Multiplier Effect: Case Studies in Distributions for Publishers

Jon Peck | Courtney Yuskis | @drupalcolorado 2016.08.14

FOUR KITCHENS

Jon Peck

Architect, Four Kitchens

fourkitchens.com
@fluxsauce
linkedin.com/in/jonpeck

Courtney Yuskis

Digital Engagement Director,
Meredith Agrimedia

meredithagrimedia™
farming · growing · living

meredithagrimedia.com
linkedin.com/in/courtneyyuskis

The Opportunity: Meredith Agrimedia

Scope

Agriculture.com

- Online presence for Successful Farming
- Ag news and commodities data
- Reuters Newswire
- >25,000 pieces of content
- Migration from Symfony 1.4 framework

WOODMagazine.com

- Online presence for WOOD Magazine
- Free downloadable plans
- Print article index
- 4,050 pieces of content
- Migration from Teamsite Interwoven

Commonalities: Launch Objectives

- Exceptional UX (content -> audience -> data)
- Maintain brand reputation
- Improve editorial efficiency
- Audience acquisition and engagement
- Ease of testing new revenue models

Commonalities: Technical Details

- Content structure and hierarchy
  - Articles, Images, Slideshows, Authorship, Taxonomies
- Publishing workflow
- In-house integrations
  - Single Sign-On, subscription management
  - Meredith standard analytics
  - RAMP Video
  - Gigya
  - KARMA
  - Lithium
- Platform
  - Hosting, CDN and proxy
  - Memcache, Solr

Disconnects & Challenges

- Partial institutional adoption of Drupal, but no standard
- Disproportionate budget across two properties, but need to deliver complete solution for both
- Multiple client-side product owners
- IT policies and procedures
- 3rd party integrations differ by site

The Results

Efficiencies

- Agriculture.com (1st site): TTL 6 Months
- WoodMagazine.com (2nd site): TTL 5 weeks
- Cost effective and simple to maintain
  - Common changes can be tested and deployed in minutes
  - Structure, tools and techniques are standardized
- Components can be reused within organization
- Quickly identify gaps, redundancies, and opportunities

Client Feedback

- Continued use of train-the-trainer, across sites
- Removed content distribution bottlenecks
- Repurposing/surfacing evergreen content
- Enhanced content automation and distribution
- Content and channel agnostic
- Eliminated advertising discrepancies

By the Numbers

- 43% to 92%: Sitewide viewability (advertising performance)
- 67% decrease in page load times
- 44% increase in exposure to sub-brands
- 88% on-page scroll rate
- 349% increase in successful on-site searches

How We Did It

What is a Drupal Distribution?

- Officially: full copy of core with additional software
  - https://www.drupal.org/documentation/build/distributions
- Practically: framework of dependencies and custom code

Installation Profiles and Distributions

- Installation Profiles configure Drupal
  - Provides installation, configuration steps
- Distributions contain all software
  - Typically includes at least one Installation Profile
- “More details about distributions” - drupal.org
  - https://www.drupal.org/node/1089736#distributions-vs-installation-profiles

Why Not Multisite?

- Fragile, difficult to maintain, doesn't scale
- Candidate for deprecation in Drupal 8, removal in Drupal 9
  - https://www.drupal.org/node/2306013
- "Much Ado about Drupal Multisite" - Josh Koenig
  - https://pantheon.io/blog/drupal-multisite-much-ado-about-drupal-multisite

Types of Distributions

- Monolithic
- Atomic
- Hybrid

Monolithic Distributions

- All code in the same repository
- Advantages
  - Easy to distribute and start working
  - Everything in the same place
- Disadvantages
  - Nigh-impossible to code review
  - Magnificently bloated
  - Messy history
  - Mirroring repositories
  - Patching nightmare

2001: A Space Odyssey (1968), Metro-Goldwyn-Mayer

Build Process

- Converts source files into standalone artifacts
  - Contains everything needed to run
- Monolithic Distributions are artifacts

