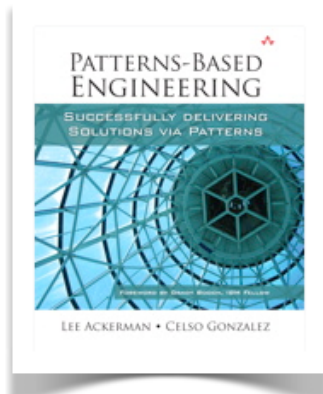


Powering the VelocityFactory™ with Patterns



Pattern Implementations as discussed in “Patterns-Based Engineering: Successfully Delivering Solutions via Patterns” are an important and proven type of Bot that we use within the VelocityFactory here at The Emphasys Group.

The following examples discuss real-life situations where pattern implementations were used and the impressive results achieved.

Services Team: Portlet Proof of Concepts

In the first example, a software vendor’s services team needed to build custom portlets to support proof-of-concept engagements. Before they used patterns, the work was performed manually, and it would take about 40 hours to create and test a custom portlet. In a proof-of-concept environment, this is unacceptable as it significantly increases the cost and time to complete the proof of concept. This was identified as an opportunity to build a pattern as there was an existing and recognized best practice that was used many times.

The pattern was responsible for creating a large number of files (approximately 95) needed for the portlet, code was added to handle common configurations and errors, and the pattern would also generate optimized and tested code. After the pattern was applied, the Pattern User would add business logic to the portlet. Using the pattern to build the portlet took approximately 20 minutes, and the majority of that time was spent on adding the business logic and then deploying the portlet. The pattern was used over a number of years and resulted in over 80 portlets being generated, saving over 3,000 hours of development time.

Software Vendor: Product Update

The second example is based on a team working for a software vendor. The application that they worked on needed to meet a hard deadline for a customer and ship with a set of 130 SOA-based services that adhered to the OAGIS message format. A significant number of steps were needed to create each of the services, resulting in over 100 pages of documentation. The team discovered that performing this work manually would push them three months beyond the customer-imposed deadline.

The team spent a day learning how to create pattern implementations and then went to work applying those skills. The resulting pattern generated a significant portion of the required code, including data objects, client code, server code, and even unit tests. When using the patterns, the team was able to reduce the documentation from over 100 pages to 2 pages. They were able to save over 1,400 hours in their project and deliver the solution to the customer in time to meet the deadline. They also ended up shipping the patterns with the product, enabling customers and business partners to benefit from the patterns.

Entertainment Industry: Enhancing MDD

Let's take a look at a couple of additional examples that go beyond companies that sell and service software. First we'll look at a company in the entertainment industry.

In this case the organization was following an MDD approach. More specifically, the team used a UML modeling tool for creating models that would then drive their development. The models were used for documentation, and a manual effort was required to follow through and create the solution. The organization had concerns about productivity, quality, and governance. Some patterns were used, but only manually, and they needed interpretation in the transition to development. Development phases were too long, and transcription errors occurred when moving from the models to code. In addition, adherence to corporate architecture standards was inconsistent, and best practices were not always followed.

The organization performed an analysis of its development approach, models, and resulting artifacts. As a result of this analysis a set of patterns was identified and then delivered as pattern implementations. By using these patterns, it was estimated that the architects in the organization were able to achieve up to 50% improvement in productivity, resulting in millions in development savings. In addition, they were able to eliminate many defects from the resulting solution by using automation in place of manual efforts.

Government : Integrating Departments

The last example we'll look at in this section is based on work completed for a European government department. The department was required to externalize its data via SOA-based services so that it could be shared and integrated across the government.



Within the department there were multiple divisions, and each was expected to meet this requirement. The planned strategy was to use an enterprise service bus (ESB) to assist in externalizing the services and support integration and sharing. In creating a solution, the department required a development platform that all of its employees could use that would be simple, agile, and highly productive. In addition, they required a strong level of governance and high quality; each member of the team had to adhere to the corporate architectural standards.

During the analysis of the situation it was determined that supporting the integrations would take three days of effort from a highly skilled resource for each of the services. There were a significant number of services to create. Each of the services would be created according to a specific best-practice-based approach. Recognizing a recurring best-practice approach, they leveraged the work done in the analysis and recognized the opportunity to create a set of patterns to generate the required integrations.

Using pattern implementations, rather than performing the work manually, allowed each of the services to be completed in approximately ten minutes and required a much lower skill level. The organization expected to save over 50 person years of effort by the switch to using pattern implementations.

Get Started

Have a project in mind? We can help with [Consulting and Coaching](#), a [Velocity Assessment](#) or skill building via [Training and Workshops](#).

Give us a call or drop us a note to [start the discussion](#):

Email: info@emphasysgrp.com

Phone: 888.784.7759