**Centric Website Refresh – Site Content Framework**

Type: Case Study

|  |  |
| --- | --- |
| **Case Study Title** | |
| The site title is the main name for the case study. This could focus on the technology implemented, the strategy used, or the type of project. | Example: “End-to-end Business Diagnostic”, “Pharmaceutical Managed Care Rebate Project”, “Web Portal Development” |
| Agile Test Driven Development | |

|  |  |
| --- | --- |
| **Case Study Tagline** | |
| The tagline is a marketing-based summary of the service offering. Although, not required, the tagline benefit is to provide our customers with an easy to understand, quick sentence describing the service offering’s value. This is the key value message that our clients should receive. | Example: “Centric improves operational efficiency while planning significant business growth and complexity.” |
| A modern approach to Test Driven Development yields tangible benefits during development and paves the way for future production regression testing efficiencies. | |

|  |  |
| --- | --- |
| **Case Study Content: The Business Need** | |
| 1-2 paragraphs explaining the AS-IS client situation and why they needed our help. Define the problem/issue that existed at the client site and why they contacted Centric. | Example: “The client leadership team knew its current operating model was not optimal and had collected various inputs on potential improvements. As they approached their annual business planning process, they recognized the need for a comprehensive assessment of its operations.” |
| **Business Need:**  A major Insurance client is undertaking the development of a new quoting application. The new platform is expected to have a lifetime of 15 years with ongoing and numerous enhancements.  A traditional design, build, test development methodology generating an unmanageable number of defects and an unpredictable schedule. Specifically:   * Traditional testing was prone to long regression testing cycles and a high defect density. The regression testing duration was difficult to estimate and required a large team to execute. * The time between defect creation and fix was too long – and defects found later in the development cycle are much more expensive to fix than those detected early in the cycle. * Far too many defects were finding their way to production. * Small seemingly simple code changes would take too long due to a lack of confidence in avoiding un-intended side effects.   The client recognized that once in production the resource required to “keep the lights on” would materially effect the amount of money that could be spent enhancing and expanding the quoting systems functionality. A more sophisticated Test Driven Development (TDD) approach was the solution. | |

|  |  |
| --- | --- |
| **Case Study Content: Centric’s High-Impact Approach** | |
| What approach/methods did we utilize to make the project successful? | “Centric led a team that identified key priorities for the client’s business plan, specifically the team worked to:  • Ensure strategic investments were applied to the appropriate business areas  • Identify and summarize external business drivers including the volume growth, competition, product complexity, regulatory, and performance expectations  • Assess implications of business drivers on organization  • Interview key customers to gather perspectives and insights around client strategy, products, services and delivery  • Review core business capabilities and assess strengths, weaknesses and opportunities, including business development / account management, service/capability development, service delivery and enterprise management support functions  • Summarize findings and prioritized opportunity areas” |
| **Centric’s Modern TDD Execution**  TDD is a software engineering ‘best practice’ where development teams write follow a RED-GREEN-REFACTOR workflow:   * RED – The developer writes a failing test essentially capturing the requirements in a test * GREEN – The developer implements the business functionality writing just enough code to pass the test * REFACTOR – The developer refines and improves the code without adding new functionality.   While TDD results in good test coverage and lasting regression test suite, it is also a software design and development technique that results in easy to maintain, loosely couple software.  **Approach**  Centric implemented and trained the project team on TDD techniques and tools for Unit and Integration Testing. Pair programming was used to match experienced TDD developers with those new to the practice. Our teams leveraged continuous integration to provide rapid feedback and detection code defects and to automate the application build process. A closer look at the implemented TDD process is provided below:   * The developer writes a failing test, writes the code to pass the test and refactors as necessary. * Once the developer completes the business features, all of necessary unit or integration tests are executed. * A peer code review is performed starting with the tests. This review process helps ensure that no requirements were overlooked and the code meets the team’s design standards. * The code is then committed to the source control management system and is merged into the code base. * The continuous integration server detects the code commit and begins its compilation workflow.   + The latest source is retrieved and compiled from the source control management system.   + All Unit and Integration tests are executed and the results are published.   + The compiled source is deployed to a development/staging area. * Finally, to make the overall process transparent and visible a custom-built dashboard displays the results of the entire process giving the development team immediate visibility into build quality.   **Toolset**  The client predominantly worked with the Microsoft .NET Framework utilizing Team Foundation Server (TFS). TFS fills the Source Control Management (SCM), Continuous Integration (CI) and Work Item Tracking roles. The TDD toolset needed to be compatible with .NET and TFS’s SCM and CI tools.  While Microsoft has a Unit Testing framework (MSTEST), the assertion patterns are not widely used outside of their ecosystem. There are two dominant and community supported TDD test writing styles, xUnit and Behavior Driven Development (BDD). The client already had some experience unit testing using both patterns so developers could choose the pattern to use case-by-case. NUnit was chosen for the xUnit style tests and MSpec for the BDD style tests. Both NUnit and MSpec have solid .NET implementations and were compatible with TFS and broad community support. | |

