Domain Driven Design For Websites
A state-of-the-art survey.

Version of 06 Dec 2011

IAWEB Group 1 - Christian Kaiser, Rene Kaiser, Nadja Lauritsch and Lisa Tomes

Abstract

This paper introduces the Domain Driven Design approach and discusses how to develop websites with this approach. DDD is not only a method of presenting the important objects of a domain together with their relations, it is a paradigm which leads to better results by combining a set of well-known principles of other domains. The main focus is on the model, and therefore knowledge of the domain is essential. A basic principle in this context is to "think things not documents", to not think about the website when creating the model with domain experts. The challenge is to get knowledge from experts and end-users, and to develop a ubiquitous language doing so. DDD also focuses on URI structure design and a suitable layout. In this paper it is discussed how to develop a website with the Domain Driven Design approach. The BBC and a County Council's website are two case studies which are described. The paper gives an overview how to evaluate the domain model with methods like Focus groups, the Greeking Test and the Thinking Aloud test.

Contents

Chapter 1

Introduction

The real challenge of complex website creation projects is not the technical realization, but the conversion and the mapping from the real world to a website. This chapter is a short introduction to the Domain Driven Design approach.

"There is technically nothing new or revolutionary in DDD, there is only a guide to a better way of thinking."
Casey Charlton

1.1 Domain Driven Design

Domain Driven Design (DDD) is a technique of representing the objects within a subject and the relationships between those objects. The goal is to build better websites by focusing on the model underneath it. The domain is the subject area, for example considering an online book store, the domain is selling, storing and shipping books. A model describes some parts of the domain and it is a set of concepts, which are implemented on the website. The most essential aspect is to really understand the model well. DDD is independent of the technology used. The term itself was coined by Evans [2004] in the fundamental book "Domain-Driven Design: Tackling Complexity in the Heart of Software".

However, how to design the next project with DDD approach? In the beginning it starts with an simple example: A developer and a domain expert come together to discuss a new project. Next they start a dialog where the developer and the domain expert talk in their jargon. The problem with this conversation is that both are talking too specific in their special field and there is misunderstanding preprogrammed. The first step is to crunch the knowledge. Due to the different language they talk, it is essential to find same vocabulary. A ubiquitous language plays a key role, due to it helps to solve the users domain-related problems. Ubiquitous language means universally present language. Scenarios can be described and discussed with the project members using the shared language. The scenarios are a kind of prototype for the domain experts. They are very powerful, due to the details can be discussed which results in knowledge of the model. The critical aspect of most projects is in understanding the domain itself.
With the following steps the domain can be modeled:

1. *Binding the model and the implementation* — With the prototype the essential link has to be forged, the relevant information has to be gotten and it has to be improved during the subsequent iterations in an agile development.

2. *Use the ubiquitous language* — The users and the developer have to explain their specific vocabulary. The goal is to find a language without any misunderstandings.

3. *Developing a knowledge-rich model* — The objects have behavior and follow rules and shouldn‘t be only a data schema.

4. *Distilling the model* — Analyzing the domain for getting information for modeling.

5. *Brainstorming and experimenting* — A brainstorming attitude is a very good mindset for developing the model.

Domain Driven Design is a difficult technical challenge which can pay off, opening opportunities. For the developers this means improving technical knowledge, but also improving on the specific domain knowledge. DDD is oriented to agile development and a methodology with the following focus:

- Ubiquitous language
- Collaboration between the domain experts, testers and users
- Behavior of the domain objects
- Bounded context of the responsibility of the objects

For introductionary literature the authors recommend Evans [2004], Abel and Floyd [2006], Khan [2009] and Laribee [2009].

1.2 Problems That May Occur at a Naive Try

What is happening without applying the Domain Driven Design approach? A project without totally focusing on the model leads at the first time to more success, due to it is fast at the beginning in designing, but the expectations for future development were very high. The main problem here is the unstructured design and also no common language according to Evans [2004]. There are many aspects, which influence the project and put it off course:

- Failure to communicate
- Complexity
- Objectives changing during the project
- Bureaucracy
- Unclear objects
- Lack of resources

It starts with the language, when the terminology is disconnected from the embedded code or HTML. The technical members also use different language in speech and writing. In every model it is important to play with the model as to talk about the system. It helps to have a better understanding about the system, when it is talked about different scenarios. Many complex projects do attempt some sort of a domain model, but they don‘t maintain a tight connection between the model and the implementation. To understand the domain is important in every model, due to without a deeper understanding the websites can not be modeled. By using DDD it is not essential how the source code looks like, the question is how to get the source code (HTML etc.). The most fundamental concept is to develop a model with close collaboration between the developer and the domain experts. It is a way to handle complex business logic.

