Pnambic Computing

Lessons from Restructuring a Website

Lee Carver

Executive Summary

In this case study, a legacy company website is restructured. This process exposes many useful lessons for all types of restructuring efforts.

Key lessons are:

- Product structure depends on components and their composition. Restructuring changes both the components and their composition into the delivered product.
- Restructuring is change, and change requires content management tools.
- Introduction of content management tools poses several challenges beyond those that arise from the restructuring work.

Key recommendations include:

- Recognize restructuring projects early, so systematic analysis and development processes can facilitate success of the restructuring work.
- An inventory of the product components is essential. Early in the project, this inventory will grow in detail and depth. Once the restructuring commences, the inventory can track progress and milestone completion.

Lessons from Restructuring a Website

Restructuring is often considered as a code related task. However, this task can arise for any composite entity. One alternative that offers instructive lessons about restructuring is website evolution and reorganization. Many of the challenges and lessons that occur in large restructuring projects can be seen more transparently in this small restructuring effort.

Although some of these observations may seem obvious, it is useful to gather all of them into one list. The scattered and incomplete natures of many discussions makes restructuring appear to be more of an art than a science. Advanced preparation for these challenges improves the chances for success in a restructuring project.

Restructuring a web site turns out be an excellent object lesson in the challenges faced by restructuring effort:

- The initial cleanup effort is not recognized as a restructuring effort.
- The initial proposal quickly runs into unanticipated complexity.
- Comprehensive inventories are essential for successful restructuring.

Restructuring means that the internal structure of a product will change. In order to manage these changes, restructuring efforts rely upon a multifaceted infrastructure. An essential part of that infrastructure is content management tools and their supporting systems.

For the website overhaul, the introduction of content management tools exposed the need for deployment, delivery, and site management operations. Although many of these operations remain manual, they are been identified and isolated. The separation of managed and delivered content also exposes the need to debug and test changes independently of release.

The discovery of these latent concerns does not change the inherent complexity of the product or its development process. Although these concerns may be newly recognized, the underlying activity has always been present. Perhaps ignored, or swept under the rug, but the roles and task remain regardless of official recognition.

1 In Over Your Head

I originally developed my company web page around 1996, during the early moments of the Internet big-bang. It had evolved over the years, but I did no significant work after getting a day job in 2004. I added some content from time to time, but even I had a hard time finding live links to the coolest pages. Since I was planning to be serious about consulting work, it was time to upgrade the site to a more modern feel.

My initial idea was to place the web site into a git repository. This has numerous benefits, especially for a coder geek like me. There were obviously the website pages themselves, and an old Web1.0 alumni registration application that I maintain (mostly to establish prior-art :-)). So, a couple of git repositories under the company name, and I would be all set.

Not really.

2 Storage and names

The first problem to address is the question of a content management provider. How and where is the definitive version of the content be stored? This implies that the refactoring project will include adoption of a content management system and some kind of external storage location.

Lesson: Restructuring requires content management. Plan to add this if it is not already present.

Adding content management to an existing project is both a technical and a political challenge. Even in the best of circumstances, different content management schemes should be reviewed and assessed. In large projects, gathering and meeting the content management requirements of multiple stakeholders presents additional challenges.

For the website restructuring project, I reviewed some website templates and creation tools. Although several of these were very powerful, I wanted manual control of the top level pages for the website. This means that the entry pages will be hand-crafted web-pages, and git is an appropriate content management system for this kind of content.

Often, the selection of a content management tool defines the storage for the content. With proprietary web site tools, this often means a locally hosted database or a directory. Multiple editing sessions often requires a centralized server or a cloud based provider. Unlike proprietary content management tools, there are a number of third party cloud-based git-providers. Although github is popular, I selected bitbucket due to its superior support for private repositories. Also, the group name for the company was available on bitbucket.

Lesson: Selecting and installing the content management tools is complex. The analysis must address stakeholder goals and review tool capabilities. Plan for this subtask if content management is not already present.

Before moving any content, the task of choosing a name for its repository remains. Although it could be titled for the company, or “web”, or “site”, each of these has some inherent ambiguity. How might these generic names conflict with future sites or projects in the same management domain? Since this content is a specific implementation of the web site, and that implementation is found in the public_html directory, that directory name was chosen for the repository name.

Lesson: Naming repositories requires consideration and forethought. Plan ahead, especially where approval is necessary.

How does the delivered content correspond to the managed content? If the git repository is configured inside the public_html delivery tree, then some of its artifacts (e.g. '.git') might be accessible to hackers. If the git repository is external, then a deployment process is needed to install the release bits into their delivery location.

