HMI custom publishing system

PROJECT DOCUMENT
FEBRUARY 2009
version 1.0 unedited

CEO:	Jeff Haggin
CFO:	Valter Calamita
Director of IT:	John Stein
Product Manager, Architect:	Chad Augur
Development Manager, Sr. Architect:	Chris Brand
Development Lead, Sr. Engineer:	Ed Cass

Contents

Executive Overview

This document is a complete view of development past, current and future of the HMI **conduit**™ Custom Publishing System. Herein are the objectives and goals of the project, an overview of the application itself, and a detailed development path including versioning, scope and features, and time-line.

"What is the HMI **conduit**™ Custom Publishing System?" is an analysis of the purpose of, challenges addressed by, and benefits to Haggin Marketing and their clients in the development of the **conduit**™ Custom Publishing System.

"The Application" gives an overview of the application. A description of development methodologies, technologies and procedures employed to support the continued success of the project. In this section there is also a description of each of the components comprising the system.

"Versions Methodology" looks at the development cycle past, current and future giving descriptive overviews of functionality and component development in each release of the application.

"Versions Scope and Functionality" defines the particular challenges, workflow elements and solutions encompassed in each release of the application.

"What is the HMI conduit™ Custom Publishing System?"

In short the HMI **conduit**™ Custom Publishing System is engineered to accommodate the powerful and unique structure of the Haggin Marketing creative production process. This process can be broken into these core elements of operability: data & asset ingestion, account management & project management, creative development, content management, production design, output & asset routing and data export & analytics. In one-way or another past, current and future development of the system is targeted at supporting all elements of this process.

In this section of the document these four questions will be answered:
How does **conduit**™ fit into Haggin Marketing?
What challenges does **conduit**™ address for Haggin Marketing?
What benefit does **conduit**™ bring to Haggin Marketing?
What benefit does **conduit**™ bring to Our Clients?

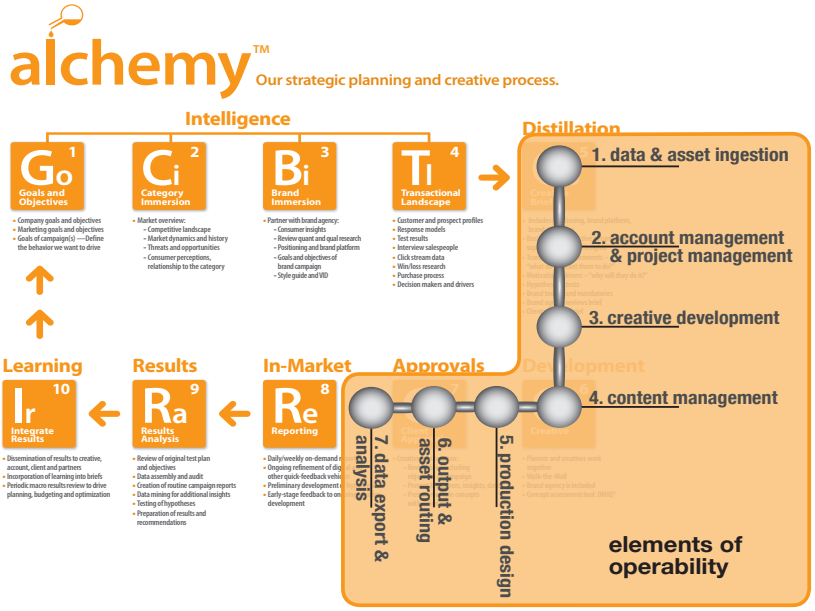
How does conduit™ fit into Haggin Marketing?

Alchemy™, HMI's strategic planning and creative process, provides an apt metaphor and core corporate philosophy that this system will prove to support. In alchemy, intelligence is applied through the use of processes by which the rough stone is made the perfected stone. For the alchemist the apparatus and mechanisms of measurement are foundational to the success of the experiment. The purpose of **conduit**™ is to act as an essential part of this apparatus and mechanism.

conduit™ directly supports stages 5-7 of Alchemy™, HMI's strategic planning and creative process. In these stages the labour of the process occurs, it is where Intelligence becomes manifest. Adapting to requirements of the rules set forth and supporting the creation of the product itself; the system has two-fold purpose, management of content and assets and the automated production of media. Breaking apart some aspects of this process, the system assists in the management of content directed to multiple media outputs, and a capacity to track that content through those streams. Thus, the accurate delivery of high quality, creative and on-brand marketing is further assured by this system.

Being so positioned **conduit**™ is engineered to accommodate the results of stages 1-4, and to inform stages 8-10 of the Alchemy™ process. The system is configurable in order to adhere to the Intelligence stages, as distilled in the fifth stage, of the process. Given that content and assets pass through this apparatus, consistent and measurable results are used to inform the final stages of Alchemy™. This information is targeted at a number of analytical methodologies examining production efficiencies, campaign efficiency, positional testing, and range development. Development of the HMI **conduit**™ Custom Publishing System is driven by the challenge to create an apparatus and mechanism supporting the HMI Alchemy™ process.

(figure 1) Elements of Operability



What challenges does conduit™ address for Haggin Marketing?

There are three challenges **conduit**™ addresses for Haggin Marketing. The retention and expansion of clientele, an efficient and concise production methodology accommodating specialized needs, and the ability carry through and leverage those elements in the market.

Retain What You Have and Build From There

A goal voiced by Lake Capital that a system such as **conduit**™ should achieve is “stickiness”. The system should augment the services provided by Haggin Marketing. It should add to the inherent value in the services provided and the assets managed by the agency. Resulting in clients who not only happily stay with the agency, but also become evangelists thereof. At its core **conduit**™ takes this challenge as a mandate, and approaches the development process in the holistic focus of achieving that stickiness.

Meeting a Unified Process with Specialization

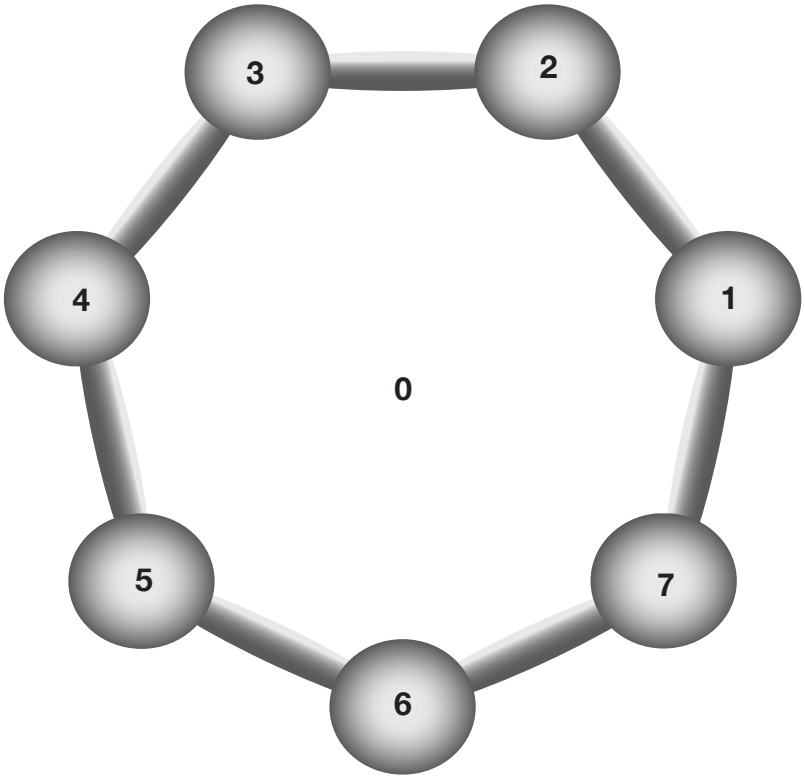
A persistent issue plaguing an organization such as Haggin Marketing is how to meet a consistent and measurable process to specific client needs. The HMI **conduit**™ Custom Publishing System reduces this paradox by defining what is common to both of these needs. There are seven elements apparent in the creative and production cycle.

1. data & asset ingestion
2. account management & project management
3. creative development
4. content management
5. production design
6. output & asset routing
7. data export & analysis

By creating an agnostic core system targeted at supporting these seven elements of operability generally, **conduit**™ can at once accommodate and process specific needs through a consistent and measurable structure.

(figure 2) Structural Diagram of the Elements of Operarability

Structural Diagram of the Elements of Operability



0. HMI production process
1. data & asset ingestion
2. account management & project management
3. creative development
4. content management
5. production design
6. output & asset routing
7. data export & analysis

Sell More Stuff (the SMS principle)

A primary goal that has been identified by Haggin Marketing is to achieve a true one-to-one direct marketing product. Through the application of rules and strategies developed in the Intelligence stages of Alchemy™, **conduit**™ is scalable to any determined range of products, permits for the specialization of those ranges by end-users, and has the capacity to report on those interactions through user-based logging. Appyling this information to evolve the system and to develop of more pertinent and effective marketing interactions.

What benefit does conduit™ bring to Haggin Marketing?

Meeting an architectural methodology with the creative and production processes of Haggin Marketing, **conduit**™ allows for measureable outcomes and increases of efficiency within the agency. The system supports a strong focus on bringing effective creativity to market knowledge through the development and production of on-brand and high-quality marketing media. In gathering these processes into a singular architecture, greater efficiencies are achieved in human and material resources. Staff focuses more closely on product development and quality. Usage of hard goods and services (i.e. paper, couriers, shipping and internal printing) are

reduced as users realize the benefits of the system.

What benefit does conduit™ bring to Our Clients?

In the short term the system supports the strength of the Haggin Marketing process; thus increasing the efficiency and accuracy of the services invested in by the client. In full maturation **conduit**™ will provide public interfaces to many levels of its functionality. Permitting the client, to plan and update projects, edit aspects of content, preview campaigns, and dynamically generate pre-templated marketing materials in many formats. Consumer interfaces allow an individual to select and personalize ranges of products for output to multiple media-streams. Thereby bringing the client that much closer to the meeting the direct needs of the customer.

It can be seen that the HMI **conduit**™ Custom Publishing System is uniquely positioned to provide an immense amount of value to Haggin Marketing's product offerings. The system provides a unified structure to the creative and production process of Haggin Marketing; while accommodating client specialization. Development of the product has been actively focused at accomplishing the long-term sustainability of the system, as well as, the company it supports, increasing the overall value to the client.

The Application

This section addresses the development cycle and application specifics of the HMI **conduit**™ Custom Publishing System. Philosophies and methodologies are briefly described. A high level examination of the technologies used to engineer the system is explored. Procedures employed to assure stability of the system and data integrity are detailed. The team behind the code is introduced. And, finally the components comprising the system are described.

Overview

HMI's **conduit**™ Custom Publishing System is a Rich Internet Application suite. It employs common and well-tested philosophies and technologies in order to engineer a powerful and flexible set of functionality. The system takes into scope the following elements of operability: data & asset ingestion, account management & project management, creative development, content management, production design, output & asset routing, and data export & analytics. In addressing these key elements an architectural methodology arises distilling into a definable, reproducible and measurable system that meets a consistent process to specialized needs.

The architecture of the application addresses this dualistic nature. Centering on an agnostic database structure and modular / component oriented sets of functionality and user interface the **conduit**™ DATA CORE is built to suit a generally applied process versus the specific needs of one particular end-user. Specialization is accommodated in two ways. Being that the core has no particular end-user defined many components of the system may be configured (Release EARTH 2.0) to fit any end-users' business logic specifically (as qualified by being an HMI client). Given the modular nature of component development the application is designed to consider sets of functionality and interfaces not currently in development.

Development Philosophies and Methodologies

Pragmatism, demands of the environment, and fire-tested approaches are central and necessary in the creation of an effective Development Strategy. Detailed below are the core principles driving the development of **conduit**™.

Agile Software Development[1]

A group of software development methodologies that are based on similar principles generally promoting a project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages team work, self-organization and accountability, a set of engineering best practices that allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals.

This particular development environment employs some aspects of the following Agile project management and development methodologies, Rational Unified Process[2], the Dynamic Systems Development Method[3], and Extreme Programming[4]. An essential and common element of all these practices is the principle of iterative or incremental development.

Iterative Design[5]

This can be described as a User Oriented Design[6] philosophy where user interaction with the system is central to the continued development. Based on a cyclic process of prototyping, testing, analyzing, and refining a work in progress, iterative design is a methodology where interaction with the designed system is used as a form of research. These interactions and systems of feedback inform the evolution a project, as successive versions, or iterations of a design are implemented.

Open Source[7]

An approach to design, development, and distribution offering practical accessibility to a product's source (goods and knowledge) that facilitates a great breadth of innovation and support. Some consider open source as one of various possible design approaches, while others consider it a critical strategic element of their operations. Before open source became widely adopted, developers and producers used a variety of phrases to describe the concept; the term open source gained popularity with the rise of the Internet, which provided access to diverse production models, communication paths, and interactive communities.

The open source model of operation and decision-making allows concurrent input of different agendas, approaches and priorities, and differs from the more closed, centralized models of development. The principles and practices are commonly applied to the peer production development of source code for software that is made available for public collaboration. The result of this peer-based collaboration is usually released as open-source software, however open source methods are increasingly being applied in other fields of endeavor, such as Biotechnology.

MVC[8]

Model-view-controller (MVC) is an architectural pattern used in software engineering. Successful use of the pattern isolates business logic from user interface considerations, resulting in an application where it is easier to modify either the visual appearance of the application or the underlying business rules without affecting the other. In MVC, the model represents the information (the data) of the application; the view corresponds to elements of the user interface such as text, checkbox items, and so forth; and the controller manages the communication of data and the business rules used to manipulate the data to and from the model.

[1] en.wikipedia.org/wiki/Agile_development
[2] www-01.ibm.com/software/awdtools/rup/
[3] en.wikipedia.org/wiki/Dynamic_Systems_Development_Method
[4] en.wikipedia.org/wiki/Extreme_Programming
[5] en.wikipedia.org/wiki/Iterative_design
[6] pragmaticmarketing.com/publications/magazine/1/4/the-business-case-for-user-oriented-product-development
[7] en.wikipedia.org/wiki/Open_source
[8] en.wikipedia.org/wiki/Model-view-controller

Technologies Employed

Compact, easily deployable, easily extensible, easily managed, widely used, widely supported these are just a few descriptors for the technology backbone of **conduit**™.

BACK END

LAMP Stack—Linux, Apache, MySQL and PHP[9]

The acronym LAMP refers to a solution stack of software, usually free and open source in nature, used to run dynamic Web sites or servers. The original expansion is as follows:

Linux, referring to the operating system;
Apache, the Web server;
MySQL, the database management system (or database server);
PHP or others (i.e. Perl, Python), the programming languages.

The combination of these technologies is used primarily to define a web server infrastructure, define a programming paradigm of developing software, and establish a software distribution package.

Though the originators of these open source programs did not design them all to work specifically with each other, the combination has become popular because of its low acquisition cost and because of the ubiquity of its components (which come bundled with most current Linux distributions). When used in combination they represent a solution stack of technologies that support application servers.

This is the solution of choice employed by Google, Yahoo, Facebook, SalesForce and most other high volume web sites and applications of note. The mechanisms for creating redundancy and scaling with this technology are well understood, and well known as the most cost effective solution providing the shortest development cycles.