1.3 Structure of this Report

The report is structured as follows. In Chapter 2 the Domain Driven Design approach for structuring content on websites is discussed in a step-by-step manner. Technical details of how to construct and represent a domain model using OWL and Semantic Web technology are discussed in Chapter 3. Another key aspect, the iterative testing of the model and website with users, is presented in Chapter 4. Chapter 5 discusses selected case studies where Domain Driven Design has been applied successfully. The final Chapter 6 briefly summarizes the key lessons of Domain Driven Design from both a theoretical and practical perspective.

Chapter 2

The Domain Driven Design Approach

This chapter is discussing, in general terms, how to apply the Domain Driven Design approach to Web site creation, to the question of how to structure and present Web content. The following consists of a number of key aspects, recommendations and principles which are mainly based on Mike Atherton's[1] slide set "Beyond the polar bear" (Atherton [2011]), on material of Rissen et al. [2011] and of Michael Smethurst[2]. And, not surprisingly, on Eric Evans' fundamental book on Domain Driven Design (Evans [2004]).

2.1 Approach

Domain Driven Design (DDD) is about user experience design and interaction design. The aim, in our context of creating a Web presence, is to structure the content and define the structure of the (interlinked) websites. A consistent user experience (UX) consists of a shared model, language and understanding. User experience goes far deeper than presentation and interaction and has to be carefully developed (see Atherton [2011] slide 88 ff.).

A basic principle in this context is *'think things not documents'*. The domain model engineer should model things of the real world and not Web pages. However, blanking out from one's mind all thoughts of the resulting Web site is a challenge. The outcome should be reflected iteratively throughout the lifetime of the project to refine the understanding of the domain. The Domain Driven Design approach is not limited for websites of a certain minimum complexity and should be considered to be applied for projects of all sizes, not just large site networks such as those of the BBC[3]. In brief, what is domain modeling (Atherton [2011] slide 18 pp.)?

- A way of representing the important 'things' within a subject, and the relationships between those things
- A way of using the subject knowledge of users and experts to influence software design
- The first state in the BBC's process of designing new websites, and one of the few tangible artifacts
- Inspired by Eric Evans's 'Domain-Driven Design' book

At first the domain has to be explored. Identifying the domain should be straightforward as it can be derived from the project's requirements and goals. To gather information about the domain, it is essential to employ domain experts and integrate them in the design process at an early stage. Besides experts, typical users should be integrated in the process as well. More on that will follow in Chapter 4. When talking to domain experts, it is important to use *their* language. Domain experts do not need to have technical knowledge, and often they are sparsely available for inputs. The domain model engineer has to capture what the experts consider important. How would they search in their domain? The main aim is to capture their mental model. After the experts shared their domain knowledge, it is useful to sketch the understood back to them in order to rule out

[1]`http://www.reduxd.com`, `http://www.slideshare.net/reduxd`
[2]`http://www.slideshare.net/fantasticlife/`
[3]Practical case studies will be discussed in Chapter 5.

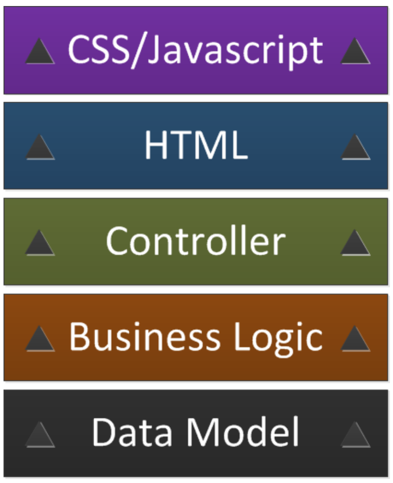


Figure 2.1: A bottom-up approach, redrawn from Atherton [2011].

misinterpretations. In other words, the key aspects of developing the model are, according to Atherton [2011] slide 81 pp.:

- Talk to experts and end-users
- List the important things and sketch relationships
- Develop a ubiquitous language
- Think only about the mental model, not web pages

Creating the model is not a trivial process, pros and cons for modeling relations one or the other way have to be weighed. A frequent problem is when for example multiple versions of a masterpiece exist and belong to that *super concept*. If the pages of the different versions are not connected, the users might not get what they want. A good approach would be to have a hub page where all versions are listed and linked.

It is recommended to think bottom-up *everywhere* in the design process and start with the domain model, working the way up to the layout and interaction aspects of the website. See Figure 2.1 for an illustration of such possible layers. Mike Atherton recommends to

> "Design for a world where Google is your homepage, Wikipedia is your CMS, and humans, software developers and machines are your users."

For human users especially, the shortest route to well-organized content has to be enabled, which can only be achieved by careful design. Usually, there are numerous ways how to structure content, with different pros and cons. The optimal categorization for the specific users has to be found. The link structure has to correspond with the users' mental model to be understood. A *breadcrumb navigation*, as discussed by Bonnie et al. [2003] and Nielsen [2007], is often beneficial for a website.

