**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 now as well as future 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:**  The client is in the process of migrating to a new quoting platform. The new platform is expected to have a lifetime of 15 years with numerous enhancements throughout. The current processes resulted in many defects and an unpredictable schedule.   * The process was prone to long regression testing cycles that turned up too many defects. This regression period was difficult to estimate and required a large number of manual testers to support. * The time span between when defects were created and fixed was too long. * 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. * Plagued by poor design and sloppy implementation that led to re-work.   The client preferred to invest resources in driving business value by building new features rather than keeping the lights on. The client knew that there were development approaches that led the development teams to build tests as development takes place and that those tests could be run together often. | |

|  |  |
| --- | --- |
| **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**  Centric was tasked with assisting the client in modernizing their development practices to address the pain they had been experiencing. Centric chose to use Test Driven Development.  Test Driven Development is a software engineering ‘best practice’ where development teams write follow a RED-GREEN-REFACTOR workflow.   * RED – The developer writes a failing test essentially encoding the requirements in a test * GREEN – The developer writes just enough code to pass the test * REFACTOR – The developer makes any adjustments or code cleanup to improve the code without adding new functionality.   While TDD results in good test coverage and an ad-hoc regression test suite, it is primarily a software design/development approach to create easy to maintain, loosely couple software.  **Approach**  Centric implemented and trained the current staff on TDD techniques and tools for Unit and Integration Testing. We used pair programming to match experienced TDD developers with those new to the practice. Our teams leveraged continuous integration to provide a rapid feedback mechanism of problems with code and to automate the integration and build pipeline.  Centric trained the developers in the RED-GREEN-REFACTOR cadence augmented with the continuous integration practices.   * The developer writes a failing test, writes the code to pass the test and refactors as necessary. * Once the developer completes the features, she run all of the necessary unit or integration tests. * She then has the code reviewed by a peer starting with the tests. This review process helps ensure that no requirements were overlooked and the code meets the team’s design standards. * She then commits the code to the source control management system and it is merged to the code base. * The continuous integration server detects the code commit and begins its 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 development/staging area.   We also wanted to make the entire process transparent and visible. We built a dashboard that provided visibility into the entire build process and gave immediate feedback from builds with failing unit tests.  **Toolset**  The client predominantly worked with the Microsoft .NET Framework. Their development process heavily utilized Microsoft 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) built into their product line, 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 one year of using TDD the client has experienced a transformation of sorts resulting in tangible benefits.  *Code Coverage*  There are currently more than 15,000 coded Integration and Unit test that are part of the test suite. These tests are executed every time code is committed. Defects for well-tested features are found soon after they are created. New code written in that timeframe has close to 80% code coverage. Existing code that has been modified during this development cycle is having tests added as the code is changed.    *Fewer Defects*  The client was very good at tracking defects. Comparing the same product to the previous version at the same time in the lifecycle, showed 400 fewer defects entering production.  *Culture*  The public build transparency and the 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*  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 suite is ran. 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. In this instance TDD has helped reduce the average cycle time of feature cards on the client’s teams.  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 instruct the new team member by examining the tests.  Other benefits:  Teaching? New team members can be oriented by looking after regression test?  Developers challenged to not have their code break build?  Developers challenged to have their tests catch issues?  Compliance?  Automated gathering of build quality metrics? Build quality dashboard?  (like these benefits, but left me wanting more)   * Side-effect of a regression test suite - ? * Requirements are encoded in tests - How * Significantly fewer defects – benefit, would like to know more, why is there significantly fewer defects? * More confidence in making changes – benefit, how did this manifest? People work quicker? Less hesitation to refactor? * Better design in many places – benefit, would like to know a tad more * Reduced gold plating (don’t know what this is) * Over 400 fewer defects compared with earlier development cycles – benefit, great metric, but would like to know how the comparison was made   Questions to answer here:   * why did we choose cucumber, water, ruby as opposed to msft standard tools? * What specific technologies did we use (want details) * Describe how our tdd works (feature card, build test using…, build code for function, check in code, check in test code, build, each night build runs, each hour build runs, in morning see report that illustrates what broke, etc. * Current metrics, currently have over xyz tests testing approximately abc lines of code (may be better metrics out there, but these popped to mind, don’t have to use these metrics) * Some metrics would be nice here – how many tests run every night as part of regression test. What’s our code coverage? How many people currently look after regression test * Did we do anything to automate the creation / deployment of standard set of test data? That get’s reset after tests run? | |

|  |
| --- |
| **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 |  |