LATTIX

Case Study: L.L. Bean

Challenge

L.L. Bean's development team had hit a wall. They had embraced the Agile methodology to handle the rapid enhancement and maintenance of their retail technology infrastructure. But over the past decade, L.L. Bean's software code base had become brittle, increasingly difficult to build, and difficult to enhance.

Despite the use of good software development practices and standards, normal code evolution created a complex entanglement of interdependencies. This caused increased software development and maintenance costs, and decreased reuse potential. On top of all that, multiple code bases diverged over time, which significantly increased complexity.

Solution

Lattix Architect was selected to improve the agility of the L.L. Bean code base. L.L. Bean liked the ability to scale and the ease with which alternative architectural structures could be explored with Lattix Architect. L.L. Bean found Lattix Architect offered a comprehensive, easy to understand interface, as well as mechanisms for prototyping and applying architecture rules. Lattix supported "what-if" analysis without code modification. All of this meant increased agility of the software development process.

Results

- L.L Bean reduced the size of their code base by 10%, which dramatically improved the understanding and maintainability of the application.
- Architectural governance reduced software maintenance cost, improved quality, and increased agility, by enabling architectural remediation during ongoing development.
- Change impact analysis was facilitated.
- L.L. Bean found that increasing the visibility of the software architecture greatly reduces architectural drift (or erosion) as the system evolves and at the same time reduces ongoing maintenance costs by facilitating communication.
- L.L. Bean found that architecture-based analysis improves the productivity of software development.

Background

L.L. Bean has been a trusted source for quality apparel, reliable outdoor equipment, and expert advice for nearly 100 years. L.L. Bean's software is used to manage its sales, which include retail stores, as well as online sales, inventory, and human resources. More than 100 architects, engineers, and developers work on continual improvement and enhancement of the company's information technology infrastructure.

For the last eight years, their infrastructure has suffered the typical problems associated with rapid evolution, such as increased fragility and decreased intellectual control. This resulted in increased difficulty building the system.

While the company's software development processes have long included a number of good practices and coding rules to help avoid these issues, in the end the speed of evolution overwhelmed the development teams. Maintenance and evolution of the software infrastructure were recognized as chief concerns by the company's upper management. Investigation into the core cause of the problems pointed to the fact that the code had become a complex entanglement of interdependencies. It was decided that the code must be restructured.

A significant part of L.L. Bean's information technology infrastructure is written in Java and runs on Windows, Unix, or Linux based servers. The system has gone through a rapid evolution over the last eight years due to several massive development efforts undertaken in response to increased demand from multiple business units. New front-end systems, strategic web site updates, regular infrastructure improvements, external product integration, and security have been among the key drivers. The current system has more than one million lines of code assembled into more than 100 JAR files. In turn, the actual code is organized into nearly 1,000 Java packages and more than 3,000 Java classes.

Requirements

- Decrease testing and maintenance costs by eliminating interdependencies
- Make the architecture visible to everyone in the organization
- Govern the architecture of software development to prevent architectural drift (or erosion) and negate the need for large scale refactoring in the future
- Create a clear and detailed understanding of the existing static dependencies
- Provide a single, consolidated code base to support a producer/consumer paradigm
- Dynamically generate a view of the interdependencies of deliverable software assets
- Minimize the cost and effort required to compile and assemble deliverable software assets
- Increase the level of reuse by increasing communication between development teams
- Find a comprehensive, easy to understand global view of the architecture
- Provide support for restructuring and communication among various stakeholders, including IT managers, architects, and developers

Lattix Architect

Lattix has pioneered an approach using system interdependencies to create an accurate blueprint of software applications, databases, and systems. To build the initial view, Lattix Architect was pointed at a set of Java JAR files. Within minutes, the tool created a Dependency Structure Matrix (DSM) that showed the static dependencies in the code base.

Lattix Architect DSMs have a hierarchical structure, where the default hierarchy reflects the JAR and the package structure. The Lattix Architect DSM offers partitioning algorithms to group and re-order subsystems. The result of this analysis showed the layering of the subsystems as well as the grouping of subsystems that are coupled through cyclic dependencies.

This approach to visualization also overcomes the scaling problems that L.L. Bean encountered with directed graphs. Furthermore, Lattix Architect allows users to edit system structures to run what-if scenarios and to specify design rules to formalize, communicate, and enforce architectural constraints. This means that an alternative structure, which represents the desired architectural intent, can be manipulated and maintained even if the code structure in not immediately a true reflection. Once an architecture is specified, Lattix Architect allows that architecture to be monitored in a continuous integration or build environment. Key stakeholders are notified of the results.

About Lattix

Lattix is a leading innovator of software architecture management solutions that deliver higher software quality, accelerate development timeframes, reduce costs, and lower risk throughout the application lifecycle. Lattix solutions have also been applied to a wide variety of complex systems that include software, hardware, activities/processes, and organizations. More information about Lattix can be found at http://www.lattix.com.

Understand, Define, and Control Software Architecture