Although it becomes an incomplete answer, I base the layout of the repository on the expected structure of the Maven site plugin. This calls for a src/site tree, with most static html and css located in a resources subtree. Although direct use of Maven's site plugin turns out to be infeasible, the overall structure is useful and sound.

Lesson: Introducing content management also exposes latent tasks for deployment and delivery. Plan to add support for these tasks if content management is also being added.

The decisions so far include:

- Name and location of the content service
- Name of the initial content repository
- Processes for deployment of release version to delivery platform.
- Placement (and structure) of content within repository

To implement these decisions, an empty git repository is created outside the public_html tree. Finally, we have a content management repository ready to receive its first bit of managed content. All that remains is to copy to contents of the public_html tree into the src/site/resources tree of the git repository.

Even before the first file is copied into the git repository, several new problems leap out:

- There is a lot more content than the top level web page. There are nearly 30 separate top level items, with 20 directories.
- There are several directories for personal homepages. Many of these are small, but one is huge.
- Various external components are embedded in the delivery site, including an alumni directory, research results, and a WordPress site.
- There is historical junk for stuff that has been abandoned.
- Some content looks like configuration files for external services. Maybe these are configurable, but I would not want to regenerate the web site without them.
- In several places, file system links are used to select between alternative content. This is especially common for index.html files.

These problems should not have come as a surprise. They are a surprise because the website overhaul effort failed to start with proper preparation for a restructuring task.

Lesson: The initial cleanup effort is not recognized as a restructuring effort. Therefore, the project gets in trouble when the predictable challenges of a restructuring effort arise.

This large set of entities reveals that the website is a composite entity. At the end of the cleanup effort, the website contents should be effectively unchanged. The content will be managed, but otherwise it should be delivered to users unchanged.

Lesson: It helps to be crisp about your goals.

One emergent goal was that it should be possible to regenerate the entire website from the version control repositories.

Lesson: One goal of any restructuring project is the regeneration of an end product.

Often, an effort is recognized as a restructuring project when the scale of the problem becomes evident. Scale presents challenges largely due to unknown issues hidden in the massive context.

When coupled with effective analyses, an inventory of the product's components will expose many hidden issues. There might still pose difficult steps, but simply executing all of these is sufficient for the final success of the restructuring project.

Lesson: An effective inventory for the restructuring project is essential to its success.

3 Website Inventory

The initial inventory exposed 27 top level items, and a total of more the 4900 separate items under the public_html tree. In addition, there were another 29 items in the public_ftp tree.

The website inventory also reveals that the content has several segments. In addition to the expected alumni application, one forgotten segment provides a separate bundle of research results. Although most of the user pages are small, and reasonably part of the main repository, one user area is very large. That user will require their own content management tools

Lesson: An inventory is likely to reveal many more segments and partitions than originally anticipated.

One goal of the inventory process is to properly categorize all of the elements to be restructured. This exposes hidden categories and properties. The website inventory exposed several important structural entities. The links, sub-trees, and configuration files each require special consideration within the content management tool.

Any time an entity has different ownership, alternate processing, or reflects a semantically unique kind of data the inventory should add a new category element. Categories should be expected for directories and files. Additional categories may be needed for system controlled entities, configuration files, and user managed content. An effective inventory may include a high-dimensional set of categories.

Lesson: An inventory is likely to reveal many more categories and types of entities than originally anticipated.

In addition to different types of data, ownership and control of the content varies across the website. The alumni system and the research results are managed independently of the primary website, but are accessible from the website's links. Similarly, the WordPress directory needs to be present in the delivery tree, but it has its own content management system. The design will need to accommodate externally managed content.

Lesson: Restructuring projects should include support for externally controlled components.

The directories for personal pages reveal an unexpected issue that is also present on the main page. The index.html file is often a symbolic link to another file on the page. In a few other places, links are used to control content delivery. The content is not changed, but the site's configuration defines which content is delivered for each request.

The use of links in the delivery tree exposes a distinction between content creation and content delivery. The delivery site is responsible for selecting the result content for each request. These links belong to the "site management" role, not the content creator role. Although this role is achieved through manual processes, the important distinction is its separate from content management.

Lesson: Introducing content management also exposes latent tasks for site management. Plan to add support for these tasks if content management is also being added.

For the website overhaul, a comprehensive site management solution remains elusive. For the time being, any links to switch content will be managed manually.

Lesson: Restructuring projects reveal technical debt.

Although the revealed technical debt is undesirable, that situation was demonstrably acceptable. There are often good engineering or economic reasons to leave the technical debt in place. For the time being, any links to switch content will be managed manually.

