Stitch: A Release Management Application

Jeffrey Moore
Grand Valley State University

Follow this and additional works at: http://scholarworks.gvsu.edu/cistechlib

Recommended Citation
http://scholarworks.gvsu.edu/cistechlib/169

This Project is brought to you for free and open access by the School of Computing and Information Systems at ScholarWorks@GVSU. It has been accepted for inclusion in Technical Library by an authorized administrator of ScholarWorks@GVSU. For more information, please contact scholarworks@gvsu.edu.
Stitch: A Release Management Application

By
Jeff Moore
April, 2014
Stitch: A Release Management Application

By
Jeff Moore

A project submitted in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Information Systems

at
Grand Valley State University

April, 2014
# Table of Contents

Abstract ................................................................................................................................. 4  
Introduction ............................................................................................................................. 4  
Background and Related Work .............................................................................................. 5  
Program Requirements ......................................................................................................... 6  
Implementation .................................................................................................................... 10  
Results, Evaluation, and Reflection ....................................................................................... 12  
Conclusions and Future Work ............................................................................................... 12  
Bibliography .......................................................................................................................... 13  
Appendices ........................................................................................................................... 14  
  Appendix A ........................................................................................................................... 14  
  Appendix B ........................................................................................................................... 15  
  Appendix C ........................................................................................................................... 16  
  Appendix D ........................................................................................................................... 17
Abstract

The Stitch web application is a custom tool for use by the Release Management team at Amway to plan and track software releases and infrastructure projects. There are many systems used at Amway to track feature development and infrastructure changes across applications. The goal of this application is to “stitch” all of the disparate data into one system allowing our release team a common interface into all changes associated with a release. In addition, a calendar view allows for visibility into when and where events are scheduled to occur, providing an opportunity to detect conflicts. Stitch was developed using the .NET MVC 4 framework, separating business logic, data, and presentation elements, with code stored in a Git version control repository and tracked in the cloud using Microsoft’s Visual Studio Online.

Introduction

Buildmeister.com defines Release Management as:

“The development of software applications is an evolutionary process, moving towards some predetermined end goals. These goals are usually in the form of a Release, either internal or external, to deliver a set of required functionality. Software Release Management is the process of ensuring releases can be reliably planned, scheduled and successfully transitioned (deployed) to Test and Live Environments. Software Release Management is not just about "automating the path to production" although that is certainly an important part. It also about adopting a holistic view of application changes, using the "Release" as the container to ensure that changes are packaged, released and tested in a repeatable and controlled manner. Release Management is often likened to the conductor of an orchestra, with the individual changes to be implemented the various instruments within it. Software Release Management is intrinsically linked with the more well understood and adopted Software Change and Configuration Management disciplines.”

Managing software application releases can be a challenging task. The responsible team must implement processes that deliver fixes and new functionality to end users quickly while also maintaining quality. The challenge can vary depending on the scope of the application, the user base, and the technologies on which it is built and integrate. Add that with the use of different tracking systems per dependent application, and things start to get messy. Some applications have sophisticated release processes while others are very rudimentary. For any given application, a release team finds itself being asked questions such as: When will a bug or set of new features be live in production? What functionality is contained in a release? Is there anything else scheduled during the release launch? When is the best time to schedule a change with the least impact to the end user? A change can be defined as, “any activity that is
physical, logical, or virtual to applications, databases, operating systems, networks, or hardware that could
impact services being delivered” (Kim 81). Without a schedule to work towards and without proper
tracking, these questions prove difficult to answer. Stitch aims to simplify attaining these answers by easily
and conveniently displaying this data to the asking party. The main goal of the application is to present the
release team with the release dates for all tracked applications, their dependencies, associated bugs and
features, and other scheduled changes in a common interface so the team can quickly get the information
they need and make important decisions based on the data.

Background and Related Work

About a year ago, an effort was made to reuse and consolidate processes that were previously
independent. One outcome of this effort was a corporate Release Team, tasked with streamlining
deployment and release processes for all applications in our catalog. Currently, the release team oversees
100+ applications all with varying release schedules. There are also varying tools used to store schedules
and track releases. Some teams use SharePoint and some teams don’t use anything at all. For example, the
customer facing eCommerce website leans heavily on Microsoft Team Foundation Server for version
control, build management, and ALM (application lifecycle management). The website, project name
“Global Web,” integrates with several other systems. The backend system that places order information
leverages a home-grown application for work tracking and Subversion for version control. Another
example is the application that stores bonus data for the Amway Business Owners. This application uses
JIRA for tracking work, Subversion for version control, and Jenkins for build management. To further
complicate things, production incidents, or outages, are tracked in another system, and any integration with
Amway Affiliate systems may result in yet another tracking system. Very quickly one can see how many
systems a Release Manager would need to interface with to get a clear picture of what is changing and
when. What is the solution?