Steps in a Build

- Download packages and apply patches
  - Drush Make (Drupal 7 and below)
  - Composer (Drupal 8 and above)
  - NPM (JavaScript)
- Compile assets
  - Ex: SCSS to CSS, JavaScript minification, image reduction
- Package for deployment
  - Add to Source Control
  - Copy or Archive

Build Systems

- Aquifer
  - https://github.com/aquifer/aquifer
- BLT
  - https://github.com/acquia/blt
- Grunt Drupal Tasks
  - https://github.com/phase2/grunt-drupal-tasks

Atomic Distributions

- Build process to get components
  - Every custom module in its own repository
- Advantages
  - Explicit separation of history
  - Great for versioning
- Disadvantages
  - Dozens or hundreds of repositories
  - Pull requests are a dependency nightmare
  - Need to maintain build process
  - Slow builds
  - Impractical

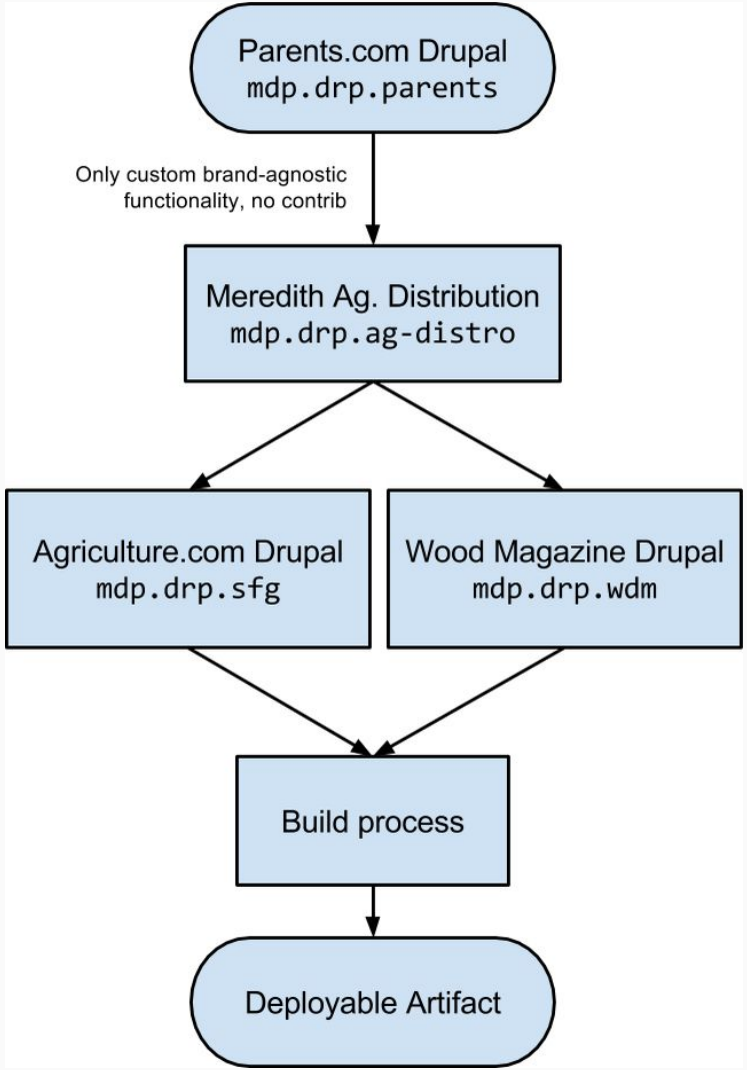
It Came from Beneath the Sea (1995), Columbia Pictures

Hybrid Distributions

- Build process to get components
  - Custom code in Distribution repository
- Advantages
  - Centralized code
  - Easy to work with
  - Faster builds
- Disadvantages
  - Still many dependencies
  - Still needs a build process

The Fly (1958), 20th Century Fox

Meredith Agrimedia’s Distribution



- Distribution (Parent): `ag-distro`
  - Fork: `sfg`
  - Fork: `wdm`
- Changes in parent pulled asynchronously