CakePHP[10]

CakePHP is a rapid development framework for PHP that provides an extensible architecture for developing, maintaining, and deploying applications. Using commonly known design patterns like MVC and ORM within the convention over configuration paradigm, CakePHP reduces development costs and helps developers write less code.

User Interface

XHTML[11]

The Extensible Hypertext Markup Language, or XHTML, is a markup language that has the same depth of expression as HTML, but also conforms to XML syntax.While HTML prior to HTML5 was defined as an application of Standard Generalized Markup Language (SGML), a very flexible markup language, XHTML is an application of XML, a more restrictive subset of

SGML. Because they need to be well-formed, true XHTML documents allow for automated processing to be performed using standard XML tools—unlike HTML, which requires a relatively complex, lenient, and generally custom parser. XHTML can be thought of as the intersection of HTML and XML in many respects, since it is a reformulation of HTML in XML. XHTML 1.0 became a World Wide Web Consortium (W3C) Recommendation on January 26, 2000. XHTML 1.1 became a W3C Recommendation on May 31, 2001.

CSS[12]
A stylesheet language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document, including SVG and XUL.

JAVASCRIPT[13]
A scripting language widely used for client-side web development. It was the originating dialect of the ECMAScript standard. It is a dynamic, weakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and was designed to look like Java, but be easier for non-programmers to work with. JavaScript, despite the name, is essentially unrelated to the Java programming language, although both have the common C syntax, and JavaScript copies many Java names and naming conventions. The language's name is the result of a co-marketing deal between Netscape and Sun, in exchange for Netscape bundling Sun's Java runtime with their then-dominant browser. The key design principles within JavaScript are inherited from the Self and Scheme programming languages.

Output

XML[14]
Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

XSLT[15]
An XML-based language used for the transformation of XML documents into other XML or "human-readable" documents. The original document is not changed; rather, a new document is created based on the content of an existing one. The new document may be serialized (output) by the processor in standard XML syntax or in another format, such as HTML or plain text. XSLT is most often used to convert data between different XML schemas or to convert XML data into HTML or XHTML documents for web pages, creating a dynamic web page, or into an intermediate XML format that can be converted to PDF documents.

IDML[16]

The flexible XML options, available in the new InDesign® Markup Language (IDML), and rich scripting support in Adobe® InDesign CS4 and Adobe InDesign CS4 Server software allow for the creation of powerful automated workflows publishing content across various systems and output media.

[9] en.wikipedia.org/wiki/LAMP_(software_bundle)
[10] cakephp.org
[11] w3.org/TR/xhtml1, en.wikipedia.org/wiki/XHTML
[12] w3.org/Style/CSS/, en.wikipedia.org/wiki/CSS
[13] developer.mozilla.org/en/About_JavaScript, en.wikipedia.org/wiki/Javascript
[14] w3.org/XML
[15] w3.org/TR/xslt, en.wikipedia.org/wiki/XSLT
[16] adobe.com/products/indesign/scripting/

Procedures, Fail Safes and Otherwise Good Ideas

All old adages apply “Save often!” “Check the trash before emptying.” “Back Up!” “Cover your @$$.” And, **conduit**™ adheres to an established and tested methodology for doing just those things. Below is a description of the theories and technologies employed in the development process in order to avoid catastrophe.

Tri-Tiered Development to Deployment Environment

The development to deployment environment is a tri-tiered procedural support structure. It consists of a local/working copy environment, a testing environment, and a production environment. This methodology provides a barrier between code &/ structure that changes on a daily basis, and code &/ structure that is deployed as a stable release.

Local Environment

Code Base

This environment should closely resemble the setup of the testing and production environments understanding the differences in OS and tools used by individual developers. All creation and modification to code occurs in this environment. It is here that the “working copy” of the Subversion repository exists (see Version Control).

Database

A direct copy of the most current database should be deployed on each local environment. All hooks to the production database should be carefully disabled, leaving only connections to the database on the local environment.

Testing Environment

Code Base

The testing environment provides each developer with access to a replicated production environment wherein all code that has been committed to the version repository should be updated here and tested thoroughly before final deployment is requested.

Database

A direct copy of the most current database should be deployed on the testing environment. All hooks to the production database should be carefully disabled, leaving only connections to the database on the testing environment.

Production Environment

Code Base

Only approved code and architecture exists on the production server. Most typically code pushed to this environment will be associated with a particular release number (see **conduit**™ Versions Methodology and **conduit**™ *Versions Past, Current and Future*). Modifications to the code base should never be made in this environment.

Database
The production database is housed on a separate server in order to assure data integrity, maximum resource efficiencies and overall peace of mind.

Version Control[14]
A practice facilitating the management of multiple revisions of the same unit of information, most commonly used in engineering and software development. It is employed to manage the ongoing development of digital documents like application source code, art resources such as blueprints or electronic models, and other projects that may be worked on by a team of people. Changes to these documents are usually identified by incrementing an associated number or letter code, termed the "revision number", "revision level", or simply "revision" and associated historically with the person making the change. A simple form of revision control, for example, has the initial issue of a drawing assigned the revision number "1". When the first change is made, the revision number is incremented to "2" and so on.

Technology of Choice
Subversion (SVN)
ADDRESS: Subversion.tigris.org
A version control system used to maintain current and historical versions of files such as source code, web pages, and documentation. Its goal is to be a mostly-compatible successor to the widely used Concurrent Versions System (CVS). Subversion is well-known in the open source community and is used on many open source projects, including Apache Software Foundation, KDE, GNOME, Free Pascal, FreeBSD, GCC, Python, Django, Ruby, Mono, SourceForge.net and Tigris.org. Google Code also provides Subversion hosting for their open source projects. BountySource systems use it exclusively. Subversion is also being adopted in the corporate world. In a 2007 report by Forrester Research, Subversion was recognized as the sole leader in the Standalone Software Configuration Management (SCM) category and a strong performer in the Software Configuration and Change Management (SCCM) category. Subversion is released under the Apache License, making it free software.

SVN and conduit™
Subversion is employed in two ways within the development environment. The entirety of the code base is managed in SVN. This creates a detailed history of code changes, their association to feature requests, improvements and bug reports, and the individual responsible for committing those changes. Through implementation of scheduled shell scripts or CRON jobs the entirety of the database is backed up on and hourly basis and stored within the SVN repository directory.

Version control procedure for conduit™

Requirements

Versioning systems are implemented for the following purposes:

1. Common repository for files and project work
2. Backup
3. Versioned history
4. Easy roll-back of unsuccessful code
5. Change tracking for code review
6. Ownership of code changes
7. Deployment mechanism

Repository Information and Credentials

Location

svn://Subversion.hagginmarketing.com/haggin/conduitvenus

Username

email address name (first name+first initial last name)

Password

Same as username

Repository Structure

Underneath the root Subversion URI described above there are directories for each project. Assuming that the projects in the repository structure are Conduit Venus, Haggin Web Page, and Project X, the directory structure would look like this:

svn://Subversion.hagginmarketing.com/haggin/
    conduitvenus/
    hagginwebpage/
    projectx/

Under each project there are branches, tags and trunk directory. The directory structure can then be displayed as follows.

svn://Subversion.hagginmarketing.com/haggin/
    conduitvenus/
        branches/
        tags/
        trunk/
    hagginwebpage/
        branches/
        tags/
        trunk/

Versioning Strategy
In an iterative design environment the methodology employed for version deployment is called “Branch for Release”. Using this strategy all development takes place in the trunk while releases are moved to branches and deployed from there creating distinct versions that are easily tracked.

See **conduit**™ *Versions Methodology* for a more detailed description of the **conduit**™ version theory and particular version descriptions.

Bug Fixes
Bug fixes or patches are addressed by branching an existing release branch, redeploying changes to the code, and merging those changes back into the trunk as is necessary.

Accessing Subversion
Subversion doesn’t present itself as a typical file system. There are multiple tools out there for accessing a Subversion repository. Recommended suggestions include:

The Command Line:
References: Subversion.tigris.org/
svnbook.red-bean.com/en/1.5/
macpot.com/archives/2005/07/tutorial_subver.html

Subclipse
Address: subclipse.tigris.org
This is a cross-platform Eclipse plug-in. It can be inserted into any eclipse-based product such as Eclipse, Flex Builder, or PDT and allows access directly from the project pane.

TortoiseSVN
Address: Tortoisesv.tigris.org
A Windows based Subversion client Tortoise provides integration into the windows file system and lets you perform most actions directly from Windows Explorer.

Versions
Address: versionsapp.com
Versions provides a very intuitive clean interface. Being a MAC application this is probably the application most of our users will be interested in unless they are utilizing an Eclipse product for their development.

Functions
Checkout
To use a project you have to check the project out. The checkout process moves a copy of the application to a local files system. This copy

directly refers to the copy in the repository, and creates what is known as a working copy.

Update
The update function will pull all changes from the repository to your local copy and provide feedback regarding any conflicts between your copy and the repository copy.

Commit
Commit pushes your changes, to your working copy, back to the repository allowing them to be accessed by others or deployed as necessary.

Workflow

Create Project and Initial Commit
If a project does not already exist request its creation by sending and email to chrisb@hagginmarketing.com. Once the creation of the repository is confirmed then perform the initial commit of your source files to the trunk of the project.

```
svn import /Users/local/Sites/conduit_test_app/ Subversion.
hagginmarketing.com/haggin/conduitvenus/trunk
```

Checkout
Create a working copy of the application by using SVN to checkout the project from the repository and place it on the local file system.

```
svn checkout Subversion.hagginmarketing.com/haggin/conduitvenus
```

Update and Commit
Prior to every commit an update should be done to ensure there are no conflicts between working copy files and files that have already been committed to the repository. After a successful update a working copy can be committed to the repository. If an update results in conflicts first resolve the mismatched lines of code between a working copy and the repository within the working copy, then run another update prior to committing.

```
svn update /Users/local/Sites/conduitvenus/
```

Commit statements for **conduit**™ will almost always be in response to a feature request, improvement, or bug report. In these instances SVN statements should follow the following form:

```
svn commit -m "CONDUIT-# : commit statement"
```

This associates code changes directly to a feature request or improvement, or bug report, and this information is used to track

solutions against requests in our change management software, JIRA.

It is important to update and commit with regularity, this will keep working copies in sync with the most current state of the project. Updating with regularity also ensures that any particular working copy is current in more than one place, providing a simple return path to any code committed by any one developer.

Deployment
Using Subversion projects are deployed to test or production servers, providing a clean and efficient way to move your project through the deployment path.

[14] en.wikipedia.org/wiki/Version_Control

Feature Requests & Improvements, Bug Reporting
Understanding how the code truly works is based on an understanding how it is used. **conduit**™ makes use of tracking and reporting technology at every iteration in both the development cycle and in user response.

Technology of Choice
JIRA
ADDRESS: atlassian.com/software/jira/
Lets you prioritize, assign, track, report and audit your 'issues,' whatever they may be—from software bugs and help-desk tickets to project tasks and change requests.

Client Reporting
The **conduit**™ interface directly connects the user to JIRA through a "HELP" menu-item. Internally at Haggin Marketing JIRA is used to manage HelpDesk tickets, IT projects, and other internal processes, thus it is an process common to internal users of the application. {With application going public this choice of technology may need to be reconsidered}.

SVN commits
Any commit of code to SVN should and can be associated to a particular feature request/improvement, bug report, action item, or reference item in the JIRA system under the **conduit**™ project. (for more see *Version control procedure for* **conduit**™).

Reference Library
JIRA is actively employed as a repository of reference and research materials pertinent to the conduit project. Example data for import, xml schemas, and sample exports of data from the system are all associated with various feature requests and bug reports. There is also research material on pertinent technologies housed here, many of these items are

associated with other tickets in the system. All items can be filtered using the research ticket type.

Code Documentation

Well-documented code facilitates two main functions, standardization of the code base, and effective support for the development of other documentation. Standardization of the code base is dependent on a well-defined function library. Through implementation of templated header information in each PHP file the entirety of the code-base is organized with versioning, commenting and functional code in an automated fashion. This plays a key role in maintaining standards when integrating new development staff into the pre-existing code base and in the continued development of the application. Functional documentation is also important in the development of user documentation, and in the generation of technical papers focused on **conduit**™. {this strategy and choice of technology may need to be revisited with the development of increasingly specialized interfaces employing different underlying architectures (i.e. COCOA—iPhone…}

Technology of Choice

phpDOC

ADDRESS: phpdoc.org/

phpDocumentor, sometimes referred to as phpdoc or phpdocu, is the current standard auto-documentation tool for the php language. Similar to Javadoc, and written in php, phpDocumentor can be used from the command line or a web interface to create professional documentation from php source code. phpDocumentor has support for linking between documentation, incorporating user level documents like tutorials and creation of highlighted source code with cross referencing to php general documentation. A complete list of features is available.

Back up

There are essentially three levels of "back up" employed by **conduit**™. Production and database servers are integrated into the daily back up schema of Haggin Marketing. Active redundancy is used on the database server and second server. Active code resides on the development server and the production server. The development server is home to the Subversion repository, a de-facto "back up" of the code base. Crontabs push hourly database updates to the SVN repository.

Technologies of Choice

-rSYNC

ADDRESS: samba.anu.edu.au/rsync/

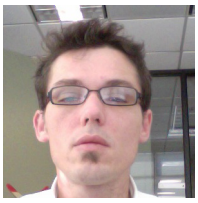
An open source utility that provides fast incremental file transfer. rsync is freely available under the GNU General Public License and is currently being maintained by Wayne Davison.

CRON
ADDRESS: en.wikipedia.org/wiki/Cron
Cron is the name of program that enables unix users to execute commands or scripts (groups of commands) automatically at a specified time/date. It is normally used for sys admin commands, like makewhatis, which builds a search database for the man -k command, or for running a backup script, but can be used for anything. A common use for it today is connecting to the internet and downloading your email.

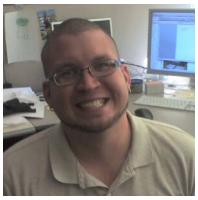
Made Possible By the following Technology providers:

The People

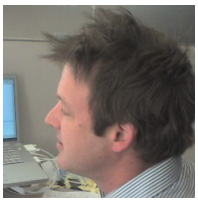
The development team comprises a set of skills ranging a wide scope of creative, technological and management backgrounds. The team bridges common gaps in the creative, production and development aspects of software projects. Bringing a wealth of knowledge, creativity and efficiency to our environment. From heads down developers, to results proven technologists, to balanced and visionary management the development team is poised to produce a quality system.



Chad Augur
Product Manager, Application Architect, UI engineer
caffeine source: Diet Coke
smokes: Djarum
specialities: Smoke and Mirrors, Thought Leadership, 36 hours Shifts, InDesign, XML, HTML, CSS, the 20 minute spin

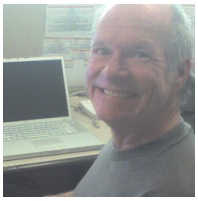


Chris Brand
Development Manager, Senior Architect
caffeine source: Coffee & Diet Coke
smokes: Only under pressure, then is is menthols or some other flavoured cig.
specialties: Hazing, Team Leadership, Telling Chad no, Database Architecture and maintenance, IT management, PHP, technologist.



