective in detecting functional defects – deviations from what the customer wanted. However, testing is not as effective in detecting the non-functional defects that represent structural flaws in the construction of the application. These structural defects are difficult to detect through testing and are frequently the most expensive to fix because they involve interactions between multiple tiers of the application, which are often written in different languages and hosted on different platforms. Structural defects are at the root of most devastating operational problems such as outages, security breaches, data corruption, and performance degradation. Static analysis of the structural quality of business applications augments other quality assurance techniques such as testing and design inspections to eliminate such defects.
Cutting IT Costs by Applying Lean Principles
Page 7

Highlights

If structural defects escape into operations, the cost of correcting them can expand by an order of magnitude.

There are three main keys to eliminating rework (Figure 3) in Lean Application Development.

1. **Detect and remediate defects early**: Every component added to the source code should be analyzed at build time when it is integrated with the evolving application to see its impact on the system as a whole. Detection and repair at this point can be an order of magnitude cheaper than if these structural flaws slip into the final stages of testing, when they are deeply embedded in the application and a larger portion of the code has to be torn down, fixed, and rebuilt. If structural defects escape into operations where multiple versions of the software are in use, the cost of correcting them can expand by an additional order of magnitude.

   Early detection of defects is especially critical when using Agile. Agile methods are popular development techniques, but their fast pace and short release cycles truncate the time available for identifying and remediating the structural problems remaining in the source code. Although Agile teams typically rebuild their application daily to integrate newly developed components, their short schedules do not allow time for evaluating structural quality at the application level unless the analysis is automated. Agile teams...
Highlights

Organizations that rigorously apply Lean techniques have been able to reduce rework by 80% to 90%.

benefit from detecting structural defects early so the most severe ones can be remediated before release if a schedule buffer provides time for addressing them. Addressing structural problems prior to release helps prevent waste in maintenance.

2. **Eliminate the causes of defects:** When structural quality diagnostics are coupled with a powerful root cause analysis method such as Orthogonal Defect Classification\(^2\), application teams can identify the process sources of structural defects so that their root causes can be eliminated. Typically an organization will not investigate the causes of every defect, but instead focus on those that recur most frequently or have the potential for devastating impact. While early detection minimizes rework, root cause-based process improvement virtually eliminates it.

3. **Train developers continually for defect-free work:** Techniques such as Pareto charts of structural quality violations identify the most frequent types of structural defects being injected into applications. The results can be fed into training activities to help developers avoid these recurring structural defects in the future, further eliminating sources of rework. Some projects have a formal launch process for each phase of development, giving the opportunity to review current issues and the types of errors to be avoided during the next phase. Developing workflow and a knowledge center to classify and guide developers in understanding and remediating structural defects can raise the bar on the performance of an application development organization.

Combinations of these three Lean techniques enabled by software analysis and measurement technology can reduce, and in many cases eliminate, the structural defects that cause the most severe rework in application development. Organizations that apply these techniques rigorously have been able to reduce rework by 80% to 90\(^{13}\), dramatically improving productivity and shortening time-to-release.
As much as two-thirds of all maintenance effort can be classified as waste resulting from poor application design and coding.

III. Lean Application Maintenance

The two primary sources of waste in application maintenance are rework due to defects or badly constructed software and the staggering time maintenance staff spend trying to understand what is going on in poorly constructed applications. Correcting defects frequently accounts for as much as one third of all post-release work, and time spent understanding the code has been shown to account for as much as half of all the effort expended by maintenance staff\textsuperscript{14,15} (Figure 4). When the overlap between these two activities is removed, as much as two thirds of all maintenance effort can be classified as waste. Since these sources of waste result from poor application design and coding, they must be addressed as a critical focus of Lean Application Maintenance.

![Figure 4 – How Maintenance Staff Spend Their Time\textsuperscript{15}](image)

By identifying structural defects that have a high probability of causing operational problems and result in maintenance rework, software analysis and measurement provides strong support for Lean Application Maintenance. In addition, by addressing the changeability of the code before a release, the time spent understanding the application can be dramatically reduced in maintaining subsequent releases. Similarly, by sustaining the quality of the application over its lifetime, IT can avoid degraded applications and costly redevelopments.
There are three main keys to eliminating waste (Figure 5) in Lean Application Maintenance.