To continue building pages, the following recommendations are given in Atherton [2011], slide 84:

- Start with simple stub pages for all objects
- Think about all representations, not just HTML
- Design the document to put the most important things at the top
- Consider requirements for caching and load-balancing

Michael Smethurst discussed the aspect of *Designing for your least able user* in both a talk (Smethurst [2009a]) and a blog post (Smethurst [2009b]). He recommends to think of search bots as the least able users, which index the content and metadata of each page. The content of *meta elements* is important, but nowadays outsmarting search engines is almost impossible. Smethurst recommends to stick to the common rules for accessibility and avoid Flash for navigation. While the evolution of the Web from a static collection of HTML pages to more dynamic channels, where users become publishers (Web 2.0), enabled a lot what could not be done before, JavaScript and AJAX can be an accessibility disaster. The reason is that when such technology enables to change the (view on) content in the browser without a full page reload, it is tough if not impossible to assign clear URIs so that the state can be restored. If it is used, other browsable/crawlable routes to content should be provided.

2.2 Developing a Model

To then actually create the domain model, it is recommended to go through the following process, based on Smethurst [2009c]:

- Identify the domain objects
- While chatting and sketching with the domain expert a picture of the types of things they are concerned with should be built. Make a list of these objects
- Identify the relationships between the domain objects
- As the knowledge of the domain increases a picture of how the objects interlink will be built.
- Sketch basic entity relationship diagrams with the domain expert, keep sketching until the picture clears
- The resulting domain model will inform the rest of the project and should be one of the few 'artifacts' the project ever creates
- Design the database
- Translate the domain model into a physical database schema, make the models
- This means they should capture all the constraints of the database schema plus all the extra constraints of the domain model
- Make *hello world* pages for the primary domain objects (name), aggregations (linked list)

On how to model the concepts and relationships of the domain model and which technology to use, details will follow in Chapter 3. The links between the content items are key to explore and experience. To be able to present content in a useful way both end-users and experts have to be asked, as stated before. If there is a ubiquitous language created, it

> "Defines a common lexicon used by all members of the team, at all levels of the design." (Atherton [2011])

The question how the content is accessed has to be reflected. Is it watched by desktops or mobile devices? Will those users see completely different versions or are there just slighter differences? How will visitors usually jump to the site? Will they jump right into the middle of the site by clicking on search engine results (*deep link*), or mainly browse their way through the site, starting at the homepage? Are they only readers or can they contribute as well (Wiki)? It helps to think of a website as a coherent whole, not a collection of individual subparts. The internal organisational structures through the website are not wanted to be exposed, however, such influences are difficult to rule out in the model creation process.

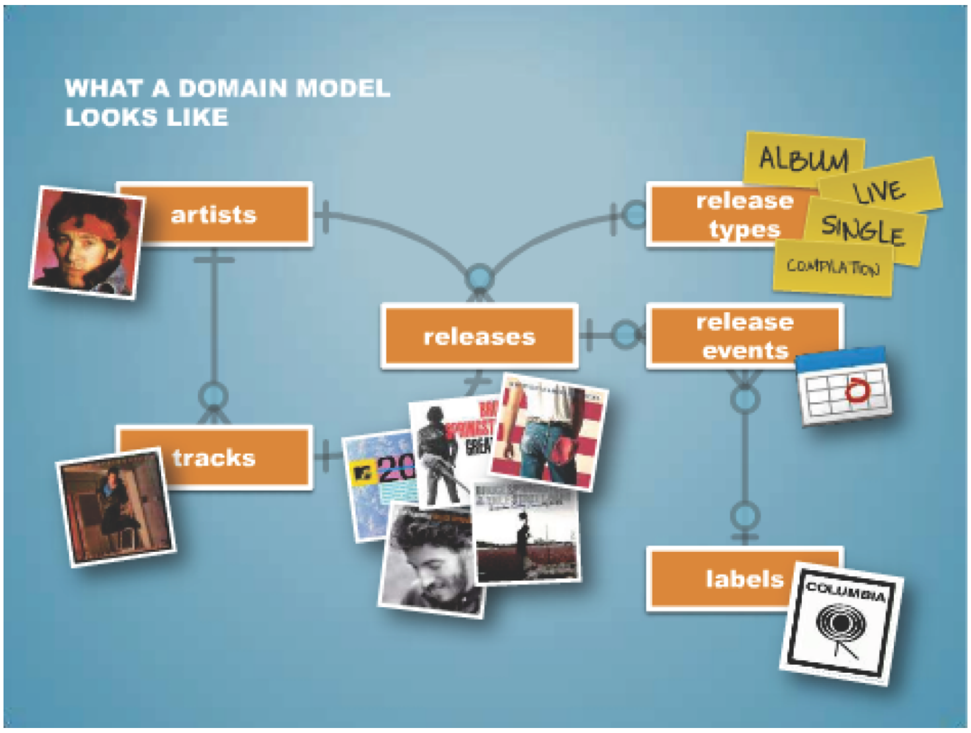


Figure 2.2: Simple example of a domain model, from Atherton [2011].