The obvious answer is to have all teams using the same system. However, that is not practical for
a number of reasons such as cost, customization, time, and knowledge. Having been an Environment
Manager responsible for build processes and deployments over the past few years, I could see just how
painful tracking a release could be for the Release Managers. Projects and applications that do not have
strict release processes or guidelines often result in either development handling deployment or throwing it
over the fence for the release team and operations teams to figure out how to drag it over the finish line. In
the former, our production environments can become non-standard, cobbled together environments while
the latter leaves the deployment and operations teams discouraged and wary of the solutions going into
production. Neither of these scenarios is ideal. It is much better to have a standard deployment process
(hopefully automated) in which the release managers know exactly what is moving into each environment
and when. This allows them to communicate effectively to the QA Teams, Business Owners, and
ultimately the end users. If the important version details of each release are hard to attain, communication
breaks down, leaving the testers and business owners with unclear expectations. I decided that an application that could pull data from each source system into a common place would be very helpful. As my capstone project was just about to start, this would be a good opportunity to learn something new while providing value to my co-workers.

There are at least four formal Release Managers on the Deployment team. The Release Managers are my project stakeholders. Each Release Manager is responsible for several different applications, some very big in scope and others much smaller. As the team works to align processes as best as they can, they must communicate amongst themselves to ensure deployments are not stepping on each other, especially in the cases where applications are strongly coupled. Wouldn’t it be helpful to have a common calendar that displayed each application’s releases? Wouldn’t it be even more helpful if this calendar took advantage of existing data stores with schedule information and pulled it into the common calendar without manual intervention?

There are several industry tools available to assist with release and deployment. The problem with these tools is the core problem in need of solving at my company – the data needs to be in the same system, or the same few systems. The number of different systems and tools our teams are using to record release data is what lead me to develop a custom application that can pull it all together.

**Program Requirements**

As I began bouncing ideas off of the Release Managers we came up with a basic set of requirements for the application. We decided to focus on a small number of systems and applications to track to begin with, with the long term goal of adding all applications. The initial requirements are as follows:

- Provide a common interface to important release data
- Release data includes
  - Significant dates (code complete, QA release, Production release)
  - Work Items (Tasks, Bugs, Features)
  - Dependencies (Release Artifacts)
  - Projects (A single release may contain multiple projects)
- Provide landing pages for major programs (Global Web, CMP, JDE)
  - Programs can be containers for several applications
  - Program landing pages should include high level release information specific to the selected Program including Active and Upcoming Releases with dates and a calendar
  - From the Program landing pages, users can access Release Detail pages with associated release data
- Provide links to source systems (TFS, SharePoint, Quality Center)
- Provide visibility into all applications (Only 2 in scope for the initial launch)
- Provide an event calendar with filters (Program, Environment, Event Type)
- Provide the ability to drag calendar events when dates change
- Provide the ability for Environment Managers to initiate code builds

Working from these requirements, the application began to take shape. Below are the major screens and features of the Stitch Web Application. I went with a very simple layout, stressing the importance of quickly seeing release data.

The Stitch home page displays the active and upcoming releases with their type, release code complete date, and scheduled release dates to production. PS refers to Product Set and loosely correlates to a region.

For example PS1 hosts applications for North America.

Clicking on a Program link takes the user to a landing page that displays release information for that program, complete with a calendar. The release data shown in this page is being pulled from Microsoft Team Foundation Server. In TFS I created Release Work Items for the Release Managers to set dates and
move through a workflow as the release moves through Planned, Active, and Closed states. Moving an event on the calendar will update the source system with the new date.

There is also a link to create a new calendar event. This is useful for tracking changes outside of a release. Setting the Program to Global Web in this example ensures it will be tracked with other Global Web changes. There is also the ability to filter the calendar by Environment.

Clicking a Release link will take the user to the release details page.