Ed Cass
Development Lead, Senior Engineer
caffeine source: Coffee Black
smokes: Marlboro Reds
specialties: Smoking, Talkin S#!+, Advanced Problem Solving, amazing analogies, PHP expert, Team Leadership, technologist.

Paul Marshall
Software Developer
caffeine source: Coffee Black, Coke
smokes: Cigars & Cigarillos
specialties: PHP, cakePHP, Applicaiton Architecture Sx



Bill O'connor
Software Developer
caffeine source: Coffee Black,
smokes: Export A's, only on Wednesdays
specialties: PHP, JavaScript, UI interactivity, Data Transformations, XSLT.

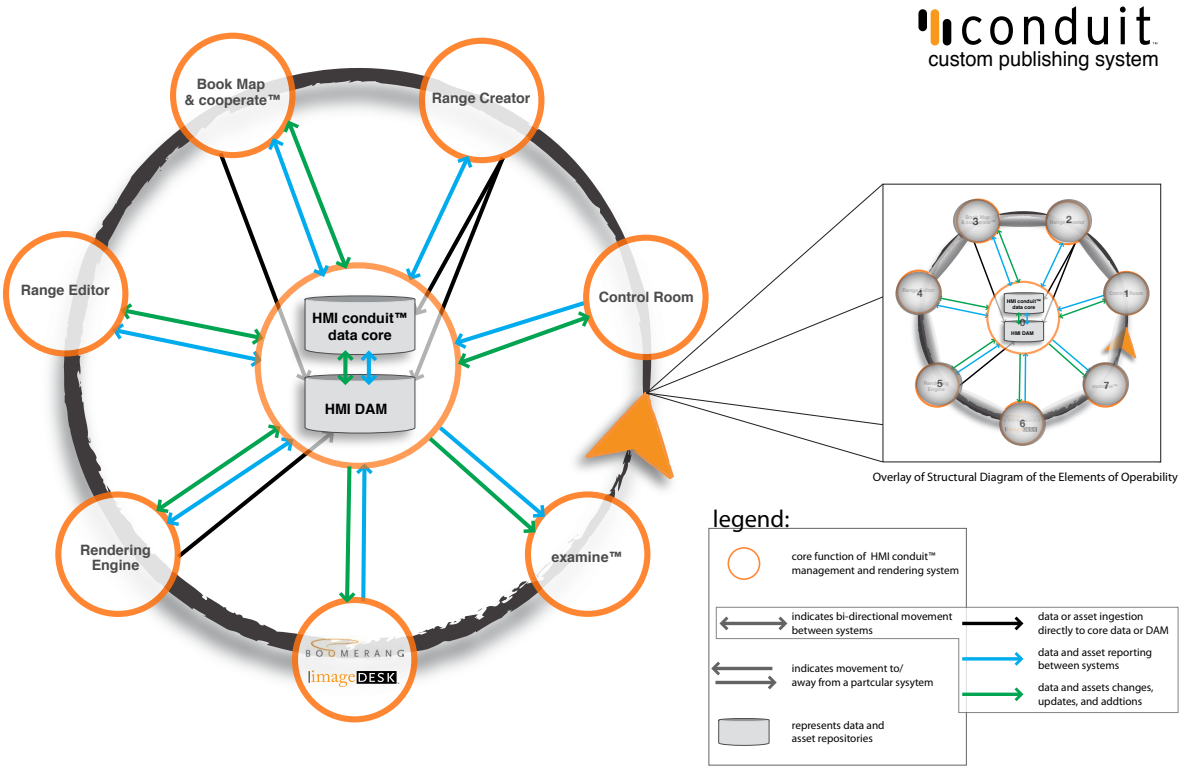
Andrea Scott
Production Specialist
caffeine source: Diet Coke
smokes: Marlboro Lights
specialties: Production Workflow, InDesign, XML

Core Components

conduit™ is a component oriented Custom Publishing System. The system is designed to suit a generally applied process versus the specific needs of one particular end-user. It is engineered to take into account the following elements of operability:

1. data & asset ingestion
2. account management & project management
3. creative development
4. content management
5. production design
6. output & asset routing
7. data export & analysis

(figure 3) conduit™ components



Each component of the system will be introduced giving component name, its current state and those elements of operability covered by it. Component names are generally descriptive in nature. The state of a component will identify where it is in the development process. Elements of Operability will be listed using the numerals 1—7. A brief functional statement will describe, at a high level, what each component does.

Backend

conduit™ **DATA CORE**
state: exists, stable, nearing maturity
elements of operability: 1—7

An agnostic database structure designed to accommodate many types of catalog data in a generalized architecture.

HMI DAM
state: future integration—existing external system
elements of operability: 1, 3—7

Digital Asset Management System is an external backend system currently supporting **conduit**™ functionality as a service. It is the core technology for the Boomerang and ImageDesk systems.

Frontend

Range Editor
state: exists, functions, nearing maturity
element(s) of operability: 4—7
This component is the workhorse of the system. Addition, removal and ranking of products, content edits, and global updates are all done here.

Browser
state: exists, functions, continued refinements
element(s) of operability: 4—7
The Browser is a common complex component appearing in the Range Editor and Range Creator. All functions of the Browser are search oriented or search dependent. Results of searches are seen in the Viewer component. Export is dependent on the selection of a “range” in the Segmented List Filters.

Segmented List Filters
state: exists, functions, continued refinements
element(s) of operability: 4—7

Or, Brower Lists. This component of the Browser is inspired by the iTunes Browser Mode. A series of segmented filter lists configurable according to client specific business logic. The list to the far left is always the Range and the last two lists, to the right, are the product level segmentation and line-list level segmentation. The results of this filter appear in the Viewer Component.

Global Change
state: functional prototype in Venus 1.0
element(s) of operability: 4—7

Range based find and replace, and pattern normalization. In maturation this component will support a dynamic style guide for the creation and management of content rules that can be then be saved and globally applied.

Export
state: back–end function set works, refinements are being made to expose this functionality to the user. Scheduled as Venus 1.1 refinement
element(s) of operability: 4—7

Range based selections from the segmented filter lists as XML or CSV files with the option to download newly created files to the user desktop. The functional relevance of this component is high, as all or part of this functionality will support the Rendering Engine, heavily.

Viewer
state: exists, functions, nearing maturity
element(s) of operability: 4—7

The Viewer is a common complex component appearing in the Range Editor and Range Creator. Comprised of a simple view wrapper, and parent and children asset records. Asset records in the Viewer are the results of filtered search actions preformed in the Browser component.

Product and Line-List objects
state: exists, functions, nearing maturity
element(s) of operability: 4—7

Common simple components, or objects, used through-out the application. Product (Parent) and Line-List (Child) objects are form-based objects exposing relevant product information for the viewing and editing/management of content (Update). The Parent asset contains all general product information (i.e. Price, Name, Marketing Description). The Child asset contains all differentiated information still sharing a common Parent description (i.e. Color, Size, View).

Range Creator
state: functional prototype Venus 1.1
element(s) of operability: 1—7

The Range Creator uses common components from the Range Editor with necessary modification for this functional set. Either import an entirely new range, or create a range of already existing products. Ranges take on a hierarchical workflow from savable search to project; all results of this component are essentially saved searches. Saved searches (user level range) can be promoted to a general range in the Range Admin, and a general range is promoted to a project in the Project Admin.

Browser
state: modification of Browser component, prototype Venus 1.1
element(s) of operability: 1—7

The Browser is a common complex component appearing in the Range Editor and Range Creator. All functions of the Browser are search oriented or search dependent. Results of searches are seen in the Viewer component. Range Creation is dependent on two things; a configurable Import framework and the use of Multi-Conditional filtering. The resulting range is viewed in the Viewer and can then be saved as a search. Range attributes (i.e. Range, Project, Active, Archived, and Delete) are all functions of the Range Admin.

Import
state: exists, functions, continued refinements
element(s) of operability: 1—7

Import is an agnostic framework configurable for specific business logic. Import allows for mass ingestion of client data into an already configured account. (user exposed functionality EARTH 2.0).

Stacking Multi-conditional objects
state: prototype Venus 1.1
element(s) of operability: 2, 4

This functionality is inspired by Entourage's Search functionality. It is based on filter objects that expose choice of attribute, operator, and string. Any number of these filter objects can be generated in order to filter a pre-existing range. The results of this filter are viewed in the Viewer Component, where further manipulation may occur.

Viewer
state: modification of the Range Editor: Viewer component, prototype Venus 1.1
element(s) of operability: 4—7

The Viewer is a common complex component appearing in the Range Editor and Range Creator. Comprised of a simple view wrapper, and parent and children asset records. Asset records seen in the Viewer component are the results of filtered search actions preformed in the Browser component.

Product and Line-List objects
state: exists, functions, nearing maturity
element(s) of operability: 4—7

Common simple components, or objects, used throughout the application. Product (Parent) and Line-List (Child) objects are form-based objects showing product information for viewing and editing/management of content (Update). Parent assets contain all general product information (i.e. Price, Name, Description). Child assets

contains differentiated informationhaving common Parent properties (i.e. Color, Size, View). The delete button for any asset removes that item from the current search only (created range) and not from the parent range(s) from which it was selected.

Control Room
state: functional prototype Venus 1.1
element(s) of operability: 1, 2

Control Room is a dashboard type component exposing user accessible information and functions. Nearly all functionality will be available to all users, however what information is viewed will be based on level of need, or role.

Your Room
state: functional prototype Venus 1.1
element(s) of operability: 1, 2

View, edit, and manage administrative and configuration information related to an account, project, range or user for all accounts and functionality to which a user has access.

User Administration
state: functional prototype Venus 1.1
element(s) of operability: 1, 2

Add new users, Edit User information including password. Edit user roles.

Range Administration
state: functional prototype Venus 1.1
element(s) of operability: 1, 2

Manage Range attributes. Promote a saved search to a Range, Activate and Archive a range, and Delete Range a range.

Project Administration
state: functional prototype Venus 1.1
element(s) of operability: 1, 2

Manage and configure Project attributes. Promote any range to project status. Configure project oriented interface items. Add users to a project; assign sub-Ranges within a project to users.

Account Administration
state: functional prototype EARTH 2.0
element(s) of operability: 1, 2

Setup and configure account specific information. Segmentation,

Product Type, Interface Options, Work Flow options relevant to Account Management can all be created and maintained in this interface.

Rendering Engine
state: redesign- functional prototype Venus 1.5
element(s) of operability: 3, 5—7

The Rendering Engine offers the functional results that all other component development is designed to support. A majority of this component's functionality is not exposed to the user; the configuration of the output is determined by Account and Project parameters. Much of the functionality of this component is directly associated with the export functions of the Range Editor. There are three levels of output types, a universal XML stream exposed to the user and used as the source information for all other output (Venus 1.0), a CSV stream exposed to the user for data delivery to clients (Venus 1.1), and a Media stream including INDD, PDF, HTML, email, and FLASH output (representing a re-design and vast expansion of the WAVE2 functionality which was too rigid). At maturation this component creates a bi-directional relationship between Media Outputs and data, permitting all media and data to be in sync (refined and completed in EARTH releases).

Universal
XML
state: in existence since Mercury 0.1, fully functional in Venus 1.0, this portion will be in constant revision in order to create a purely agnostic XML format.

Data Delivery
CSV
state: stable in Venus 1.1

Media Output
INDD
state: in existence since Mercury 0.1, semi-stable for 3 products in Mercury 0.9. A new functional prototype is planned for release with VENUS 1.5 reaching component stability in EARTH 2.0.

PDF
state: in existence since Mercury 0.1, semi-stable for the products in Mercury 0.9. A new functional prototype is planned for release with VENUS 1.5 with component stability in EARTH 2.0.

HTML
state: current capacity exists naturally with no application.

email
state: EARTH 2.0

FLASH
state: EARTH releases planned integration for Flip Book.

Book Map
state: With EARTH 2.0
element(s) of operability: 2—7

This interactive pagination and planning tool permits the user to place products into pre-selected grids with the capacity to plan multipage documents. The Book Map harnesses the Rendering Engine's power to generate dynamic user previews, the initial build of catalog pages, and output custom pre-templated marketing materials.

Product Library
state: With EARTH 2.0
element(s) of operability: 2—7

Consists of a multi-conditional filter component and an object library.

Stacking Multi-conditional Filter objects
state: prototype EARTH 2.0
element(s) of operability: 2, 4

Footprint compression of the Range Creator component, otherwise the entire functional basis for this component is identical. Results of the filtering action are viewed in the object library.

Object Library
state: prototype EARTH 2.0
element(s) of operability: 2, 4

Image-based object library of products including name &/ Identifier. These objects are dragged and positioned in the Page Map component.

Page Map
state: With EARTH 2.0
element(s) of operability: 2—7

A grid-based interactive library used to plan single and multipage media. Used to preview and update pagination and position, and build catalog pages for production use.

Boomerang
state: future integration—existing external system
element(s) of operability: 4—6

Haggin Marketing's Soft-proof routing and commenting tool, as the core of this application and the interface become outdated updates will be needed. Initial research is being done to integrate this functionality into the **conduit**™ Custom Publishing System.

ImageDesk
state: future integration—existing external system
element(s) of operability: 4—6

Haggin Marketing's asset viewing and routing tool, as the core of this application and the interface become will be needed. Initial research is being done to integrate this functionality into the **conduit**™ system.

cooperate™
ADDRESS: cooperatecreative.com
state: future development—not yet scheduled
element(s) of operability: 2—5

In conceptual examination this is a tool for cooperative creative-concept development, employing aspects of the Book Map, ImageDesk and Boomerang functionality, and the bi-directional communication of the Rendering Engine. cooperate™ allows for interaction between creative professionals using standard design tools, and the client in a web browser.

The art director builds files, as usual, in InDesign. The layout is translated into an object oriented web layout using the Rendering Engine. Borrowing functionality from Book Map the client is able to move objects in the layout within a standard web browser. The user is also able to edit content and change images within the layout. The client attaches direct comments to edits with a simple commenting tool by double clicking the object. All changes and comments are then compiled into a formatted report. This is an extension of the DALiM DiALOGUE concept however it is much easier and cheaper to maintain and deploy given the pre-existing core architectures. Client changes can also be directly integrated back into the existing InDesign file at the agencies discretion.

While this tool is in concept phase currently the **conduit**™ core is specifically designed to incorporate specified interfaces and well- tested elements of this core functionality will pre-exist its realized development.