2.3 Checking the Domain Model with Users

The following is based mainly on Smethurst [2009c]. Details of domain model testing with users will be discussed in Chapter 4.

Once a draft of the domain model is created, it is essential to speak to typical users and ask for their feedback, to get them to sketch their understanding. Then as suggested improvements are implemented, the process should be iteratively repeated. Speaking to them is essential, for example in focus groups where they are free to express their opinion in a controlled interview. There might be differences between the experts' preferred model and the typical users' preferred way of structuring the content.

> "User-centric design starts here — if you choose to model things and relationships between those things that users can't easily comprehend no amount of wireframes or personaes or storyboards will help you out." (Smethurst [2009c])

A key aspect to be evaluated with users is the interlinking with existing content or content outside the website. When extending an existing network of websites, it is important to check if some of the domain objects are already covered by existing pages. If that is the case, it should be linked to these pages instead of duplicating them. In accordance with the principles discussed in Section 2.5, keep the existing URIs, otherwise both your users and search engines will be confused.

2.4 Populating the Model with Data

Once there is enough confidence in the model's quality, the next step is to populate the model with actual data. For that process, Atherton [2011] recommends on slide 82:

- Work with a developer to translate the domain model to a data model[4]

[4]If it does not exist already from previous steps.

- Look at the business data and reshape if necessary
- Identify sources of linked data to enhance the product
- Do not reinvent where someone else has done the thinking

Aggregation and linking is essential, and embedding complex data such as media is a topic of its own. Getting the data right is not straightforward (Atherton [2011] slide 35). Data from various sources differs in quality, different labels might be used. The data has to be fit with the domain model. Data usually comes from business systems both inside and outside the organization (cp. Smethurst [2009c]). In the latter case, investigation about licensing, Open Data, and APIs is necessary.

According to Smethurst [2009c], at first the definition of the data needed to build each of the pages has priority — for each URI and all representations.

> "Just because the HTML representation does not need to show the updated date does not mean the RSS or Atom or RDF do not need it."

The next step is to build up the HTML pages and other representations, and to make sure to design the document to be semantically correct and accessible. At this stage, try to avoid to think about the layout of HTML pages — that is the job of CSS, discussed in Section 2.6.

> "In general your page should be structured into title, content, navigation — screen readers do not want to fight through calendar tables etc to get to the content."

Another strategy to decide is the caching mechanism. It has to be guessed how long content can be cached. If it is cached too long, users might get annoyed with outdated information. Not caching long enough will result in unnecessary traffic which places extra strain of the application. Cached pages will be rendered faster by browsers, improve user experience and save the mobile users connection fees. As an example, think of an online TV schedule. It is not wanted to cache today's items heavily since they are subjects to change. However, there is little problem with caching yesterday's program.

2.5 URI Design

A really important — and often neglected — part of designing a website is URI design. Those little character strings are used by visitors to remember, bookmark and link to a site. The most important feature of URIs therefore should be persistence. It is good if they are also human-readable, but persistence should not be sacrificed for it (cp. Smethurst, Michael [2009] and Smethurst, Michael [2009]).

It is often a good choice to assign unique identifiers to resources by which they can be referred to. For instance a URI of a football club's website may look like this:
`http://www.cool-club.at/trainer/huber`. However, what if the trainer named 'Huber' marries and henceforth is named 'Unterhuber'? Is it wise to change the URI to
`http://www.cool-club.at/trainer/unterhuber`, risking many broken links and bookmarks or just to leave it the way it is, risking confusion of the users of what the name of the trainer actually is?

For that kind of problem it is convenient to create unique identifiers for all trainers, players and referees of the club and design URIs the following way:
`http://www.cool-club.at/trainer/7dnc993h5b`. That may not be as human-readable as before, but guarantees persistence. As Smethurst, Michael [2009] says:

> "Cool URIs don't change."

It is possible to create HTTP 301 redirects to make URIs more human-readable, although this should be minimised for Googles *PageRank* will only work for *one* 301.

Another nice feature of URIs is hackability. If possible, URIs should be designed hierarchically in a way that cutting off the end of it (up to a slash character) leads to a page further up in hierarchy. For instance `http://www.cool-club.at/trainer` should lead to a page where all trainers are listed and not produce a 404 error.