Lesson: Not all known or revealed technical debt will be eliminated at the end of the restructuring project.

Site management also means that the content repositories do not exactly match the delivered content. For the simple case of a default index.html file, that may not exist as a named entity in the content management tools. This comes up again, but it introduces the point that the development environment is not the delivery environment, and some accommodations for testing will be required. For the website overhaul, restricted access paths and manual intervention is adequate. Larger projects will insist on robust debug support.

This will also add some debugging challenges, since the link will not be present in the source tree. Regardless, debugging of a partitioned website is going to present challenges.

Lesson: Content development requires effective means to simulate site management choices that are not captured as artifacts.

Some review of the contents from the initial inventory revealed a number of dependencies on external entities. There is a public_ftp site that does not require immediate attention. However, several of the pages reference CGI cod in a separate Sites/ directory. The external dependencies do have implications for the refactoring project.

The content of the Sites/ directory is an integral part of some website features. Components from the Sites/ directory need to be included with the components extracted from the website. These lead to a more complex repository for the products, but it does provide a cohesive encapsulation of both the website's core and the alumni application module.

Lesson: Frontier analysis of external dependencies and components is essential for successful restructuring. It often exposes components that must be included in the restructuring.

Much of the HTML content had been written with SoftQuad's HoTMetaL Pro. Although I thought it was one of the slickest HTML editors of its era, you probably never heard of it. It was purchased and retired long ago. Other elements appear to be artifacts from Microsoft's FrontPage website development tool. Although it was an adequate HTML editor, the hosting site never supported its deployment mechanisms.

Lesson: You will be dealing with a heterogenous content, and artifacts from deprecated tools.

As we work through the inventory, a number of stop-gap solutions are exposed. There are several “page of silence” documents to commemorate past events. Clearly, the intent was to use symbolic links to temporarily route users through specific alternative. Whatever site management tools are used in the future, there is a need to insert temporary content into the website. And that temporary content needs to be associated with specific projects, not jumbled all together in a stew pop.

Once the content is migrated off of the delivery/deployment platform, use alternate deployment tools that are not available on the delivery platform (e.g. Maven, etc.).

Lesson: Some inventory items are outside the scope of content management.

4 Initial Encapsulation

In the end, the company’s portion of the website splits into three repositories: public_html, alumni, and cps_site. For each of these repositories, there are contents to place in separate containers. The http_public container receives contents for http delivery while the Site container receives protected implementation content and instance specific data.

After basic pruning, the public_html container has 21 top-level elements. Most of these elements go into the http_public repository. There are three top level components that are not assigned to any of the company repositories:

1. index.html - As defined by our delivery process, this element is controlled by the site management services, not the content management services
2. One user/ container - a large tree of user-based content. Since this primarily contains personal content (vacation pictures), this should be placed into a user-based repository, not a company repository.
3. One WordPress container - a container of independently managed content. This should have independent backup, etc. However, managing this content is not the responsibility of the restructuring task. We simply need to ensure that the composition process support heterogeneous content sources.

These last few elements appear to have straightforward resolutions, especially since the same kind of repository assignment can be used to capture the user tree. Installation of that user tree will require the same kind of overlay management that the company tree requires, but nothing new should be needed. It should be a simple repeat of a known process.

A similar analysis of the cleaned up Sites shows a similarly tidy result. The only trees are for the Alumni content and the company content, so the inventory shows complete coverage.

Lesson: As the restructuring proceeds, the inventory shows progress and completion of milestones.

Although the structural complexity of these repositories was a bit of a surprise, the basic scheme of a Maven-structured git repository seems to be sound. For these components in the inventory, the migration into a git repository worked satisfactorily.

The problem of updating the delivery platform remains, but that was a latent problem that is now emerging as an explicit requirement. This doesn't actually change the amount of work to do, it just makes it more explicit.

5 User content

Time to move on and capture the user content. Although this should be similar to the company content, it should not be placed under control of a company group. This requires definition of an additional repository group, and the creation of a repository within that group. Since the content of this repository is for delivery through the public_html site, the repository continues the public_html name for this aspect.

Despite the best of intentions and expectations, the encapsulation of the user/ container immediately runs into trouble. The user/ website has a structure that is completely different from the company website. The majority of the content is contained in scrapbooks, or collections for jpeg and other image files

The top-level user/ container has 35 items. Although there are 25 top-level items in the user/ container, these account for only a fraction of the overall content. 7 of the top-level items are "scrapbook" containers. These include images from several activities, mostly vacation photos and the like. These vary in size from a few hundred KB to a few hundred MB. Two of the scrapbooks exceed 100MB. These elements mostly contain pictures, and should not be placed into a git repository.