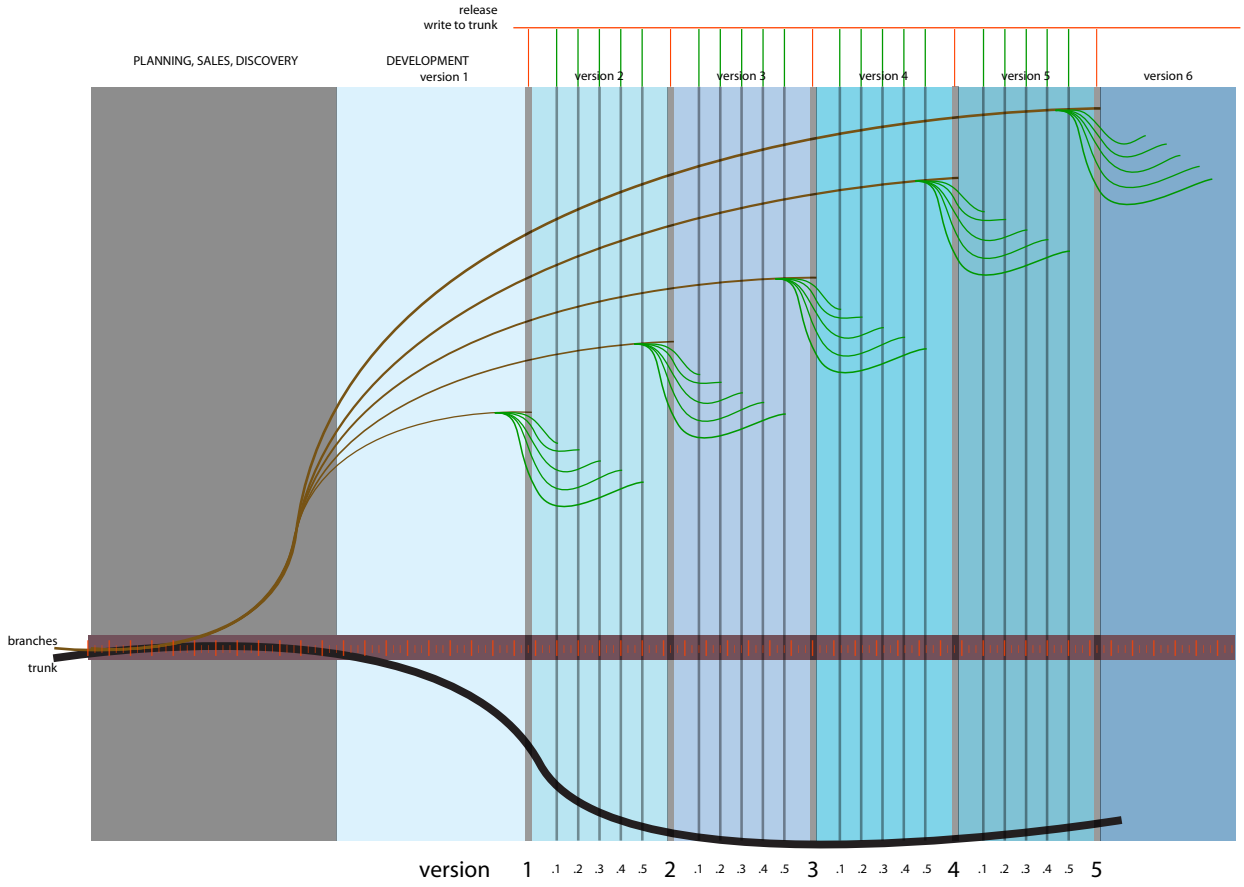
examine™
state: future development—not yet scheduled
element(s) of operability: 1 & 7

Analytics package examining user and error logs in relation to interface and campaign development. Used to inform the final stages of Alchemy™.

conduit™ Versions Methodology

Each scheduled official release represents the introduction of at least one new complete group of functionality. Additionally these releases will show refinements to past functionality as well as support for bug fixes or otherwise unforeseen factors that plague development cycles. Thus our announced versioning will make significant leaps in numbering methodology. Base Versions (i.e. MERCURY 0., VENUS 1., EARTH 2.) all begin with the lowest available ".x" in their set (i.e. MERCURY 0.1, VENUS 1.0, EARTH 2.0) and end with the highest available ".x" in that set (i.e. MERCURY 0.9, VENUS 1.9, EARTH 2.9). In the series of releases there are typically three major Revisions any Base Version (X.) these are most typically .1, .5, and .9. These Revisions represent significant additions or improvements to the scope of the Base Version. Modifications to a Revision (.2,.3,.4,.6,.7,.8) will be team-based milestones not announced publicly. And, Refinements to Modifications (X.x.x) are also team-based milestones representing short-term goals attached to our internal development process and review.

(figure 4) Branch Trunk Version Theory, for more see the **Version Control** *section*



conduit™ Versions Past, Current and Future

The project began purely as a concept with a few catalogs to analyze and an idea of what technologies were needed to accomplish the goal of building thousands of pages from a data driven system. The pitch to the first client was a description of workflow and some screen-shots concepting this solution. On that some quick moves had to be made putting

a solution into place. Over the course of the project the concept has evolved in stages of modular development, testing each theory in production, thus building an educated foundation for continued progress. While not ideal in some instances for short term efficiency;… **conduit**™ Mercury was the evolution of concept to proof, **conduit**™ Venus is the realization of that proof as a stable and extensible application, and **conduit**™ Earth will be a comprehensive core application with modular functionality development designed to accommodate the entire HMI internal and client production workflows.

MERCURY 0.1-0.9 (released and retired)

MERCURY 0.1 (released December 2007)

Implements a project specific data structure. Employs a rudimentary HTML and PHP interface allowing for the editing of products and line-lists, and the rigid ordering of categories, products and line-lists. Data output to XML. Introduces Page Building for one product type with Wave2 application. Requires a coarse replication and modification of the entire system to accommodate a second product type.

MERCURY 0.5 (released April 2008)

Introduces the Range Editor representing a massive leap forward in interaction with the data through a modern interface built in FLEX. Stabilizes rigid database, and PHP adapts to accommodate multiple databases as a solution to multiple product types. Continues build of Wave2 application to address order, data leaks, and a second product type based on the original page rules logic.

MERCURY 0.9 (released June 2008—mostly failed)

Bolstering and Stabilization of the Range Editor to accommodate Adding and Removing products and line-lists from a range, addition of user feedback mechanisms, addresses a number of missed functions and interaction in the 0.5 release. Stabilization of Wave2 in order to accurately produce Two independent product Types, addresses bugs with style, image placement, and inconsistent application of page rules.

VENUS 1.0-1.9 (in development, scheduled end Summer 2009)

VENUS 1.0 (released September 2008)

The **conduit**™ DATA CORE implements an agnostic data structure accommodating all types of catalog input. Maturation of the Range Editor as a stable component with a move in technology from FLEX to a CakePHP using XHTML, CSS and Javascript in the UI. Provides complete content management of a particular “range” of products including the Addition, Removal, Editing and Ordering of product information. Stabilization of import and export through the backend; supporting multiple projects with singular or mixed product type ranges. Introduces the beginning stages of the Dynamic Style Guide in the Global Changes Component, including find/replace, pattern search normalization, and un-do global changes.

VENUS 1.0.5 (released December 2008)
A refactor and stabilization project, does not effect any portion of user functionality or experience. Improves code adherence to framework standards. Increases the agnostic nature of the entire code base to further the goal of accommodating multiple client data and interaction sets.

VENUS 1.1 (in development & production)

Introduces the Range Creator a filter based component used to create search results from any pre-existing range of products, save those searches as personal shortcuts, and promote those searches to ranges that can be edited in the Range Editor. Introduces the Control Room, a component exposing account, range and administration information and functions available to the user. Introduces limited Role Based controls. Improves Range Editor through the addition dynamic searches, continued bug fixes and updates to Global Changes. Includes heavy R&D for Rendering Engine.

VENUS 1.5

Refines the Control Room and Range Creator, and broadens the role-based architecture implementation. Introduces an Account Administration interface allowing for the definition of new account segmentation and interface options. Introduces Project Management functionality to the Project Administration component, implementing the ability to assign sub-ranges, milestone projects, and message regarding the flow of assignments within projects. Beginning phase of the Rendering Engine.

VENUS 1.9

Refines the Project and Account Management tools. Web services to HMI DAM integrating images into product data. Introduces a client-specific Drag-&-Drop Logo library with in product level components. Beginning steps to full Rendering Engine Integration. Explorations of additional client integrations become feasible at this point (i.e. AMEX, Bare Necessities, USAA).

EARTH 2.0

Refines and extends the Rendering Engine to accommodate PDF, email, and HTML output. Introduces Book Map, a page planning and small-media generation tool. The Book Map component is comprised of an object oriented Product Library and a dynamic grid Page Map tool allowing for the search of products, and placement of the of those objects through drag-&-drop functionality, creating Book Maps and/or limited marketing materials based on output selection. Extends the project and account management functionality, and addresses any linger issues from **conduit**™ VENUS.

conduit™ Versions Scope and Functionality

Mercury 0.1-0.9

The introductory phase, entailing a lot of R&D and long hours to get a system in place to accomodate a singular purpose. This phase ended in the Summer of 2008, and provided the ground work for the successful release of **conduit**™ VENUS 1.0.

Challenge
With a loose scope definition, catalog samples, and limited sample data sets how should a system be engineered to automate the management of content and the generation of catalog pages?

Workflow

Client Data: ➠ Import: ➠ Manage:➠ Export:➠ Generate Pages:➠ Deliverables

Solution

Client Data:CSV,Access ➠ **Import:**PHP, MySQL ➠ **Manage:**PHP, HTML/FLEX, MySQL ➠ **Export:**XML ➠ **Generate Pages:**Wave2, INDD ➠ **Deliver:**Pages, flat CSV data

As a proof of concept this solution got the job done with a high level of human intervention. This phase resulted in a workable solution of one client having a limited range of products; leaving a number of usability and automation gaps to be addressed in further development.

Venus 1.0-1.9

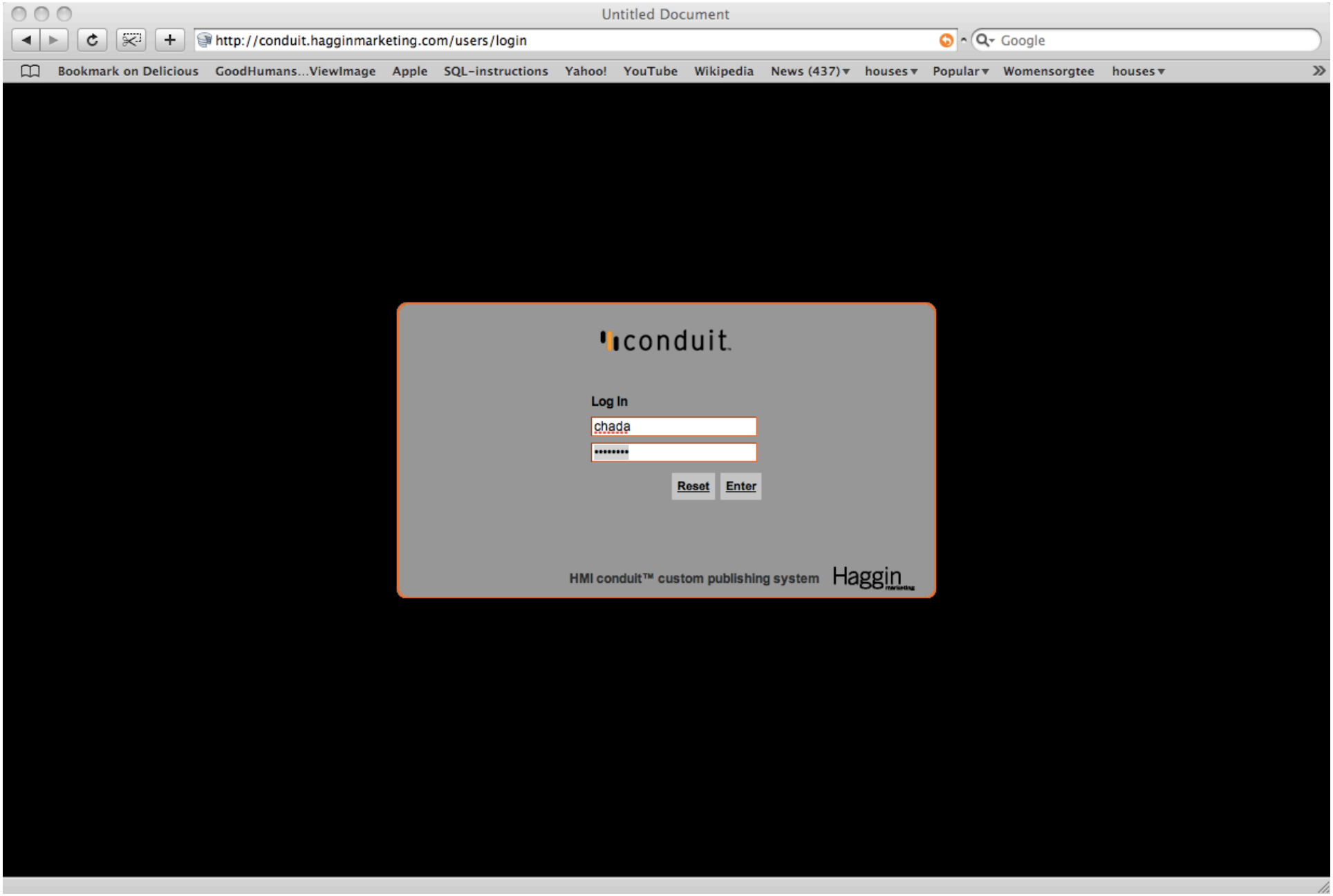
The application moves from a funcitoning prototype to a robust data core, cutting edge CMS, and multi-channel output engine. Architecture, Engineering, Code and Process standards maintain the backbone of continued and agile product develpment. Component development centers on the elements of operability.

VENUS 1.0

Challenge
Building from the Mercury information: How will the reliability, robustness, and efficiency of the system be increased? How will the system's architecture, functionality and interface by generalized for any client? And what processes are in place to accomodate client feedback (i.e. bug reporting, improvement requests, usability surveys)?