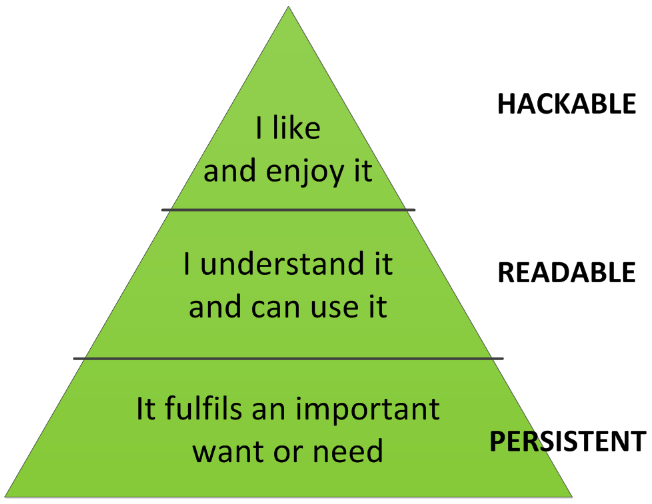


Figure 2.3: URI design, redrawn from Rissen et al. [2011].

Since many requests nowadays come from mobile browsers, the Web server should return, if possible, an appropriate representation (for example one URI for desktop browsers, one for mobile browsers). It is problematic to expose the technology stack by adding for instance *.shtml* to the URI. Users can be offended by it and it might change in the future.

For sake of Search Engine Optimisation (SEO) keep in mind “One URI per concept; one concept per URI”. Search engines, such as Google, can point to a site much more precisely if one of the pages deals with exactly one topic, in the best case the one the user was searching for.

2.6 Layout & Interactivity

At this stage of the process it is time to think about the layout and interactivity of the website. Both are essential for the user experience, however, certain browser limitations have to be respected. Appealing layout is a central aspect of user acceptance. Interactivity can be achieved by adding functionality using JavaScript or AJAX, for example. The following is mainly based on Smethurst [2009c] and Atherton [2011].

To apply layout and décor CSS:

- Layout the wireframe pages directly in the browser using CSS
- Design for the least-able user first
- Practice responsive design and use media queries
- Experiment with different layouts for the markup by moving elements around the page

The visual design and branding at this stage might be influenced by company guidelines or *corporate identity* defaults. The process using CSS should be exactly the same as taking a paper wireframe and applying design treatments over the top. The designer will experiment with different treatments to see how far the design can be stretched with the markup given. Part of the process is the creation, adaptation (for example using Photoshop) and placement of artwork — background imagery for headers, dividers, buttons, list items etc.

It is recommended to design a browsable site first and only then add in JavaScript/AJAX over the top. In this order, there is a better chance of making an accessible web site, one that degrades gracefully. As mentioned above, designing for search engines as the least able users is a good idea. Search bots in general do not like forms or JavaScript. Sites that degrade well for accessibility also degrade well for search engines. According to Smethurst [2009c], making every subsidiary resource addressable and providing these resources serialized as XML or JSON makes adding AJAX relatively trivial.

> "Design in the absence of content is not design, it's decoration." (Jeffrey Zeldman)

If wanted, the website can then progressively be enhanced using (for example) JavaScript and AJAX (Asynchronous JavaScript and XML). AJAX allows to implement asynchronous calls from the browser to the Web server and to refresh only parts of the page. However, it should not be mandatory to use — a share of the users will visit the site with JavaScript disabled.

2.7 Iterative Testing

At this stage check how search engines list the pages and subsites for certain search terms. Which metadata of the page is displayed in the search results? The model development should be iterative, testing user understanding throughout, additionally verifying with experts, checking against requirements and available content. Iterative testing is key in a Domain Driven Design process — from Atherton [2011]:

- Test at every stage, from the domain model upward
- Test not just interaction, but the conceptual model itself
- Test with real people, not (only) against personas or user stories

According to Smethurst [2009c], testing with real users at every stage of development is essential, but it is particularly important to conduct both usability *and* accessibility tests. This is similar to testing with traditional wireframes but it is on the real application with real application behaviors and real data (no lorum ipsum). Testing results may require changes to layout CSS, sometimes to markup, sometimes to the data, and even to the underlying domain/data model. It might even mean renegotiating contracts with data providers outside the company — all design and usability issues can be fixed — some just need more lawyers than others.

> "Bear in mind if you are using data from existing business systems there may need to be heavy investment to make changes to that database model."

The key here is to *really* test and test again iteratively as long as there is a reason to improve, following the same steps for each development cycle. Different sorts of (accessibility) tests have to be made, and everything, from the model to the décor, may be changed. During the process, JavaScript should be turned on and off. The model might even be expanded at a very late stage. Within a reasonable scope, never hesitate to throw away!

Chapter 3

Engineering the Domain Model

The Web is connected like a graph. Links to resources both inside and outside the site/domain are an essential part of it. The idea behind it is that existing data should be referred to instead of re-inventing the same content over and over again. The same applies to the level of domain models: linking instead of re-inventing, use of existing models instead of replication is preferred wherever possible.