|  |  |
| --- | --- |
| **Case Study Content: Results** | |
| What was the outcome of the project and how the client felt about the success of the project? What were they able to do now that Centric has helped them? | “The project set the stage for a major Outbound Redesign initiative. All initiatives are projected to drive significant value by improving service levels, absorbing future growth and optimizing cost structure. |
| **Automated Regression Testing Benefits**  After implementing TDD the client has experienced a transformation of sorts resulting in the following tangible benefits:  *Code Coverage*  There are over 15,000 coded Integration and Unit tests executed every time new code is committed Defects for well-tested features are found soon after they are created. New code written has close to 80% code coverage. Existing code’s tests are continually updated maintain or expanding the 80% code coverage goal.    *Fewer Defects*  Our client’s historic code promotion process did a great job of capturing production defects. Comparing traditional development efforts with the new TDD powered effort has resulted in a much lower number of defects making it into production.  *Culture*  The transparency of the build and code review process has created a culture of ownership. Developers go to great lengths to not be the person that ‘breaks the build’. Also, knowing that their work product will come under peer scrutiny more time has been spent in design resulting in a better product.  *Productivity*  Productivity enhancements have been numerous:   * Because one side-effect of TDD is a solid set of regression tests, developers are refactoring with confidence knowing that if they create new defects they will be caught the next time the test suite is executed. Team members also know that if there are unintended side effects of their code it will be found quickly. * TDD done right improves the design. Well-designed software is easier to modify and maintain. And for this client, TDD has actually helped reduce the Agile team’s average cycle time of completing feature cards. * Developers do not have to ‘site-read’ the code to divine it’s intent as the intent is easy to read from the tests. * New team members are productive earlier as their mistakes can be exposed via the test suite. Also, it is easier to mentor the new team member via examination of their tests. | |

|  |
| --- |
| **Graphics / Illustrations / Videos to Feature on Page** |
| Include images/graphics/diagrams of key concepts to feature with the content on this page. Please paste graphics into this document or feel free to mail to JB Woodruff ([jonathan.woodruff@centricconsulting.com](mailto:jonathan.woodruff@centricconsulting.com)). Do not worry about formatting the picture or where it will reside. The website build team will complete this for you. |
| Write Failing Test  Write just  enough code  to pass test  Optimize,  Clean up without adding new functionality, and with confidence  Testing  Pyramid |

|  |  |  |  |
| --- | --- | --- | --- |
| **Case Study Relationships** | | | |
| *Mark the related Service Offerings, Industry and Business Unit with an “X”* | | | |
| Service Offering | | | |
| Customer Strategy Service |  | Application Development | x |
| Product and Service Innovation |  | Application Lifecycle Management | x |
| Business Process Improvement |  | Business Intelligence |  |
| Business Process Management |  | Cloud Computing |  |
| Supply Chain Management |  | Enterprise Architecture |  |
| Sourcing and Procurement |  | Portal Solutions |  |
| Service Operations |  | eCommerce |  |
| IT Operations |  | Mobility Solutions |  |
| Finance & Back Office |  | Agile Approach | x |
| Business Analytics |  | Application Outsourcing |  |
| Strategic Planning |  | IT Strategy |  |
| Business Architecture |  | Microsoft |  |
| M&A Integration / Divestitures |  | Oracle |  |
| Change Management |  | Rippleware | x |
| Performance Management |  |  |  |
| Enterprise Program Management |  |  |  |
| Organization / Leadership Development |  |  |  |
| Industry | | | |
| Financial Services |  | Manufacturing / Products |  |
| Government |  | Retail |  |
| Insurance | x | Utilities |  |
| Logistics |  | Healthcare |  |
| Business Unit | | | |
| Boston |  | Indianapolis |  |
| Chicago |  | St. Louis |  |
| Cincinnati |  | Tampa |  |
| Cleveland |  | ERP / Oracle |  |
| Columbus | x | Energy & Utilities |  |