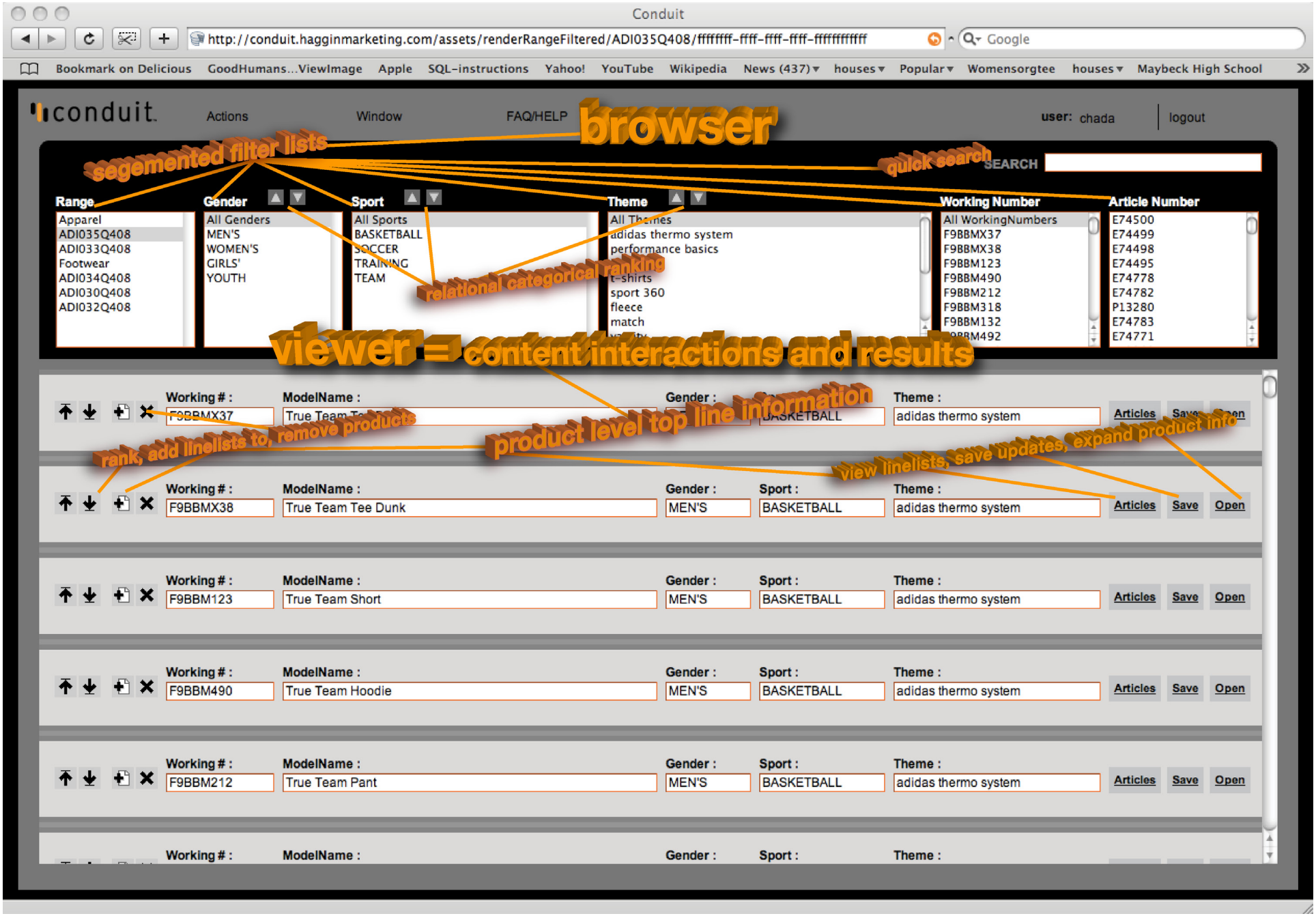
Workflow and Solution
See the following diagrams venus1.0-1 thru 6

conduit workflow & wireframe diagram venus1.0-1: **Login:** User interactions are logged and assoicated with effected data. Ground work for Role Based interactions.



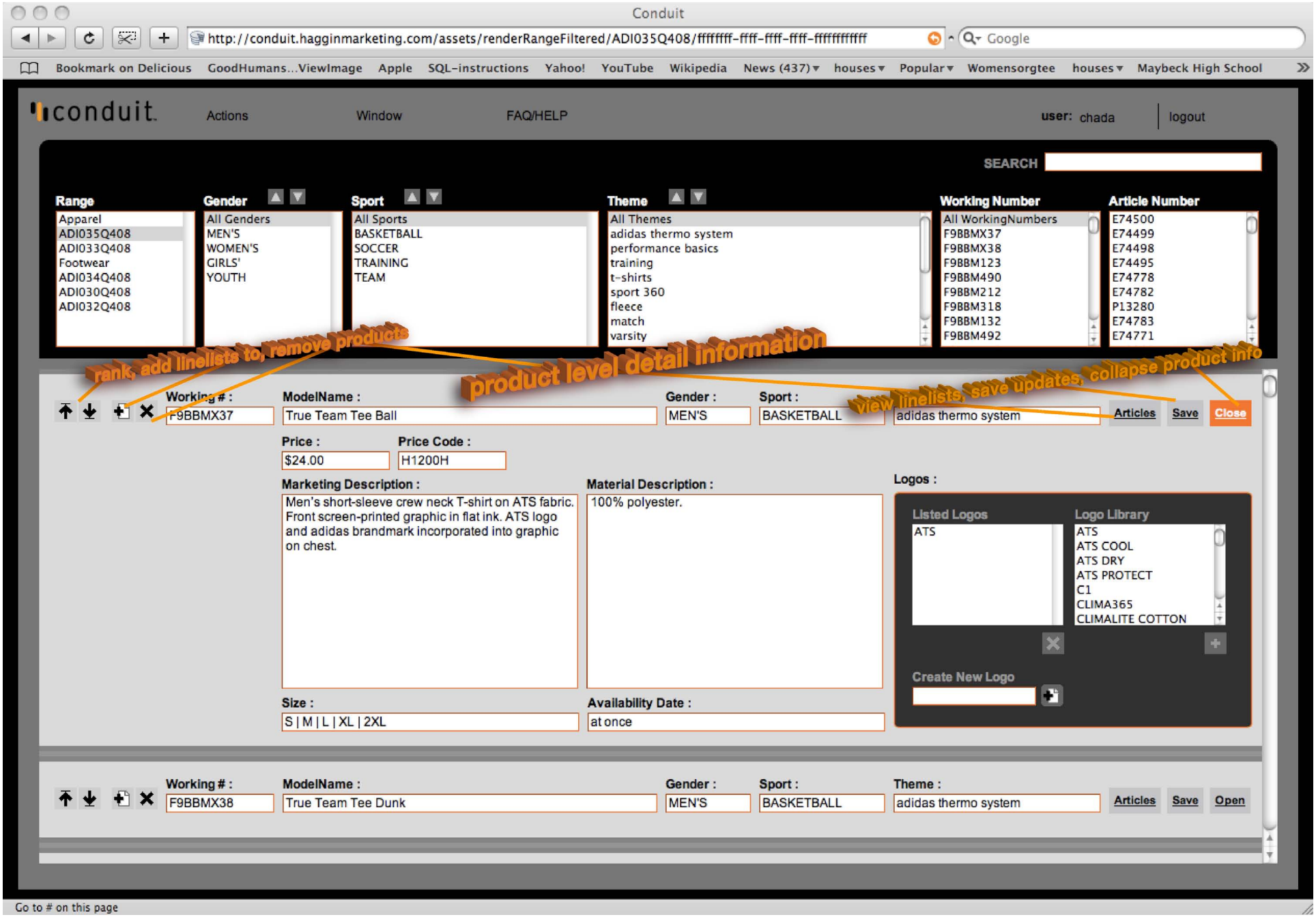
conduit

workflow & wireframe diagram venus1.0-2: **Range Editor:** Browser, Viewer . The Browser a segemented filter narrows themed product information to be viewed, edited and ordered in the Viewer.



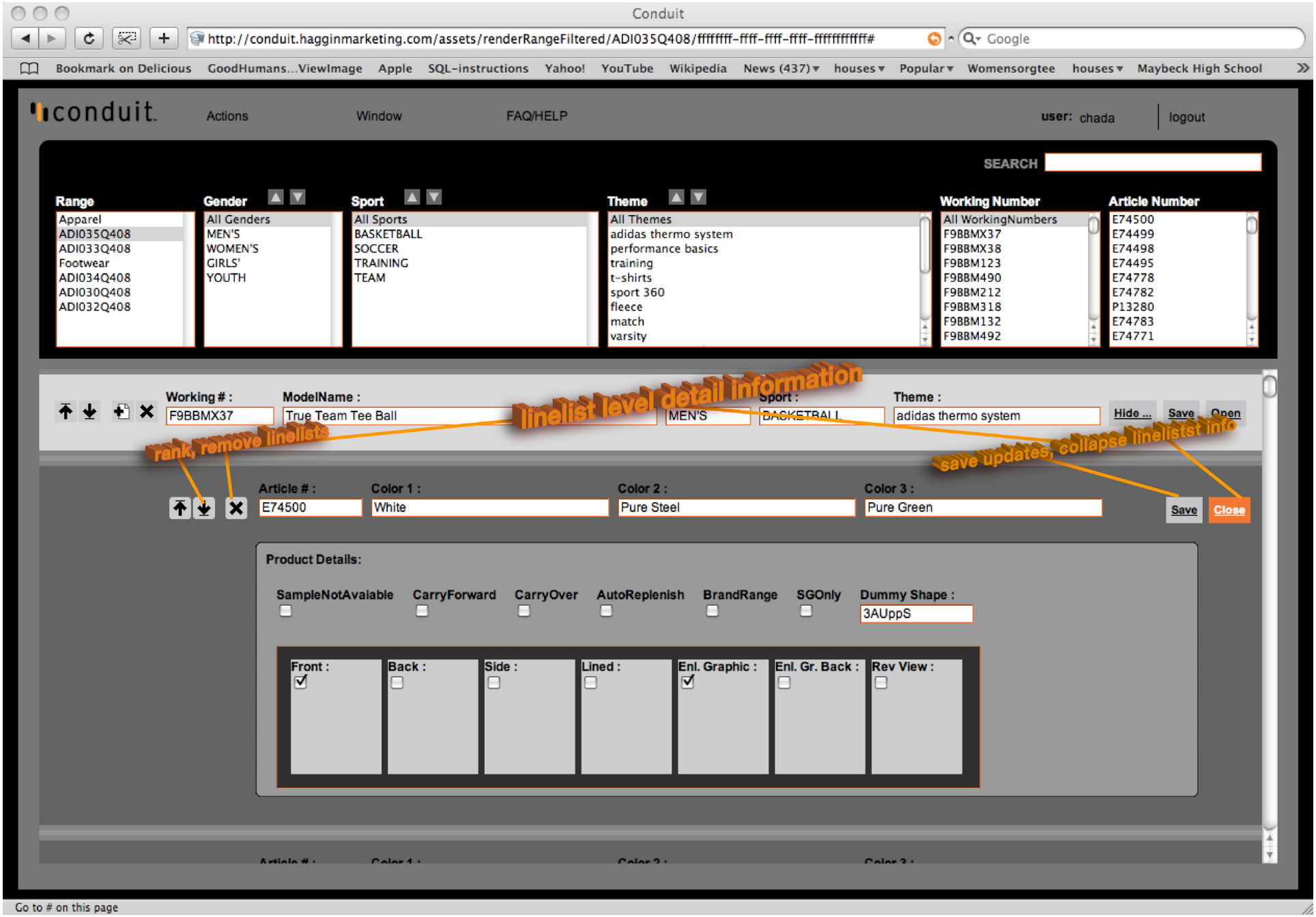
conduit

workflow & wireframe diagram venus1.0-3: **Product Level** information is organized and associated to accomodate fast viewing of a range and accessbile detail states for easy editing of information.



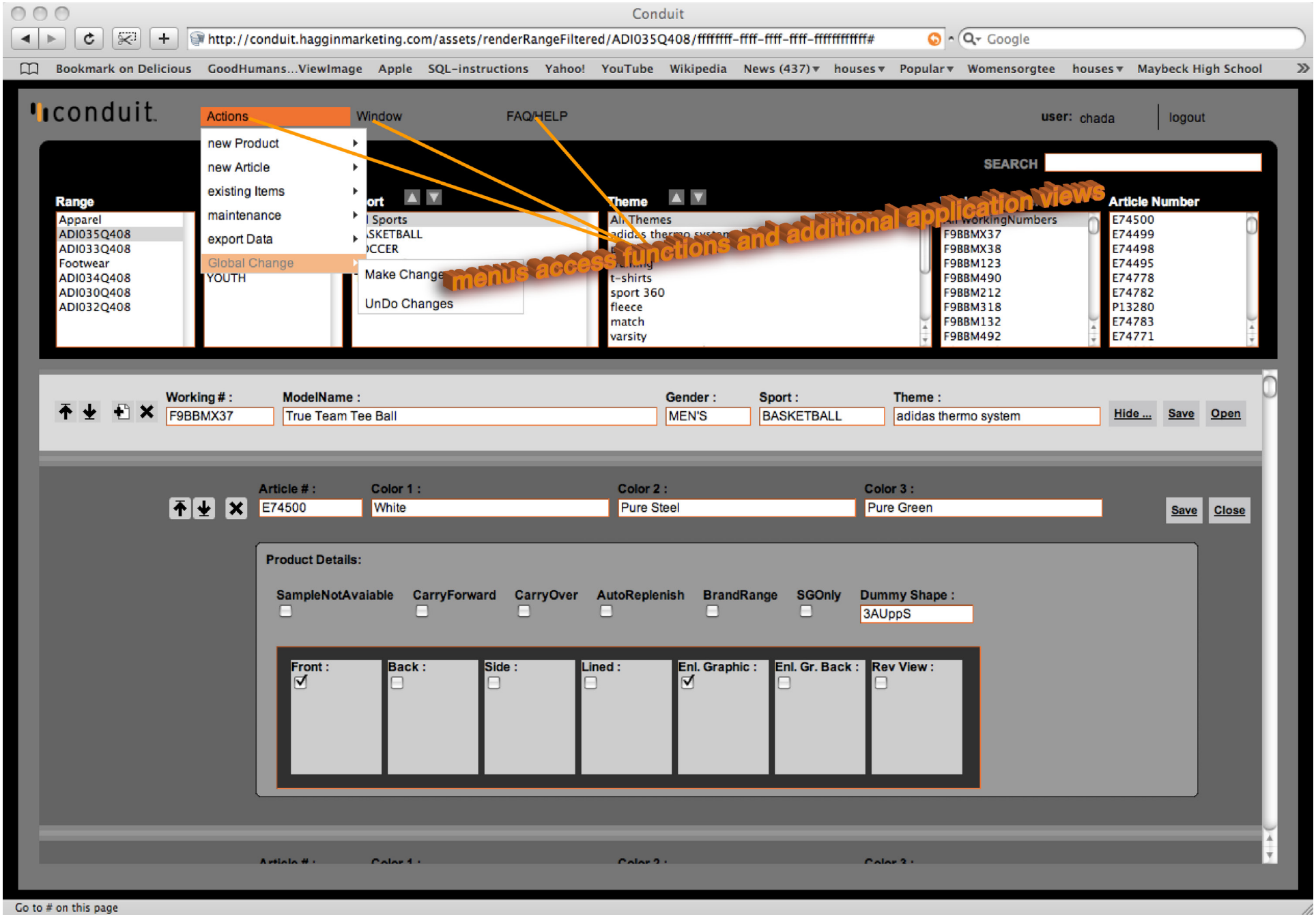
conduit

workflow & wireframe diagram venus1.0-4: **LineList** information is organized and associated to accomodate fast viewing of linelists and accessbile detail states for easy editing of information.