Testing

- Every code change is automatically checked
- Tools
  - Syntax Errors - phplint
  - Coding Standards - PHP_CodeSniffer / Coder, ESLint
  - Functional Testing - Behat / Mink, Behat Drupal Extension

Local Development

- Standardized approach, platform agnostic
- Drupal VM
  - Not required, but only supported
  - One configuration step
    - Since launch, down to zero (!)
  - Internal hosting leveraged playbooks for consistency
- EditorConfig - file format & text editor plugin for maintaining coding styles

Build Process

- Aquifer - build system
- Composer - PHP package manager
- NPM - JavaScript package manager
  - Gulp - task runner, front end build system
- CircleCI - continuous integration
- Jenkins - deployment

What would we do differently?

- Stabilize build process earlier in project
- Use a single Continuous Integration / Deployment solution
- Install Drupal VM with Composer
- Bare metal test Drupal VM when updating
- Bare metal test documentation prior to new developer onboarding
- Retroactively apply fixes based on deployment of subsequent sites
- Say No

Practical Takeaways

Client-side Champion

- Own the global project
- Identify commonalities
- Mitigate differences
- Critical for scale

Drupal Builds are the Way of the Future*

- Great for large projects
- Can be a bit overkill for small projects

- *Use your best judgement

Hybrid Distributions are Optimal

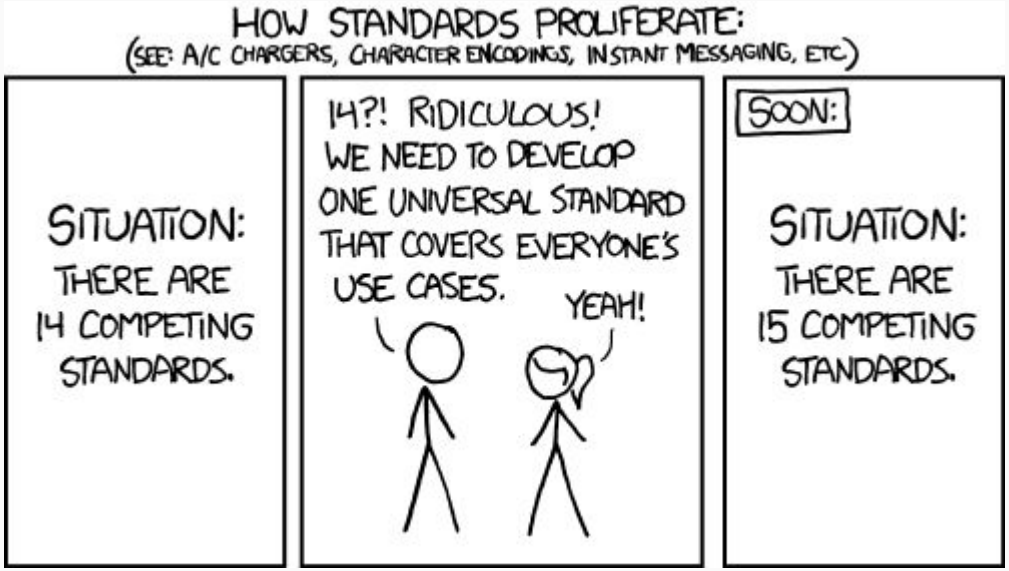
- Consolidate custom work
- Separate contributed code
- Extremists in any context are unpleasant

Automate Quality Checks

- Syntax errors
- Coding standards
- Behavioral testing
- Prevent regressions
- Compare standard components between forks to detect variance

Don’t Reinvent the Wheel

- Investigate and leverage open-source first
  - Contribute back fixes, improvements
- Avoid one-off solutions if practical
- Reusability is awesome (compromise!)



https://xkcd.com/927/

Any questions?

Thank you.

linkedin.com/in/jonpeck
linkedin.com/in/courtneyyuskis
@fourkitchens