

Jacquard: a methodology for Web publishing

Learn how to enhance communication in Web development and workflow

Skill Level: Intermediate

[Uche Ogbuji \(uche@ogbuji.net\)](mailto:uche@ogbuji.net)

Partner
Zepheira

18 Mar 2008

Learn about Jacquard, a software development methodology specialized for Web projects, and especially for Web development among diverse teams. Jacquard looks to align the work and goals of business interest personnel, Web designers, programmers, project managers, database analysts, and more. Learn about the core principles of Jacquard, and follow an example of its use in communication between a user experience team and a programmer team.

The great thing about the Web is its diversity. You can develop and manage a Web site with a seemingly endless array of tools and techniques. But this can be a headache for the interplay between the different people who have to contribute to a site. User Experience (UX) experts and Web designers try to make the visual elements as attractive, and ideally, as accessible as possible. Programmers and database experts feed the design with dynamic data that forms the meat of the site. Project Managers and line-of-business personnel (I call these "the business interest") worry about the costs, schedule, and overall value of the result. Such functional sub-teams have to work closely together to make the project work, but the difference in perspective can make it hard to collaborate closely without good standards for communication. Successful teams do manage to invent ad-hoc standards for exchange and communication, but other information disciplines have developed industry standards that help improve efficiency in such situations, and Web development teams could benefit from the same.

I have a long background in software and data architecture, focusing on Web-based

and Web-like systems. In this experience I've seen—and sometimes created—various approaches to the integration of the work products of Web teams. The spectrum runs from traditional enterprise methodologies such as Rational Process, to adaptations of agile methodologies such as Extreme Programming and Scrum. But I think Web development is unique enough to merit its own methodology. I've been developing such a methodology, which I call Jacquard, after the man some people consider the first computer programmer. Joseph Marie Jacquard created a device which could be affixed to looms, which used a pattern of holes, much like early computer punched cards, to direct the textile design. The connection between art and technology that led to this pioneering development in informatics is still alive and well in the area of Web development, and is thus a fitting metaphor for systems that weave sites through the efforts of diverse teams.

What should a Web development methodology feel like?

The Web is in many ways different from any information platform before it, and this suggests a fresh approach to development and teamwork. In general it makes sense to look outward to the Web, and not inward and backward to traditional methodologies, to find what works. Lightweight, agile process mirrors the basic nature of the Web, and so does focusing on the data, and how data is organized for sharing. The specific application or database implementation is not as important, nor are the tools you choose to use. This mirrors the Web, which builds on sharing data, and does not require uniformity of implementations. As such, **implementation independence** is one of the core principles of Jacquard.

Another principle is **support for decentralized communication**. The Web works well across geographical boundaries, and with the increase of off-shore outsourcing and flexible work arrangements, it's useful to learn lessons on decentralization and rich communication. The Web is such a rich information space that some philosophically consider it a realm of its own which parallels, and sometimes intersects, our own real world—the idea of "cyberspace." Paying attention to where idioms on the Web draw from real-world concepts and phenomena is important to usability, and so Jacquard's principle of **conceptual alignment** encourages you to take care to express the concepts behind your Web project, and to make that clear expression the foundation for communication on the project. This is the principle that I explore most fully in this article. Many of the other principles are shared with other agile development methodologies, such as **frequent iteration**.

Putting Jacquard to work

Let's say you are a snowboard manufacturer, Fluffy Boards, and you are developing a promotional site for a new model of board, called the Cumulus. The idea of Jacquard is to take a real-world scenario and simulate it in the presentation and behavior of your Web site. The business interest personnel for the snowboard

articulate the scenario for the site, and the data architect captures the most key concepts, such as:

- product
- endorsement
- customer
- customer review
- media
- media review
- recommended reseller
- featured reseller
- specification

Role of the data architect

The data architect plays a central role in a Jacquard project. He or she is responsible for making sure the data modeling and documentation is suitable for strong communication among subgroups of the team, with their varying perspectives, and especially between the business interest personnel and others. In a small project, the architect might also play other roles, but in larger projects it's important to have a dedicated architect.

Formal outline of the core concepts

The Jacquard methodology requires formal expression of the core concepts in a way that can be a shared reference across the various teams. It doesn't mandate any specific system for such expression. In this article I'll use the W3C's Simple Knowledge Organization System (SKOS), which is a very useful technology for the expression of ideas in a way natural to humans, but in a very Web-ready format (RDF). I use the Turtle syntax for RDF, which is easier to read than RDF/XML. You would still probably want to have business users use a friendly SKOS browser, rather than squinting at this text file, but the underlying detail would be the same. Listing 1 is a subset of the SKOS for the concepts listed above.