**GW.04.08.00 Release Details**

<table>
<thead>
<tr>
<th>Type</th>
<th>Release</th>
<th>PS1</th>
<th>PS2</th>
<th>PS3</th>
<th>CCD A</th>
<th>CCD B</th>
<th>CCD C</th>
</tr>
</thead>
</table>

**Open Bugs**

- 165562     AAadmin - no content label for Order Management utility function
- 162632     Content issue - payment gateway description of method of payment not mentioned (Malaysia)
- 186537     PRW - Metrics Report Has No Data
- 159169     App Offline Not Redirecting to appropriate Offline Page
- 160580     AAadmin - no content labels for Order Management utility functionality
- 162730     COPY OF HOTFIX - Net Order Volume Recipient - able to volume down to any ABD
- 159450     Missing payment method in auto-renewal online for Singapore

**Artifacts**

- Payment Gateway Database Schema Artifacts
- Payment Gateway Database Data Artifacts - PS2 Only
- Order Management Amin Link
- PS2-101 New Payment Method Types
- Payment Cache Policy
- New Product Search stored procedure
- SOA changes related to Product Service

**Projects**

- SEA Payment Gateway Implementation
- Other
- CMF (Clarify Replacement)
- North America
- "View All" link on Array.com
- North America
- Update to MyPod Assessment
- North America

This page displays the associated Bugs, Artifacts, and Projects. The bugs and artifacts link to the source system. Artifacts give the release team additional information about dependencies associated with the release.
In addition, the Release Manager can also see Open Tests that are executing for the release.

<table>
<thead>
<tr>
<th>Open Tests</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>161266 BR_MY_Content Check_Crawl of Facing Site_VIS_EN_DB_WWW_MAN</td>
<td>Active</td>
</tr>
<tr>
<td>161264 BR_KR_Content Check_Crawl of Facing Site_VIS_KO_DB_WWW_MAN</td>
<td>Active</td>
</tr>
<tr>
<td>161285 BC_MY_MID - Validate Personal PVBY Details - Current Month_ISO_EN_DB_MB_WBB_MAN</td>
<td>Active</td>
</tr>
<tr>
<td>161281 TC_US_Content Check_Crawl Admin Site_INT_EN_OBJ_WWW_MAN</td>
<td>Active</td>
</tr>
<tr>
<td>161259 EC_MY_Checkout Mgmt - Payment Options - Stored Credit Card_IBO_EN_DB_WWW_MAN</td>
<td>Active</td>
</tr>
<tr>
<td>161270 BC_MY_View Standard Filters in the LOS: Distributor and Contact List_IBO_EN_DB_WWW_MAN</td>
<td>Active</td>
</tr>
<tr>
<td>161280 BR_US_Content Check_Crawl of Facing &amp; Preview Sites_VIS_EN_DB_WWW_MAN</td>
<td>Active</td>
</tr>
</tbody>
</table>

Left: Bug displayed in TFS

Below: Artifact in SharePoint

<table>
<thead>
<tr>
<th>Artifact Name</th>
<th>EPayment Gateway Database Schema Artifacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discipline</td>
<td>Database</td>
</tr>
<tr>
<td>Dependency?</td>
<td>No</td>
</tr>
<tr>
<td>Deployment Instruction</td>
<td>Please execute the attached scripts to update the database</td>
</tr>
<tr>
<td>CMP Build</td>
<td></td>
</tr>
<tr>
<td>Release-DCS</td>
<td>GW.04.08.00</td>
</tr>
<tr>
<td>Project</td>
<td>SEA ePayment Gateway Implementation</td>
</tr>
</tbody>
</table>

From the release details page, the Environment Manager can initiate a code build for the release.

**The Build has been queued. Check the status at this link:**

GW.04.08.00.00 2014.04.15 1001
Lastly, the Calendar page displays events for all programs.

**Event Calendar**

<table>
<thead>
<tr>
<th>Month:</th>
<th>Previous</th>
<th>Next</th>
<th>Create New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuesday</td>
<td>April 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wednesday</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thursday</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Friday</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>May 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**Implementation**

Although I have experience with C#, I chose to learn and build this project in the .NET MVC4 framework using the Visual Studio 2013 development environment. MVC provides a great out of the box framework for rendering different views based on the model state and parameters passed to the controllers. I separated the logic needed to retrieve data from external sources into a Services project and referenced that from the Web project (see Appendix A). Thus, the web project contains the models, views, and controllers of the MVC framework, with the controllers making calls to the services layer to interact with
outside systems. The controller then decides which view to render based on the incoming request. I ended up with the following main components: Release, Program, Calendar, Build, TFS, and SharePoint. The TFS, SharePoint, and Build components handle requests from the browser and interface to the services layer to retrieve and update data, and execute build requests. The Release and Program components handle aggregating release and program information. The Calendar component displays event data and is displayed as part of the Program information, as well as on its own page.