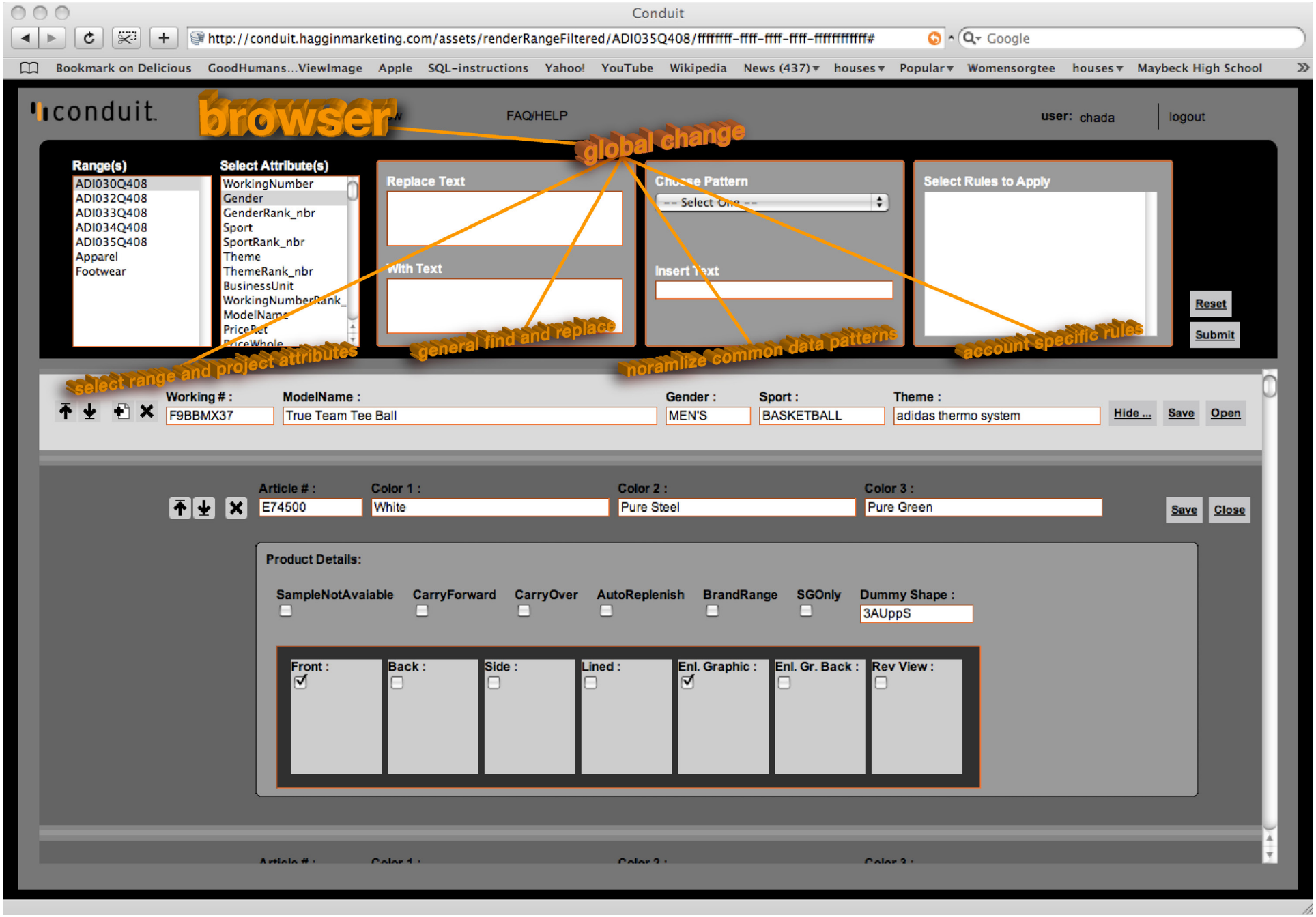
conduit

workflow & wireframe diagram venus1.0-5: **Actions Menu:** New Product, New Article, maintenance, Export Data, & Global Change are all actions that can be commited to a paricular range of products.



conduit

workflow & wireframe diagram venus1.0-6: **Global Chage:** general find and replace, pattern noramlization, and foundations for a rules engine. Applies edits across all account data in one action.



VENUS 1.1

Challenge
How will the system permit for dynamic range creations, client data imports, and how will the system accomodate the emerging role based workflow that has developed?

Workflow

Control Room
(See workflow & wireframe diagrams controlroom1.0-1 thru 3)
Login ➠ ControlRoom:YourRoom ➠ 1. or 2.

1. Select Account ➠ Select Project ➠ Range Editor
2. Select Administrative Action ➠ Drag Action into Actions Area ➠ Commit Action

Range Creator

(See workflow & wireframe diagram Range Creator 1.0-1)
Import New Range/of select pre-existing data: ➠ setup search arrary➠ commit search ➠ save search condiontions ➠ create new range from search results.

Interface Spec
Search for {Any} or {All}
DIV wrapper for filters contains all filters allowing for overflow-y with scroll

<!--Filter objects-->

{button:add another}

{button:delete current}

{Select: Attribute: ALL Attibutes, CATALOG TYPE, PRODUCT TYPE, Project, are constants. All other attributes depend on what the database tells us though we may want to limit this to all common attributes (i.e. ModelName, Color1,2,3, Segmentation(Determined by account configuration for adidas it is Gender, Sport, Theme, WorkingNumber, Articlenumber), Descriptions), etc...) with in an account.

{Select:Operators: Contains, Does Not Contain, etc...this part is pretty solid}

{Value field: We will do our first release with as a text field only. For refinements we will want the value field to be dynamically determined by attribute type and/or operator selection sometimes a text field is appropriate

and sometimes a drop down selector may be more useful.}

Close Filter Wrapper & repeat as needed.

Close DIV wrapper

{button:Submit Search}

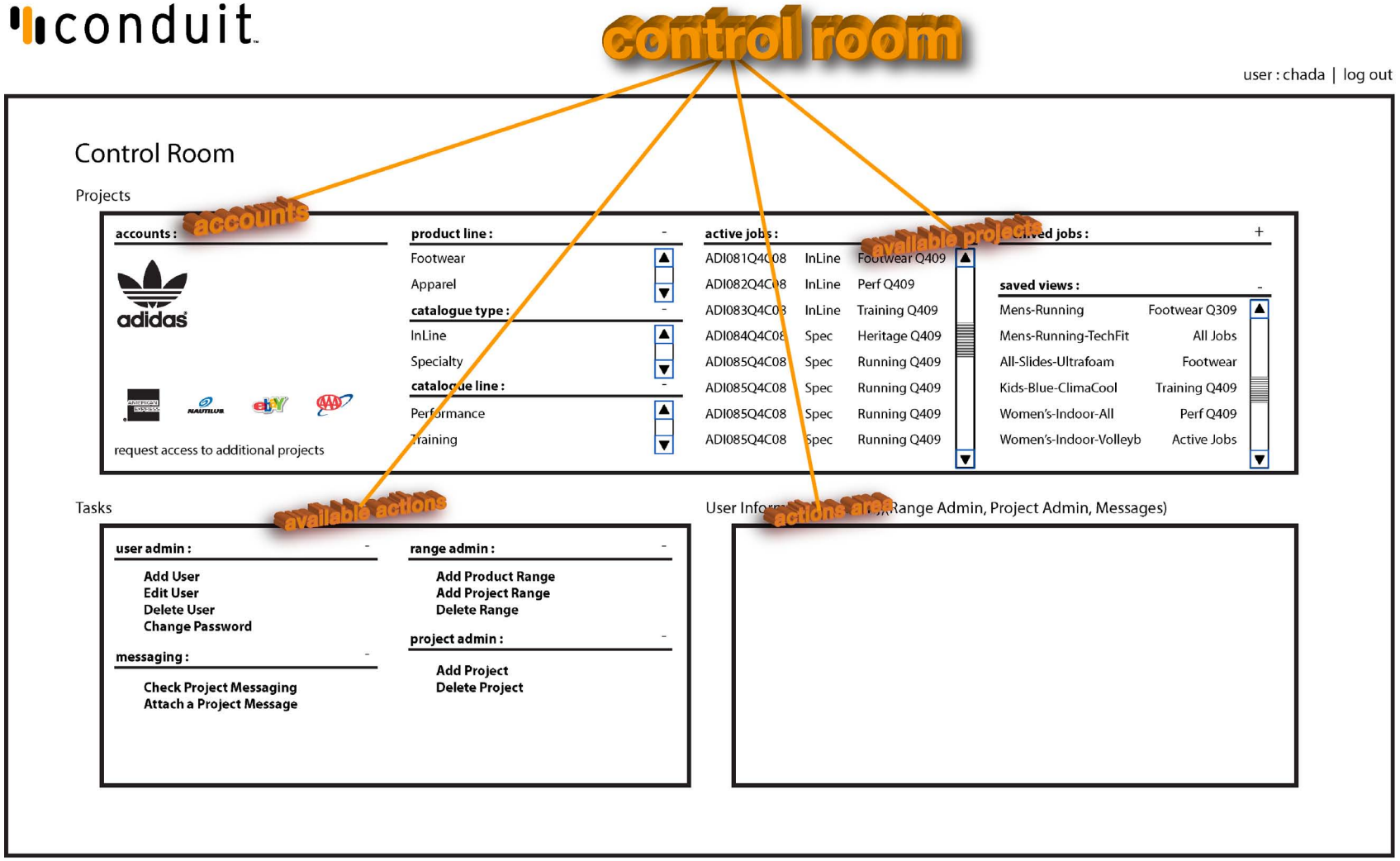
Toggle group
Append Search to existing range {T/F} if so {select:Range Name}

Or Create new range from search{T/F} if so {input: Name New Range}

Neccesitating a couple of things, we cannot create a range until after the filtering is done, which meaning result must appear in the Viewer.

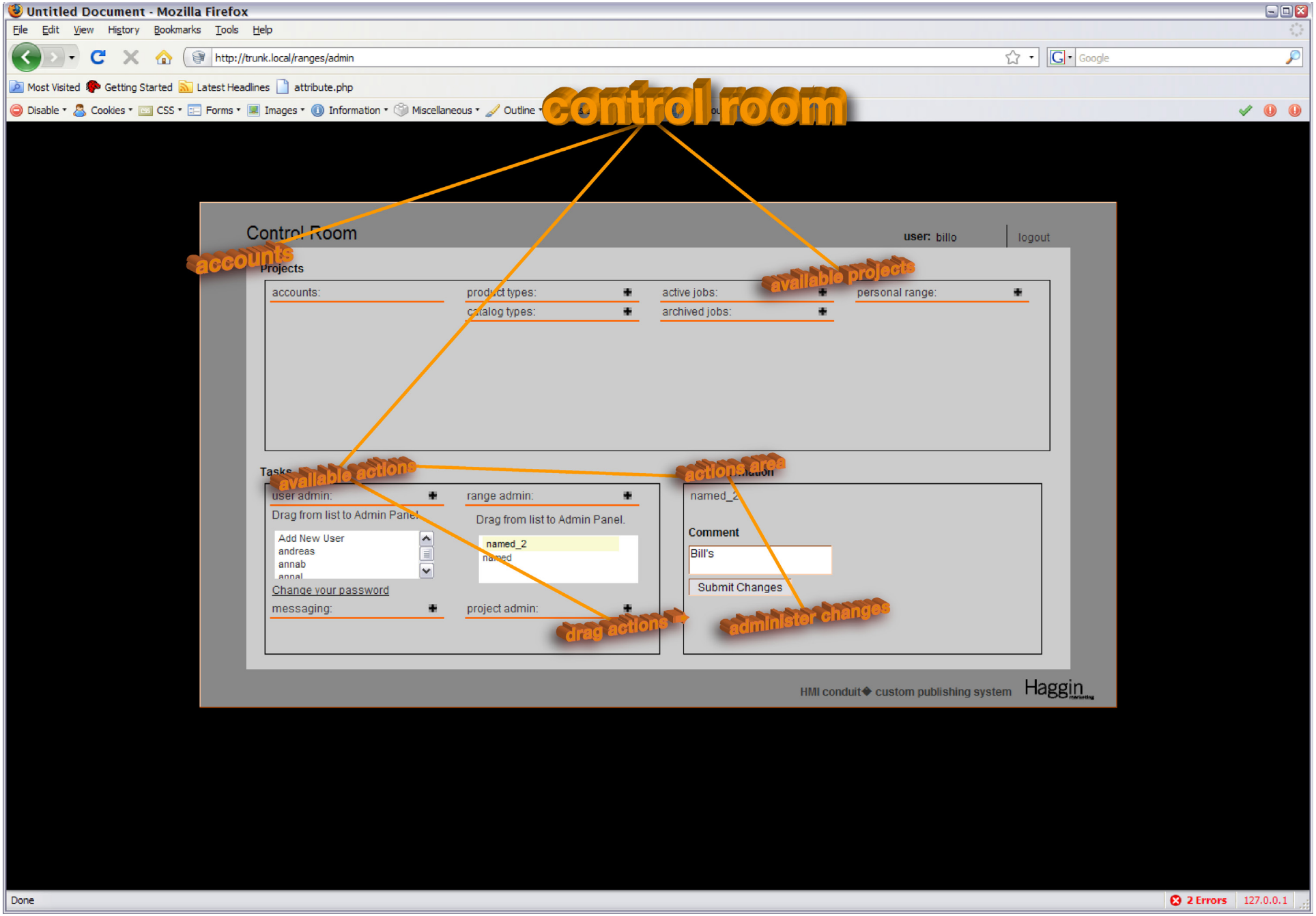
conduit

workflow & wireframe diagram control room1.0-1: **Control Room:** A user dashboard allowing for views of available projects, and adminiatration of user, project, and account information based on role.



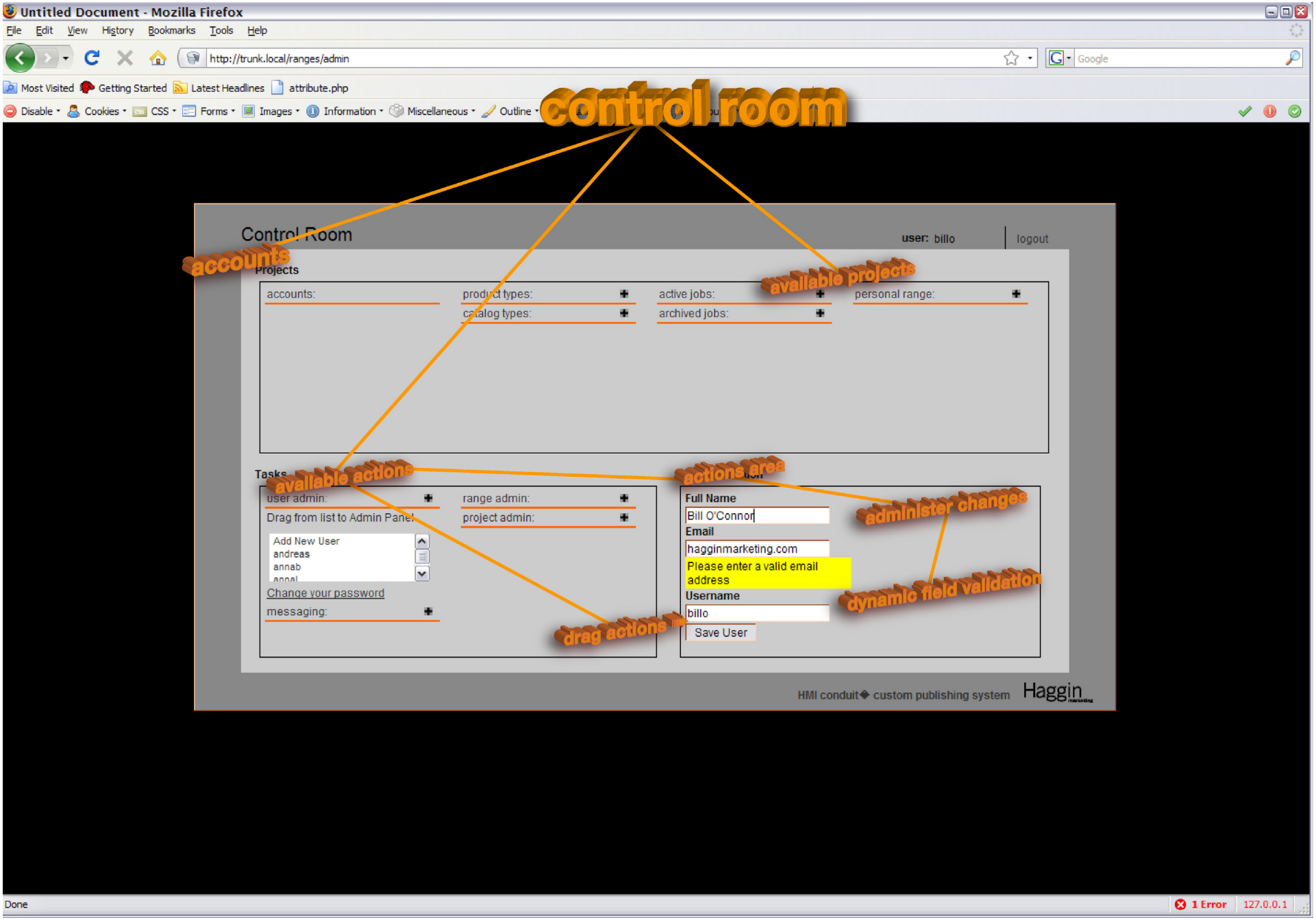
conduit

workflow & wireframe diagram control room1.0-2: **Draggable Actions**: User are able to perform specific administrative funtions, sctions are dragged form the actions pallete to actions area to make changes.