Listing 1. SKOS definition of key concepts for the Web project

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.
@prefix f: <http://www.fluffyboards.com/vocabulary#>.

f:product
  a skos:Concept;
  skos:prefLabel 'product' ;
```

```
skos:altLabel 'merchandise item';
skos:definition 'Item developed for sale by Fluffy Boards.';

f:snowboard
  a skos:Concept;
  skos:prefLabel 'snowboard';
  skos:altLabel 'deck';
  skos:definition 'Deck to be mounted with bindings for riding on snow.';
  skos:broader f:product;

f:endorsement
  a skos:Concept;
  skos:prefLabel 'endorsement';
  skos:altLabel 'formal thumbs-up';
  skos:definition 'Formal statement of approval of the product.';
  skos:broader f:review;

f:review
  a skos:Concept;
  skos:prefLabel 'review';
  skos:altLabel 'product opinion';
  skos:definition 'Statement of opinion of a product.';

f:customer
  a skos:Concept;
  skos:prefLabel 'customer';
  skos:definition 'Person or group engaged by Fluffy Boards in the purchase process.';
```

I'll just give a very brief overview of this format. The first two lines are namespaces declarations, much like those you've seen in XML. These are basically a means for organizing the concepts. Terms starting with `skos:` are built into the SKOS system, while those starting with `f:` are terms defined by the fictional example company Fluffy Boards. In the next block of code I define the concept `f:product`, defining its preferred label, as well as alternate labels. `skos:definition` is a brief but precise description of what the concept is. The `skos:broader` property is a way of organizing related concepts. "Person" is a broader concept than "boy," and "boy" a narrower concept than "person." SKOS includes properties for expressing these and other relationships between concepts, so it's a useful means for organizing the information space for the Web project, and meeting the formal expression mandate of Jacquard.

Exploring the wireframe

The main example of how such formal definition of Web project concepts provides practical benefits is in the interaction between the User experience teams developing Web site wireframes, design guidelines, and such, and the teams of programmers who make the dynamic features of the site work. The UX team develops wireframes (the presentation for the site), structured to reflect the problem space, and puts them through a user test regime. I'll use a simplified example wireframe, Listing 2, for a "Fans of the Cumulus" page, to gather reviews, endorsements, and so on, with the goal of encouraging sales of the product.

Listing 2. Wireframe for a Cumulus marketing Web page

```

<html>
<head>
  <meta content="text/html; charset=iso-8859-1" http-equiv="Content-Type"/>
  <title>Fans of the Cumulus</title>
  <link href="main.css" type="text/css" rel="stylesheet"/>
</head>
<body>
  <h1>Fans of the Cumulus</h1>
  <div id="testimonials"><!-- List of f:endorsement -->
    <div class="testimonial"><!-- Instance of f:endorsement -->
      <div class="quote"><!-- f:quote property of f:endorsement -->
This board makes me feel like I'm lounging a happy path through sweetened butter
      </div>
      <div class="source"><!-- f:source property of f:endorsement -->
Moe Bettasno
      </div>
    </div>
  </div>

  <div id="reviews"><!-- List of f:review -->
    <div class="review"><!-- Instance of f:review -->
      <div class="quote"><!-- f:message property of f:review -->
Hey we had a powder day at A Basin last weekend, and I can't believe how easily
I floated through the trees on this thing
      </div>
      <div class="from"><!-- f:customer (f:from property of f:review) -->
<span class="from">Joe Snow</span><!-- f:name property of f:customer -->
<span class="from">Denver, Colorado</span><!-- f:where property of f:customer -->
      </div>
    </div>
  </div>