1. **Improve application changeability**: The maintenance team can identify the most serious structural flaws degrading the changeability of an application using diagnostics from structural analysis of the application, such as high coupling among components especially across application layers. Prioritizing and remediating the most severe changeability problems will reduce the time that maintenance staff spends trying to trace control and data flows through the code. Improving the changeability of an application not only slashes the time wasted on understanding unnecessarily complex code, but also shortens implementation and verification cycles. As a result, the amount of functionality delivered in a single release can be as much as doubled. In addition, as the ease of changing an application improves, fewer new defects will be injected into the code during maintenance by eliminating the misunderstanding of interactions among application layers. Lessons learned in improving changeability can be used in training developers how to produce more changeable architectures and code.
2. **Reduce defect-fixing releases**: IT organizations must too frequently plan releases devoted primarily to fixing defects in the application. The defects that take the longest to fix in maintenance are frequently structural, since such flaws are harder to detect through traditional testing. Implementing software analysis and measurement as a component of the development process helps dramatically reduce the number of structural defects that slip into operational software. By detecting and eliminating these defects before release, it is possible to reduce defect fixing in maintenance to near zero in a mature development environment. The percentage of maintenance effort devoted to implementing new business functions is dramatically increased as the releases of an application devoted to defect fixing are eliminated.

3. **Sustain lifetime quality**: To track and sustain the quality of an application throughout its lifetime, software analysis and measurement should be implemented as a standard component of the maintenance process. The structural quality of many applications degrades continually over a succession of maintenance releases, to the point that management has to weigh the tradeoff between the increasing time and expense of adding functionality versus the cost of redeveloping the application. The quality of an application can be sustained across its lifetime by measuring the status of various quality attributes during maintenance and remediating the quality problems created by maintenance actions, eliminating the need for extremely costly redevelopments.

All three of these aspects of Lean Application Maintenance involve reducing the technical debt that builds up in applications. Technical debt can be defined as the cost of fixing the structural problems in source code that are not fixed before release, but will have to be corrected so they don’t cause operational problems or degrade the maintainability of the application. Technical debt has been conservatively measured to cost $2.82 per line of code, although the actual cost based on more realistic assumptions about time to repair defects is easily twice as high. Applications appear to differ in their technical debt based on the language in which they were implemented (Figure 6). Structural analysis and measurement provides executives with a way to measure the opportunity for cost reductions from applying the practices of Lean Application Maintenance.
IV. Lean Application Assets

Lean processes work at their most efficient when performed on Lean products. The best way to reduce waste is to design it out of the application from the beginning by driving design decisions based on prioritized quality goals. However, even when designed well, applications age and can become less responsive and obese with code that is no longer needed, wasting machine resources. This excess is especially acute in a cloud environment where the application owner is being charged for both space and processing time.
There are three keys to creating Lean Applications that eliminate waste (Figure 7).

1. **Design for prioritized quality characteristics**: The principles of Lean Product Development and Design for Six Sigma both encourage applications that are designed with quality priorities in mind. Quality characteristics such as Robustness, Changeability, Security, and Performance Efficiency should be prioritized for each application at the beginning of its development. Architectural and coding decisions should maximize the most important quality characteristics. These quality characteristics can be measured and analyzed during both development and maintenance to ensure the quality of the source code is consistent with its quality priorities. Architectural and coding lessons from an IT organization’s application experience related to these characteristics should be captured for consideration when making future design and coding decisions, since the most effective way to eliminate waste is to design it out of the application from the beginning.

2. **Tune the performance of application code**: Poorly designed and coded applications can waste expensive machine resources by requiring more processing than necessary. Structural analysis should be applied to detect poor coding and architectural practices that can dramatically affect processing requirements and time. Structural problems such as expensive calls inside
Highlights

An application's code base should be evaluated at each release to detect code that is no longer needed.

loops, failure to properly index queries, and embedding functions inside ‘WHERE’ clauses can dramatically increase processing loads, requiring the acquisition of additional machine resources and slowing response times. In one example, a major multinational financial services company that started proactively measuring and remediating performance problems in their applications saw processing expenses reduced by roughly $10,000 each month for a single application. Performance issues are especially acute in cloud environments where such problems can result in unexpectedly large bills.