The Internet as such is realtively young but has seen a number of paradigm changes already. Starting from a static pages, the Web2.0 has turned users into prosumers and made content more interactively accessible. Since, research focused on different forms of the *Semantic Web*. Primary routes to content have shifted from silos to aggregators (cp. the *walled garden* in Figure 3.1).

3.1 Modelling a Domain Using RDF/OWL

When modeling the knowledge about a domain as a graph, a format is needed to represent it, and a tool is desired to define the concepts and relationships of it. Complexities of knowledge call for ontological structures and connections with relationship values are needed. Re-using content (linked data) across domains unlocks value (cp. Atherton [2011] slides pp. 98-99).

The RDF/OWL format is nowadays a de-facto standard to model ontologies. It is a very generic format where the structure of the graph is defined *with* the data in the same format, not separatedly (cp. W3C [2011b]). There is even a query language for RDF (cp. W3C [2011c]). For definitions and details, it is referred to the thesis of Pammer [2010]. In the realm of Semantic Web research, Protégé — *The Protégé Ontology Editor and Knowledge Acquisition System*[1] has been developed. See Figure 3.2 for an actual screenshot. In Hruby [2005] a method for designing domain-specific models, based on the use of domain ontologies, is presented, though with software development as task behind.

[1]`http://protege.stanford.edu/`

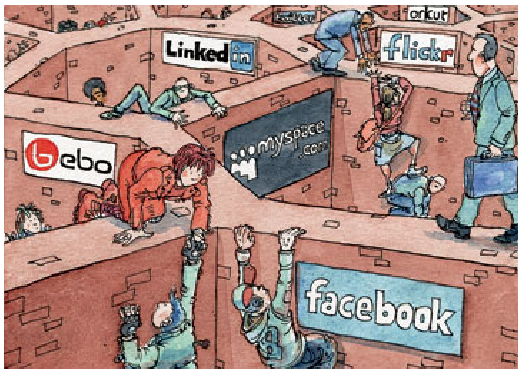


Figure 3.1: Walled gardens, illustration by David Simonds, `www.economist.com`.

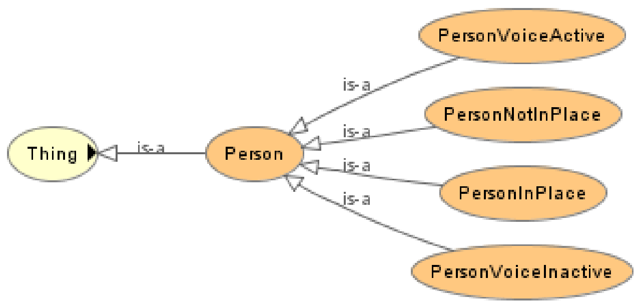


Figure 3.2: Low-resolution screenshot of Protégé, from Kaiser et al. [2011].

3.2 Ontology Engineering

Ontology engineering is still more of a science than an engineering field. Going beyond proof-of-concept research prototypes, practical problems appear. Most work involves only read-only access and a common mechanism for updating and extending distant Open Data repositories is missing. Such a mechanism would imply all sorts of issues such as spam and misuse which then in turn have to be dealt with themselves.

Ontology model examples can be found all over the Web, as standard ontologies exist for a number of domains. As an example, a closer look at the BBC Wildlife Ontology[2]. Corresponding to the website, it is a simple vocabulary for describing biological species and related taxa. The vocabulary defines terms for describing the names and ranking of taxa, as well as providing support for describing their habitats, conservation status, and behavioural characteristics, etc. It is licensed under the Creative Commons Attribution License.

The BBC Wildlife Ontology consists of the classes and properties listed in the Listings[3] 3.1 and 3.2.

```
Adaptation AnimalIntelligence BehaviouralPattern Class Collection
    CommunicationAdaptation ConservationStatus EcosystemRole Ecozone
    ExtremesAdaptation Family FeedingHabit FreshwaterHabitat Genus Habitat
    Infraorder Kingdom LifeCycle LocomotionAdaptation MarineHabitat
    MorphologyAdaptation Order Phylum PredationStrategy RedListStatus
    ReproductionStrategy SocialBehaviour Species Suborder Subspecies
    Superclass Superfamily Superorder SurvivalStrategy TaxonName TaxonRank
    TerrestrialHabitat Tribe
```

Listing 3.1: Classes of the BBC Wildlife Ontology

```
adaptation class className collection commonName conservationStatus
    distributionMap ecozone family familyName genus genusName growsIn habitat
    infraorder infraorderName kingdom kingdomName livesIn name order orderName
     phylum phylumName populationTrend redListStatus scientificName
    shortDescription species speciesName suborder suborderName subspecies
    subspeciesName superclass superclassName superfamily superfamilyName
    superorder superorderName taxonomicName threatDescription tribe tribeName
    yearAssessed
```

Listing 3.2: Properties of the BBC Wildlife Ontology

[2]`http://www.bbc.co.uk/ontologies/wildlife/2010-11-04.shtml` is taken
[3]There is a problem with the LaTeX template for Listings, please excuse.