</body>
</html>

```

The annotations in comments are the key to communicating with the developers who will write the database queries to fill in actual endorsements, reviews, and so on. Notice that page elements such as the `div` IDs and classes use terminology from the concept definitions, and that the annotations use a direct, formal expression of the terms. It's possible to use even more formal language for such annotations, but once you have enough clarity in the description of the core concepts, you can be a bit more relaxed in some of the other artifacts. Simplified English with precise references to formal terms is sufficient in this case. There are also a few terms in the wireframe annotations which are not in Listing 1, and this is only because I abbreviated that listing for this article.

Another key principle of Jacquard is that business rules are expressed in semi-formal or formal language, and are managed as the core concepts are managed. UX experts would accompany wireframes with a set of such rules relevant for each page, for example:

- There should be no more than three `f:endorsement` instances on the page.

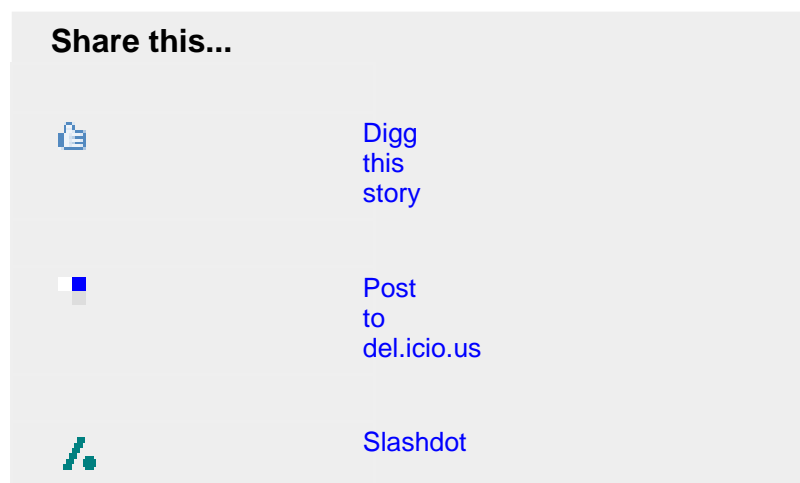
- There should be no more than ten f:review instances on the page.
- The f:review instances on this page should be limited to those with an f:about property indicating the Cumulus.

There are some interesting, formal languages for such rules, including Attempto Controlled English (see [Resources](#)), but again simple English (or whatever language is used in the organization) with clear references to the concept definition is good enough for most projects.

Like many agile methodologies, Jacquard recommends iteration as a means of refining the work product. In the process of developing the wireframes, UX experts will almost certainly introduce more core concepts than in the original business analysis. The back and forth between programmers and UX experts will shake out even more additions and refinements. Don't be shy about incorporating such changes, and use tools such as version control to avoid chaos. The iterations will also result in some more peripheral, site-implementation concepts which are useful for communication between UX experts and programmers, but might not be of interest to other parties. It may be useful to separate these into subsidiary documents, to avoid clutter.

Wrap up

There is much more to Jacquard than I was able to cover in this article, but I hope you get a sense of the methodology and how it can help streamline your Web development efforts. Jacquard is an open, community-developed methodology, so if you have thoughts or experiences that you can share to help others, please pitch in. The idea of Jacquard is to communicate clearly, but naturally, throughout all phases of the project. It may sound simple and even obvious on the face of it, but it's amazing how rarely projects deal with the important issue of how to communicate among Web project team members, so use Jacquard to help ensure that you get this right.





Resources

Learn

- Jacquard is a community project, hosted on [a wiki](#).
- SKOS can be used for organization of large content libraries, as described in "[Subject classification with DITA and SKOS](#)", by Erik Hennum, Robert Anderson, and Colin Bird, or it can be used in a less ambitious way, as described in this article.
- Attempto Controlled English is a rich subset of standard English designed to serve as specification and knowledge representation language, including for business rules. Learn more about [Attempto](#).
- Turtle is a non-XML syntax for RDF. You can learn it from the very readable [specification](#).
- Learn more about [Joseph Marie Jacquard and his loom device](#)
- Stay current with [developerWorks technical events and Webcasts](#).
- Expand your site development skills with articles and tutorials that specialize in Web technologies in the developerWorks [Web development zone](#).

Get products and technologies

- Download [IBM product evaluation versions](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.

About the author

Uche Ogbuji

Uche Ogbuji is Partner at [Zepheira, LLC](#), a solutions firm specializing in the next generation of Web technologies. Mr. Ogbuji is lead developer of [4Suite](#), an open source platform for XML, RDF and knowledge-management applications and lead developer of the [Versa](#) RDF query language. He is a Computer Engineer and writer born in Nigeria, living and working in Boulder, Colorado, USA. You can find more about Mr. Ogbuji at his Weblog [Copia](#).

Trademarks

IBM, WebSphere, DB2, Lotus, Tivoli, and Rational are trademarks of International Business Machines Corporation in the United States, other countries, or both.