Lesson: In addition to an inventory of names, the analysis should include sizes for containers and elements.

These scrapbooks were created using an XSLT-based toolchain. The generator for each scrapbook is a bit of ad hoc shell and XSLT scripting, and these generators need to be located for properly managed content control. Additionally, their independent construction suggest that a unique project might be appropriate for each one. However, the time cost of finding and correlating each of the generator projects with each scrapbook pose immediate challenges.

Lesson(again): You will be dealing with heterogeneous content, and artifacts from deprecated tools.

Lesson(again): Restructuring projects reveal technical debt.

Regardless, the current scrapbook trees are inappropriate for adding to the user/ repository. They are simply too large. For each of these seven scrapbook containers, I simply bundle them into zip files and export them to an independent computer system. At least they are now separated from the delivery tree, and could be re-installed if needed.

Lesson: Different content can require different content management tools. This is especially visible when content management is introduced.

With these actions the encapsulation phase of the website overhaul is complete. There is a good bit of exposed technical debt, and the overhaul's goal of an improved website remains unrealized. But every content bit is now controlled by some content management tool. These could be better, but one of the initial goals is finally complete.

Lesson: The project is likely to end before all desirable goals are achieved.

6 Discussion

At the start, the website overhaul project was viewed as straightforward effort to control a small company website. Due to the unrecognized complexity of the actual website, this overhaul effort quickly turned into a restructuring project. The initial adoption of even primitive content management tools added further challenges.

It is not uncommon for some product engineering project to find itself transformed into a restructuring project. Since this is such a recurring pattern, it is valuable to understand the precursor to a restructuring project. If a team is looking to cleanup a section of code, the underlying problems may require a more extensive restructuring effort.

A systematic approach to restructuring is best when the need for restructuring occurs. It's too easy for analysis based on 20% of the code to fail when applied to the entire code base. Failure to properly prepare for a restructuring project often results in worse product structure. In the worse situations, this leads to the restructuring effort being abandoned after partial implementation. If this happens repeatedly, the code can become schizophrenic, with numerous partially implemented structures.

A systematic approach to restructuring includes several phases of component inventory analysis. Depending on the entry task, this may be a large increase in the scope of the original project. The scope if further increased if the restructuring effort introduces content management to the project. Properly planning for these tasks help ensure success of the entire project.

Lesson: Restructuring often starts with a noble idea that quickly runs into unanticipated complexity. Executing the task with a full understanding of the complexities and preparation for the challenges is much more likely to result in success.

The website overhaul project started with the expectation that the contents would wind up in two git repositories. In the end, 4 git repositories were used. Additionally, 7 zip archives define the contents for other components in the website. Overall, the restructuring work was 5.5 times larger than the original task anticipated.

Lesson: The project is bigger than you think.

Restructuring is multiple processes

Every composite product involves both component encapsulation and composition. The end goal of a development process is some encapsulation of the creative content that generates the product through the available composition tools. For code, we're often re-encapsulating modules that have well defined composition processes.

The tasks from the website overhaul confirm the distinction between encapsulation and composition. There is the initial encapsulation of the content into a managed container. This encapsulation exposes a latent need for a systematic composition process. This lack of a composition tool remains an unresolved challenge for the project, but I don't need a fully automated continuous release solution. Manual composition will have to work for the time being.

Introducing content management

Although the website was originally developed off-line, back then even a basic version control system for content management was not available. Over the years, changes have been made directly to the delivered content. This has been adequate given the website's historical usage.

In exchange for this "live" development process, the project lacks any practical form of content management. Experiments and roll-backs are difficult, and content loss may be irrecoverable. There is no practical means to rebuild the website if it suffered a catastrophic loss. The ability to reconstruct the website from managed content was an latent and essential goal for website overhaul effort. The introduction of content management allows restructuring to be successful.

The basic act of managing content creation separately from content delivery exposes latent operations for deployment and site management operations. Some deployment process is necessary to migrate each release version to the delivery platform. The delivery mechanism owns the process of content delivery, so site management operations are necessary to control runtime mappings of requests to content.

Content management might be as simple as archives and backups up large repositories, or it might be some form of version management. The essential point is to place the content external to the delivery mechanism. This allows it to be manipulated or changed independent of content delivery.

Inventory Analysis

An essential step of any restructuring effort is to create an inventory of the product components. This allows for effective decision making and avoids many common restructuring challenges. In early stages, the inventory of components is expected to increase in size and detail. In later stages, the inventory can show both progress and completion. The sooner a project recognizes the need for an inventory, the better its chances for success.