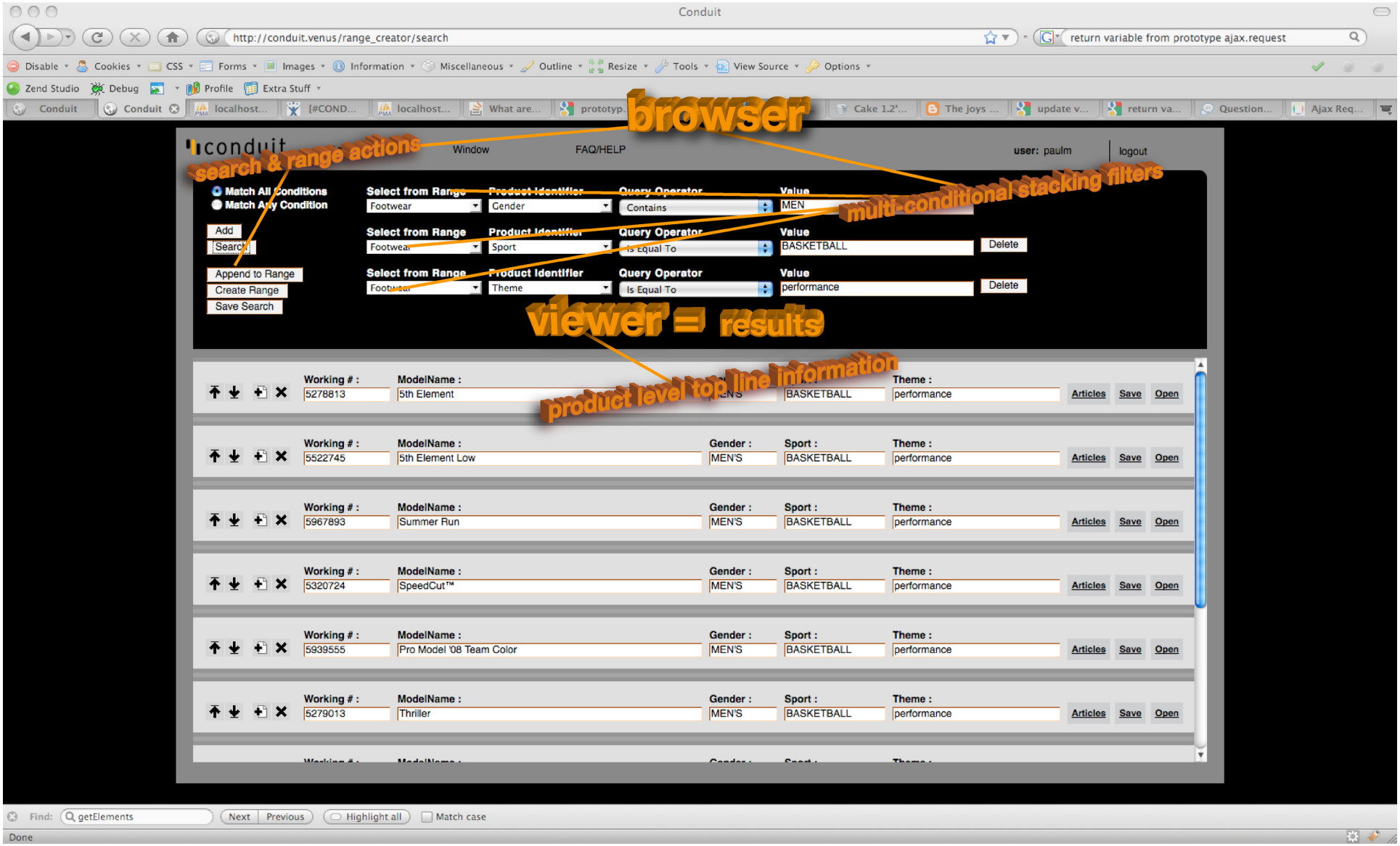
conduit

workflow & wireframe diagram control room1.0-3: **Dynamic Field Valiadation**: Any action that must meet a specific requirement can be validated upon the commission of an action avoiding bad data.



conduit

workflow & wireframe diagram Range Creator 1.0-1: **Range Creator**: Import new client data, search existing data, save search conditions, commit searches to new ranges. For user based-range creation.



VENUS 1.5

Challenge

How to open up the output funcitonality of the rendering engine, making it adaptable to any type of out put. Specifically focusing on mapping the IDML strucure from the InDesign Pages to the cSML structure in order to develop a strong and flexible cSTL engine.

Workflow

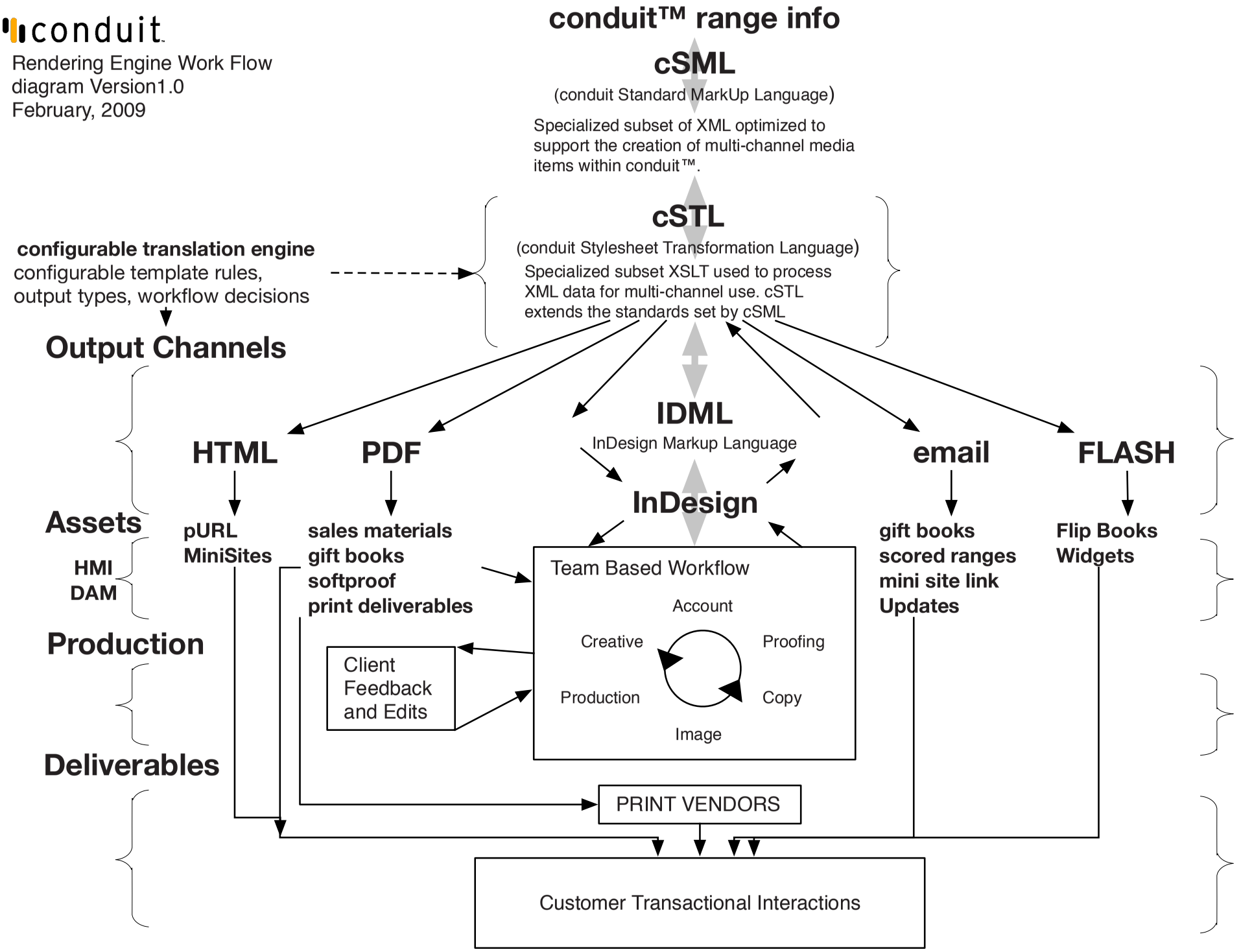
See diagram Rendering Workflow Diagram 1.0 to follow

Solution

The complete nature of this solution will be a workflow and back end solution combined with interface interactions in order to configure the cSTL rules engine to accept specific client business-logic.

conduit

Rendering Engine Work Flow
diagram Version1.0
February, 2009



VENUS 1.9

Challenge

Clean up stage. Refine and repair any bugs to all existing functionality. Continue development on the Rendering Engine and cSTL rules engine. Prepare to close this version of the application. Document all code and interface interactions.

Earth 2.0

Challenge

Take this to the next level. How do we put a truly client facing interface onto the backend of the system. What funcitonality will be usefuly in approaching client interfaces? What funcitonality will need to be created to accomplish this end?

Workflow & Solution

See diagrams Book Map Wire Frame 1.0-1 thru 4 to follow

conduit

Book Map Wireframe
diagram 1 Version1.0
February, 2009

conduit™ range info

BookMap

Specialized subset of XML optimized to support the creation of multi-channel media items within conduit™.

Product Library

Page Map

{droppable}

Product Selection

{Filter Products}
{Enter CSV list}
{Create from Existing}
selection populates
this div see proceeding
slides for filter objects

Pages

of pgs | add to map

Add to Existing

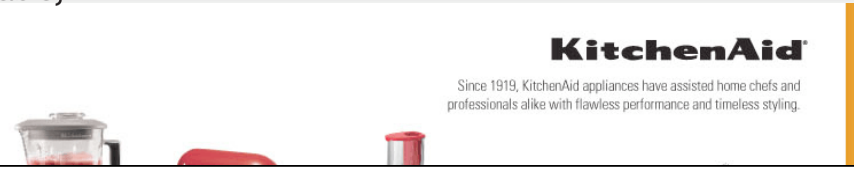
name | new map

Page#	Grid
XXXXXX	Default
XXXXXX	Default
XXXXXX	Default

grids					
0	2	4	6	8	10
1	3	5	7	9	11

{draggable}

{dbl click & edit}



Page Attributes

Attributes	Values
Account	AMEX
Project Code	AMX009Q4
Project Type	Catalog
Catlog Type	MERCH
Category Type	HOME
Product Type	Cooking Ess
Page #	23

Current Products on Page

-	SKU	Name
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME

delete page | update

Page Attributes

Attributes	Values
Account	AMEX
Project Code	AMX009Q4

conduit

Book Map Wireframe
diagram 2 Version1.0
February, 2009

conduit™ range info

BookMap

Specialized subset of XML optimized to support the creation of multi-channel media items within conduit™.

Product Library

Page Map

{droppable}

Filter Products

Range Filter

Attribute Filter

Select Operator

Text Filter

add to last | search

Pages

of pgs | add to map

Add to Existing

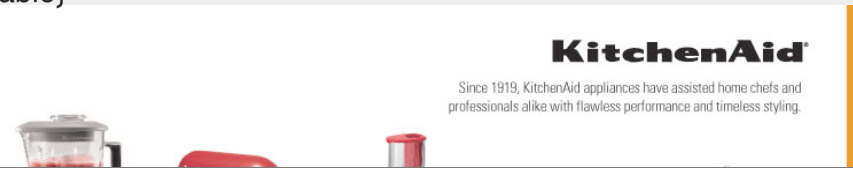
name | new map

Page#	Grid
XXXXXX	Default
XXXXXX	Default
XXXXXX	Default

grids					
0	2	4	6	8	10
1	3	5	7	9	11

{draggable}

{dbl click & edit}



Page Attributes

Attributes	Values
Account	AMEX
Project Code	AMX009Q4
Project Type	Catalog
Catlog Type	MERCH
Category Type	HOME
Product Type	Cooking Ess
Page #	23

Current Products on Page

-	SKU	Name
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME
	XXXXXX	ACME

delete page | update

Page Attributes

Attributes	Values
Account	AMEX
Project Code	AMX009Q4

conduit

Book Map Wireframe
diagram 3 Version1.0
February, 2009

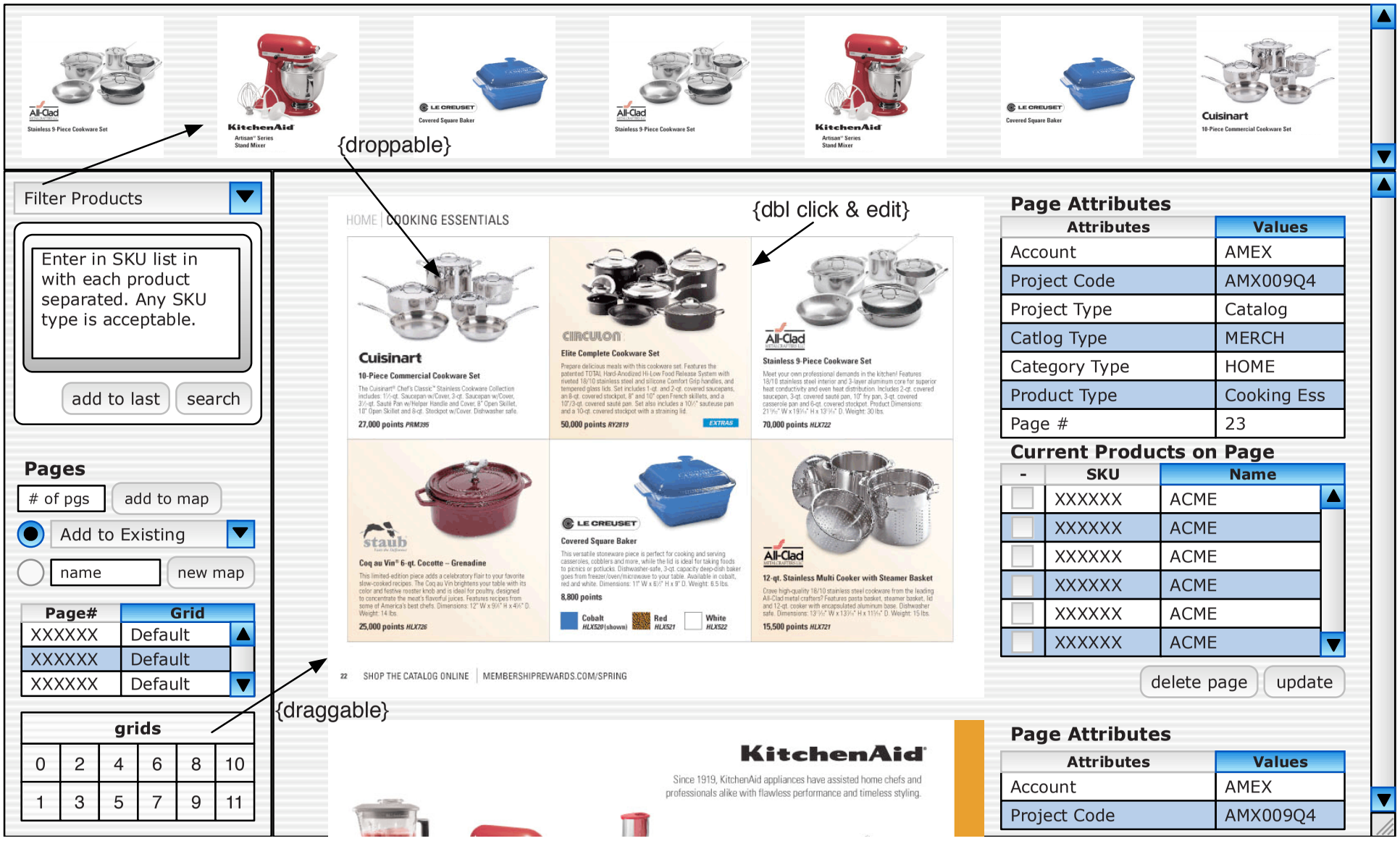
conduit™ range info

BookMap

Specialized subset of XML optimized to support the creation of multi-channel media items within conduit™.