3.3 Using Linked Open Data

As mentioned before, linking to and integrating data from existing sources is desired. As a younger research area of the Semantic Web, the Linking Open Data (LOD) movement (cp. Bizer et al. [2009]) is getting attention from both a technical and political point of view. The movement has a famous advocate in Tim Berners-Lee who spread word in a pair of TED talks (TED Conferences, LLC [2011a,b]) and promoted W3C activities.

> “The Open Data Movement aims at making data freely available to everyone. There are already various interesting open data sets available on the Web. Examples include Wikipedia, Wikibooks, Geonames, MusicBrainz, WordNet, the DBLP bibliography and many more which are published under Creative Commons or Talis licenses. The goal of the W3C SWEO Linking Open Data community project is to extend the Web with a data commons by publishing various open data sets as RDF on the Web and by setting RDF links between data items from different data sources. RDF links enables the navigation from a data item within one data source to related data items within other sources using a Semantic Web browser. RDF links can also be followed by the crawlers of Semantic Web search engines, which may provide sophisticated search and query capabilities over crawled data. As query results are structured data and not just links to HTML pages, they can be used within other applications.” (W3C [2011a])

Corresponding to Kobilarov et al. [2009a], the Linking Open Data project aims at developing best practices to opening up the *data gardens* (see Figure 3.1)) on the Web, interlinking open data sets on the Web and enabling web developers to make use of that rich source of information. Publicly available datasets — most in RDF format — are known as the *Linked Data Cloud*. The well-known and ever-growing LOD cloud diagram (Cyganiak, Richard and Jentzsch, Anja [2011], see Figure 3.3) is a visualization of data sources contributing to the Open Linked Data graph. DBpedia (Thibodeau Jr., Ted [2011]) and FOAF (Friend of a Friend, Brickley, Dan and Miller, Libby [2011]) are very well-known examples.