An effective inventory prevents a common “fluency illusion”, where a good idea is based on a sound understanding of the subset of the system. Because the chosen subset fits the proposed architecture, the proposed architecture is assumed to be universal. The solution appears to work because it has been fitted to the training data. Unless are restructuring proposal can account for 80% of the content, there are likely to be unanticipated failures in the restructuring plan.

The inventory starts with a list of all components that are known to contribute to the product. A partial inventory is a good starting point. An initial inventory should include at least the names and sizes of all components in the product, throughout the entire product hierarchy. As the analysis proceeds, the inventory grows through several analysis steps.

Category widens the range of information about each component. Every time an component has different ownership, alternate processing, or reflects a semantically unique kind of data, the inventory should add a new category element. Categories should be expected for directories and files. Additional categories may be needed for system controlled entities, configuration files, and user managed content. An effective inventory may include a high-dimensional set of categories.

Frontier analysis extends the scope of the inventory to include all direct dependencies on external systems. This includes all direct dependencies on external components, and those components. Frontier analysis recognizes the futility of understand the complete contextual universe, and makes the engineering decision to focus on a product’s immediate frontier.

Frontier analysis often reveals omissions in the scope of the project. Dependencies on one or more seemingly external components are so essential to a product’s behavior that any product restructuring must into the dependent external components. When this happens, the frontier of the included component needs to be included with the restructuring project. There are often administrative consequences or operationally visible interactions for these changes.

7 Recommendations

Not all projects involve significant restructuring, but it is import to recognize the need for restructuring early. This gives the team time to adopt structured restructuring practices. By following the lessons for previous restructuring efforts, a team greatly increases the possibilities for success.

Restructuring means change. Insist on effective content management before materially changing any content. Introduction of a content management system during a restructuring project is often required to ensure content management adoption and restructuring success.

Be crisp about the goals for the restructuring effort. One goal is regeneration of the product from its components. This goal to helps establish a sound criteria for successful completion of the effort.

One of the most powerful practices of structured restructuring is the inventory of components. Many kinds of feature and dependency analysis may be necessary. At a minimum, expect to track the complete component hierarchy through name, save, and category analysis. Use frontier analysis to address integration needs and to ensure proper scope of the restructuring effort.

Lessons Learned

The lessons here are collected directly from the white paper's text. They have been categorized into theme to improve their utility for future restructuring projects.

Engineering lessons: The sooner restructuring is recognized, the sooner systematic practices can be used to help it be successful.

- The initial cleanup effort is not recognized as a restructuring effort. Therefore, the project gets in trouble when the predictable challenges of a restructuring effort arise.
- It helps to be crisp about your goals.
- One goal of any restructuring project is the regeneration of an end product.
- Restructuring often starts with a noble idea that quickly runs into unanticipated complexity.

Content management lessons: Since the structure of the product is changing, restructuring requires attention to several content management issues. These issues are critical when content management is newly introduced to a project.

- Naming content repositories requires consideration and forethought. Plan ahead, especially where approval is necessary.
- Restructuring requires content management. Plan to add this if it is not already present.
- Selecting and installing the content management tools is complex. The analysis must address stakeholder goals and review tool capabilities. Plan for this subtask if content management is not already present.
- Introducing content management also exposes latent tasks for deployment and delivery. Plan to add support for these tasks if content management is also being added.
- Introducing content management also exposes latent tasks for site management. Plan to add support for these tasks if content management is also being added.
- Content development requires effective means to simulate site management choices that are not captured as artifacts.
- Different content can require different content management tools. This is especially visible when content management is introduced.

Inventory and analysis lessons: Inventories and their analysis help ensure that a restructuring project is successful. Inventories also help to show progress and to claim project completion.

- An effective inventory for the restructuring project is essential to its success.
- An inventory is likely to reveal many more segments and partitions than originally anticipated.
- An inventory is likely to reveal many more categories and types of entities than originally anticipated.
- Restructuring projects should include support for externally controlled components.
- Frontier analysis of external dependencies and components is essential for successful restructuring. It often exposes components that must be included in the restructuring.
- You will be dealing with a heterogeneous content, and artifacts from deprecated tools.
- Some inventory items are outside the scope of content management.
- As the restructuring proceeds, the inventory shows progress and completion of milestones.
- In addition to an inventory of names, the analysis should include sizes for containers and elements.

Technical debt lessons: Restructuring projects arise from technical debt. Rarely do they remove it all.

- Restructuring projects reveal technical debt.
- Not all known or revealed technical debt will be eliminated at the end of the restructuring project.
- The project is likely to end before all desirable goals are achieved.