The calendar was built using Entity Framework with code first and migration practices. Entity framework allows the developer to create and update databases from code classes. In this case, properties in the Event (Model) class map to columns in an Event table in a SQL database. Enabling migrations allows for modifications to the model by generating code files to both update and rollback the database. These files remain part of the solution so the developer has a record of changes to the database. To learn the MVC and Entity Framework concepts, I relied on the ASP.NET MVC 4 tutorials (www.asp.net/mvc/tutorials/mvc-4). The calendar UI comes from a third party library which uses AJAX for updating calendar events (“DayPilot for ASP.NET WebForms”).

I stored the project source code in a Git version control repository hosted on Microsoft’s Visual Studio Online. VS Online became my tool for tracking the project work through product backlog items, tasks, and bugs (see Appendix B). From Visual Studio, I could associate each commit to Git with the task I was currently working on. This method of tracking made sure I did not forget anything. When I got to the deployment phase of the project, I leveraged the cloud hosted build controller to compile the solution into a format ready for release.

Finally, I worked with an Amway Systems Administrator to deploy the Stitch application to both a QA and Production environment using automated scripts (see Appendix D). I also worked with an application Database Administrator to promote the SQL database from my development environment to QA and Production. During the deployment phase, I had to ensure that my application was built with all necessary dependencies and provided the correct configuration so that it would run in whatever environment it was deployed to. This included creating a common user account for the site to run under, granting that user access to the database and other source data systems, creating a DNS entry for the QA
Results, Evaluation, and Reflection

I feel that I have gained a lot of knowledge in the new paradigms I chose to work in. I also think I have produced an application that adds value and makes life easier for those using it. A Release Manager, Jeff Miheve, made this comment while testing the application, “Stitch brings together data from multiple sources and presents it in a way that is helpful to the user. While Stitch does this automatically, it would take hours for a user to sift through the applications to find the data. If implemented completely, Stitch would save time and resources by streamlining the release process and give visibility to what was coming down the pipeline.”

I do wish I could have added more functionality, but due to time constraints and dependencies on other teams to allow me access to their systems, I was not able to complete as much as I had hoped. Since I am not a full time developer and only program occasionally, I was both satisfied with my accomplishments, and at the same time reminded of how much I do not know. That fact will continue to drive me to master current skills and learn new, unfamiliar skills.

Conclusions and Future Work

My goal for the Stitch application as of the capstone due date was to have Stitch available to review releases from two major applications. I also wanted to build integrations to at least two source systems. I accomplished both goals by integrating to both the TFS and SharePoint APIs to retrieve release data for the Global Web and Contact Management Platform (CMP) projects. In addition, I accomplished my personal goal of learning the .NET MVC framework. Next on the agenda is to add additional functionality:

• Add integration into other source systems such as HP Quality Center and JIRA to gain insight into Programs using those systems
• Add a “Release Quality” status to each release based on a set of criteria (open bugs, slipping dates, etc.)
• Add the ability to execute deployments after code builds are complete
• Make the system very scalable by providing the ability to configure which source systems it integrates with
• Integrate application with approvals in Remedy, the change management tool
• UX/UI Facelift
Bibliography


Appendices

Appendix A
Solution layout including web and services projects
Appendix B

Project Plan
Release 1
Sprint 1:
- Landing Page: Current Releases, Future Releases
- Release Page: Release Quality (derived), Projects, Bugs, Dependencies
Sprint 2:
- Program Page: Overview, Past, Present, Future Releases
- Calendar: Release Dates Across All Programs, Program & Release Views
- Release Flow View Page: Visual Representation of Release Status

Release 2
Sprint 1:
- Approval Integration: Remedy Approval?
Sprint 2:
- Build Integration: Queue Builds
- Deploy Integration: Deploy Code

Release 3
Sprint 1:
- Additional Programs (Applications)
- Additional Data Repository Integration

Backlog Item Work
Appendix C

Class Diagrams
Appendix D
Stitch Development and Deployment Process

VS Online, Git Repository

SANDBOX

QA

PROD