Product Library

Page Map



conduit

Book Map Wireframe
diagram 4 Version1.0
February, 2009

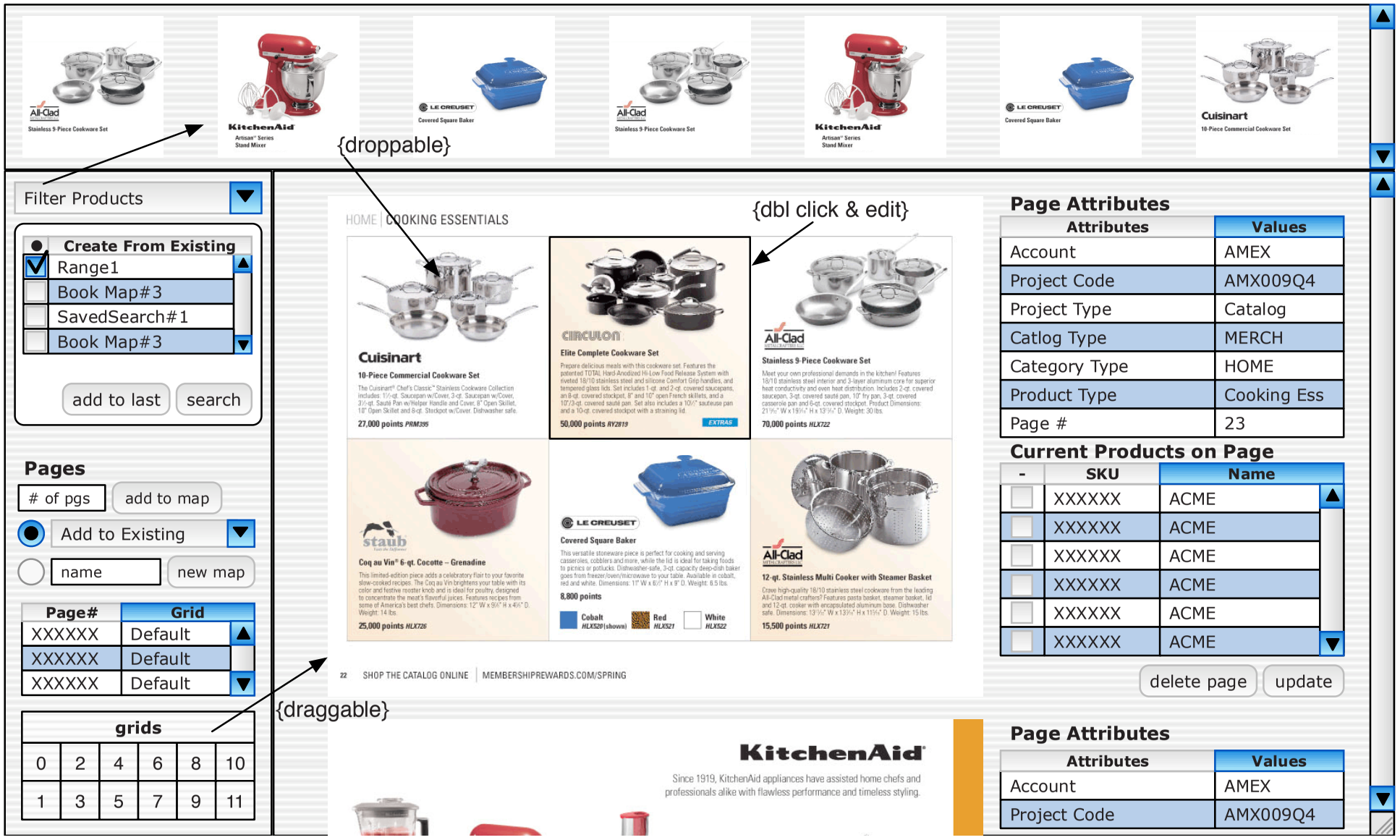
conduit™ range info

BookMap

Specialized subset of XML optimized to support the creation of multi-channel media items within conduit™.

Product Library

Page Map



Earth 2.X & beyond

Challenge

Challenges are yet to be defined. Three Items that have been identified are the extension of the Rendering Engine to output to Flash. The new interface solution cooperate ™, and the analytics package examine™.

Workflow

Add Flash component to Rendering engine: Conduit ➠ cSML ➠ cSTL ➠ Flash

More interaction with the system through more dynamic interface solutions.

Analyze the Logs and cull perinent information.

see concept sheet for cooperate™ on the following page.

create

For creative professsionals, it's work as usual, but better

With the use standard tools, the professionals design concepts, create layouts and produce pages. A simple to use plugin translates concept files into web layouts. It's easier than printing to a network printer or creating a PDF. With a click of the button a collaborative client interaction is initiated—Pushing concepts further, faster. Let's improve the way we work.

collaborate

A workstation, a connection, and a web browser, that's it

Users need only a workstation, a connection and a web browser combined with a brief introduction to interact with the system. An intuitive interface allows users to dynamically reposition elements in the layout, update content, and replace available images. The system tracks all interactions and users make comments as needed. Let's inform the way we think.

communicate

The key to any effective campaign is communication

Changes and comments are recorded, and where appropriate the edits are directly synchronized with the original file. Shorter, highly efficient feedback cycles depend on standardized reports that show user comments and file difference. Evolve concept into product into success—maximize everyone's bottom line.
Let's **cooperate™**

powered by HMI conduit

Project Timelines:

VENUS 1.1 1/6/09 8:30 AM 3/2/09 8:30 AM 7w 4d

OUTSTANDING ITEMS 1/6/09 8:30 AM 2/18/09 8:30 AM 6w 1d
EXPORT CSV fixes and download 1/6/09 8:30 AM 2/18/09 8:30 AM 6w 1d
CONFIRM all unsaved changes with in the current range before changing ranges 1/6/09 8:30 AM 2/18/09 8:30 AM 6w 1d
FIX bugginess with model logos 1/6/09 8:30 AM 2/18/09 8:30 AM 6w 1d
IMPORT recreation 1/6/09 8:30 AM 2/18/09 8:30 AM 6w 1d
ANY other outstanding bugs with CMS system 1/6/09 8:30 AM 2/18/09 8:30 AM 6w 1d

RANGE CREATOR 1/13/09 8:30 AM 2/3/09 8:30 AM 3w
Examine Wire frames 1/13/09 8:30 AM 1/14/09 8:30 AM 1d
Discovery FIND/SEARCH approaches 1/14/09 8:00 AM 1/15/09 10:00 AM 1.25d
Define and Write FIND/SEARCH functions 1/15/09 10:30 AM 1/20/09 1:30 PM 3.25d
Interface For Range Creator 1/13/09 8:30 AM 1/20/09 8:30 AM 1w
Attach Interface to FIND/SEARCH functions 1/20/09 1:30 PM 2/3/09 8:30 AM 1w 4.5d

CONTROL ROOM 1/13/09 8:30 AM 2/11/09 8:30 AM 4w 1d
Examine Wire frames 1/13/09 8:30 AM 1/14/09 8:30 AM 1d
Finish Control Room splash interface 1/13/09 8:30 AM 1/16/09 8:30 AM 3d
Attach Range and User info to appropriate components in Interface 1/15/09 8:30 AM 1/20/09 8:30 AM 3d
USER ADMIN 1/20/09 8:30 AM 1/26/09 8:30 AM 4d
INTERFACE 1/20/09 8:30 AM 1/22/09 8:30 AM 2d
Attach interface to USER admin funtionality 1/21/09 8:30 AM 1/26/09 8:30 AM 3d
Project Admin 1/26/09 8:30 AM 2/3/09 8:30 AM 1w 1d
INTERFACE 1/26/09 8:30 AM 1/28/09 8:30 AM 2d
Attach Rudimentary Project Admin funtionality 1/27/09 8:30 AM 2/3/09 8:30 AM 1w
Range Admin 2/3/09 8:30 AM 2/11/09 8:30 AM 1w 1d 1.2
INTERFACE 2/3/09 8:30 AM 2/6/09 8:30 AM 3d
Attach Range Admin functionality heavily dependent on Range Creator 2/5/09 8:30 AM 2/11/09 8:30 AM 4d

Dynamic Segmented Based Search in Range Editor 2/11/09 8:30 AM 2/18/09 8:30 AM 1w
Limited Role Based Controls 2/11/09 8:30 AM 2/18/09 8:30 AM 1w
BETA TESTING VENUS 1.1 2/18/09 8:30 AM 3/2/09 8:30 AM 1w 3d 1.1

VENUS 1.5 2/18/09 8:30 AM 4/14/09 8:30 AM 7w 4d

Outstanding Issues and Ongoing Tasks effecting 1.5 Release 2/18/09 8:30 AM 3/31/09 8:30 AM 5w 4d
Refactor & Bug Fixes to 1.1 2/18/09 8:30 AM 3/31/09 8:30 AM 5w 4d
IDML research, Template Creation, Element Mapping 1/13/09 8:30 AM 3/31/09 8:30 AM 11w

Role based architecture 2/18/09 8:30 AM 3/16/09 8:30 AM 3w 3d
Define all internal roles and examine possible external user roles 2/18/09 8:30 AM 2/24/09 8:30 AM 4d
Define interfaces, functions and fields to be assigned to user roles 2/24/09 8:30 AM 3/2/09 8:30 AM 4d 2.2.1
Design and implement ACLs 3/2/09 8:30 AM 3/16/09 8:30 AM 2w 2.2.2

Basic Account and Project Management Functionality 3/16/09 8:30 AM 3/31/09 8:30 AM 2w 1d 2.2, 2.2.3
Changes to Interface, Expose Account Setup (Pivot Fields, Segmentation, Import) 3/16/09 8:30 AM 3/24/09 8:30 AM 1w 1d
Attach account management functionality to interface 3/19/09 8:30 AM 3/31/09 8:30 AM 1w 3d
Basic Messaging 3/20/09 8:30 AM 3/27/09 5:00 PM < 1w 1d
INTERFACE 3/20/09 8:30 AM 3/24/09 8:30 AM 2d
FUNCTIONS 3/24/09 8:00 AM 3/27/09 5:00 PM 4d

WORKING WITH IDML with data from conduit™ 2/18/09 8:30 AM 3/31/09 8:30 AM 5w 4d
Lineup IDML structure with RAW XML from CONDUIT 2/18/09 8:30 AM 3/4/09 8:30 AM 2w
Script set to push XML to IDML and to push IDML package to IDS for page generation, and then to push pages to DAM. 3/2/09 8:30 3/31/09 8:30 AM 4w 1d
TEST and TROUBLESHOOT 3/16/09 8:30 AM 3/31/09 8:30 AM 2w 1d

PRODUCTION TEST VENUS 1.5 3/31/09 8:30 AM 4/14/09 8:30 AM 2w 2.1, 2.4, 2.3

Venus 1.9 3/31/09 8:30 AM 5/5/09 8:30 AM 5w

Bug Fixes, Refactor, Ongoing Concerns 3/31/09 8:30 AM 5/5/09 8:30 AM 5w
Ongoing Examination of Haggin Clients for integration into conduit system and workflow 3/31/09 8:30 AM 5/5/09 8:30 AM 5w

Image Interfaces 3/31/09 8:30 AM 4/13/09 8:30 AM 1w 4d
View Product Images 3/31/09 8:30 AM 4/7/09 8:30 AM 1w
Drag and Drop Logo Library 4/7/09 8:30 AM 4/13/09 8:30 AM 4d

Rendering Engine Expansion 4/2/09 8:30 AM 4/14/09 8:30 AM 1w 3d
Refactor and bolster XML to IDML engine 4/2/09 8:30 AM 4/14/09 8:30 AM 1w 3d
Created Reverse File 4/2/09 8:30 AM 4/3/09 8:30 AM 1d

Extend Project Management Features 3/31/09 8:30 AM 4/14/09 8:30 AM 2w

Extend Account Management Features 3/31/09 8:30 AM 4/14/09 8:30 AM 2w

Begin to integrate additional Haggin Clients into the system internally. 3/31/09 8:30 AM 5/5/09 8:30 AM 5w

EARTH 2.0 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d

Book Map 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d
Drag and Drop Pagination Tool 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d
Object Library 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d
Grid Options Development 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d
Output custom ranges to PDF, XML (for INDD, Web, email) 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d

Rendering Engine Expansion 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d
Implement output for PDF, email, HTML 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d
Expose Template Selector and manager 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d
Expose Preview for all media stream 5/5/09 8:30 AM 6/19/09 8:30 AM 6w 3d

Glossary of Terms

Definitions Coming Soon
- Alchemy™
  - Analytics
- architectural methodology
- application
  - attributes
    - business logic
    - client specific
- component
  - core technology
  - Custom Publishing System
  - DAM

development path
- dynamic style guide
- element

Elements of Operability
- maturation
- objects
- pre-templated
- program
- range
- Rich Internet Application
  - Segmentation

SMS principle
stickiness

versioning
- Versions
- Mercury
- Venus
- Earth

system
- Promote