3. Manage application size: The size of an application should be continually monitored to ensure its growth is under control and fits within parameters appropriate for its environment. Automated measurement of code size can be done in both function points and lines of code to indicate the amount of code added, changed, or deleted. When these measures indicate that code growth exceeds allowable thresholds, corrective actions should be triggered. Further, the code base should be evaluated at each release to detect code that is no longer needed. Dead code throughout the application can then be scheduled for removal in an upcoming release. Removing dead code ensures that application growth and the machine resources needed to support it are consistent with the business requirements of the application. In addition to tracking application growth, these size measures will be important for evaluating maintenance productivity, as will be explained in the next section.

V. Lean Application Management

Establishing a Lean IT organization requires rapid and continual feedback on the quality of the applications being placed in operation. The combination of Lean Application Development, Lean Application Maintenance, and Lean Application Assets provides the foundation for Lean Application Management (Figure 8), a critical step in turning an IT organization into a Lean Enterprise. By aggressively pursuing the Jidoka component of Lean, the analysis and measurement of applications provides a critical ingredient required to reduce many sources of waste associated with application development and maintenance.
The ability to continually monitor application productivity is another critical component of Lean Application Management. Productivity data provides insight into the effectiveness of Lean practices at the application, portfolio, and delivery center level. It is important to choose a consistent measure for the numerator of productivity measures – that is, for the size of both the application and the actual work accomplished during each release. When these size measures are fully automated, they can be evaluated consistently across all applications and used for comparisons across an application portfolio or between different delivery centers. Consequently, the benefits of Lean practices and other improvement activities can also be assessed at the application, portfolio, or delivery center levels. These analyses provide the insight needed by application executives to target troubled applications or delivery centers, and allocate the resources required to optimize their cost structure and service to the business.

Software analysis and measurement can be easily integrated into a disciplined software life cycle process. In an undisciplined environment, diagnostics can be used to motivate disciplined work by holding developers accountable for the quality of the applications they produce. In either environment, automated analysis and measurement of structural quality of applications is critical to the overall cost of ownership, business responsiveness, and operational dependability required to run a Lean Enterprise.
References

Bill Curtis is an industry luminary who is responsible for influencing CAST's scientific and strategic direction, as well as helping CAST educate the IT market to the importance of managing and measuring the quality of its software. He is best known for leading the development of the Capability Maturity Model (CMM) which has become the global standard for evaluating the capability of software development organizations.

Prior to joining CAST, Dr. Curtis was a Co-Founder of TeraQuest, the global leader in CMM-based services, which was acquired by Borland. Prior to TeraQuest, he directed the Software Process Program at the Software Engineering Institute (SEI) at Carnegie Mellon University. Prior to the SEI he directed research on intelligent user interface technology and the software design process at MCC, the fifth generation computer research consortium in Austin, Texas. Before MCC he developed a software productivity and quality measurement system for ITT, managed research on software practices and metrics at GE Space Division, and taught statistics at the University of Washington.

Dr. Curtis holds a Ph.D. from Texas Christian University, an M.A. from the University of Texas, and a B.A. from Eckerd College. He was recently elected a Fellow of the Institute of Electrical and Electronics Engineers for his contributions to software process improvement and measurement. In his free time Dr. Curtis enjoys traveling, writing, photography, helping with his daughter's homework, and University of Texas football.
About CAST

CAST is a pioneer and world leader in Software Analysis and Measurement, with unique technology resulting from more than $90 million in R&D investment. CAST provides IT and business executives with precise analytics and automated software measurement to transform application development into a management discipline. More than 650 companies across all industry sectors and geographies rely on CAST to prevent business disruption while reducing hard IT costs. CAST is an integral part of software delivery and maintenance at the world's leading IT service providers such as IBM and Capgemini.

Founded in 1990, CAST is listed on NYSE-Euronext (Euronext: CAS) and serves IT intensive enterprises worldwide with a network of offices in North America, Europe and India. For more information, visit www.castsoftware.com.