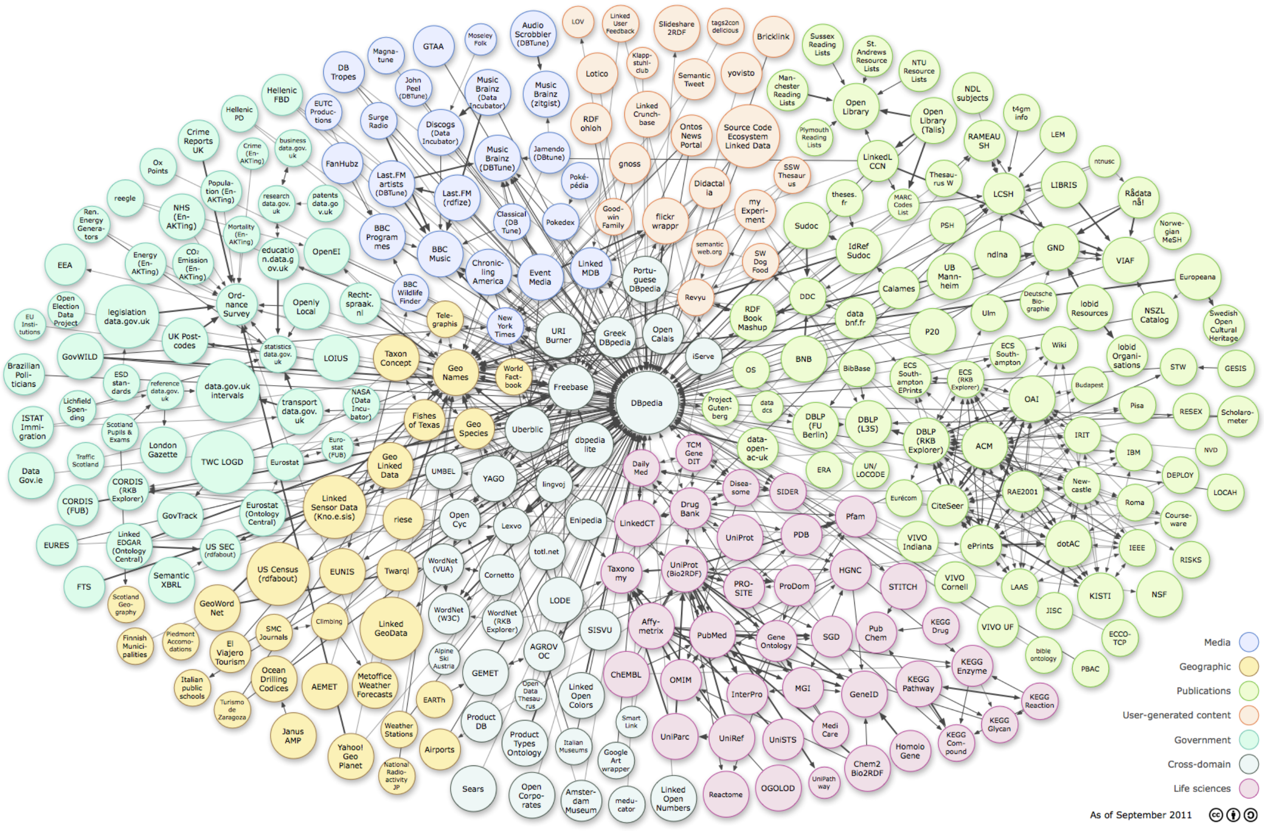


Figure 3.3: Linking Open Data cloud diagram, by Cyganiak, Richard and Jentzsch, Anja [2011], see `http://lod-cloud.net/`.

Chapter 4

Evaluating the Domain Model

This chapter focuses on testing the domain model with users. Therefore methods which are typically automated are out of scope. The provided selection of topics for evaluation is not a comprehensive list but should give an overview. A method called Focus Groups will be the topic for the first section. It is used to evaluate the domain model developed by the domain experts. Once the domain model is developed and the layout and the structuring of content are in progress, usability and accessibility tests are important to get feedback from users. One possible way to do a usability test is the Greeking Test, where the content on a webpage is made unrecognizable and tested if it is identifiable based on its position on the website. The third section will give an overview about the card-sorting technique, which can be helpful for deciding about the structure of content, especially to build hierarchies (only parts of the graph model) of the topics. The last section of this chapter will be about Thinking Aloud (TA) tests, which help to learn how the users think and deal with tasks. For tests the following things should be prepared in advance (cp. Molich [2007]):

- Ensure a comfortable test environment. It should be quiet. The best way is to use a usability lab, which is built for usability tests and therefore has a proper equipment in useful adjustment. Prepare a declaration of consent.
- Everything which is needed for the test has to be ready, for example: cards with content-terms for Card Sorting, the webpage with the content made unrecognizable printed out on paper for the Greeking Test or recording equipment for the Thinking Aloud test.
- Give test participants the chance to sketch something or write their ideas down. They should feel comfortable and free.
- If the test includes a Thinking Aloud test, recording audio and video premises proper equipment.

4.1 Focus Groups

Focus Groups are group discussions organised to explore a specific set of issues such as people's views and experiences of important topics (cp. Barker G., and Rich S. [1992]). Focus Groups were originally used within communication studies to explore the effects of films and television programmes, for instance of wartime propaganda (cp. Merton R., and Fisk M., and Kendall, P. [1956]). In the case of Domain Driven Design it is used to evaluate the domain model which is developed by the domain experts. The users will have to use the model somehow, therefore speak to users and try to get them to sketch their understanding of the domain and sketch back at them (cp. Smethurst, Michael [2009]). The most compelling quality of Focus Groups is their delivery of live-consumers for observation by developers or marketing managers (cp. Steward et al. [2007]). Ask participants to sit down on something like a round table for an open discussion. It is optional to mediate the discussion with a mediator.

4.2 Greeking Test

When the layout and the content structuring of a website is in progress, then usability tests are important to develop in a user-centric way, therefore the so called Greeking Test can demonstrate in an easy and cheap way how familiar users feel with the content structure. It is a basic rule to not make the user think (cp. Krug [2005]).

The Greeking Test is used to test if the user can identify the different parts of a webpage and was first proposed by Thomas Tullis in 1998 (cp. Tullis, Thomas S. [1998]). In the year 2011 it is common to use search engines, and therefore a good share of the users will be linked directly to the page without doing any navigation steps on the website. A clearly and easyly understandable navigation and usage of the website is important. If it is not, the user will leave the website without reading any other page. During the test the layout will be examined if it meets the user-needs.

The test can be done in the following way: Design the layout of the webpage in several different ways with a graphic tool. Simply change the position of the parts and print every possibility after the so called *greeking*. Replace the text with meaningless gibberish (greek) in a way which makes the content unrecognizable. The participant should not get distracted of reading text. Show the different possibilities to a test-participant and let her/him identify the standard page elements. The more parts are identified right the higher is the score of this so called mock-up. Let the participant rank the possibilities too, to find out which one is liked most. The result is an idea which mock-up structure will fulfill user needs best, or if none of the pages have good scores a combination of good parts can be the solution. Therefore the result combination should be tested against the other possibilities to seeif it performs better than them. Again the whole situation can be recorded for further analysis (cp. Andrews, Keith [2011] and Koch, Karin and Hye, Florian [2011]).

4.3 Card Sorting

Card Sorting is a method used to find a user-accepted structure for the navigation of webpages and navigation hierarchies. The best time to use Card Sorting is in the early design period, as it is the time where it is cheapest to rethink. The later the system has to be redesigned, the more effort and money it costs. The result of the test is a structure which can be used. The probability that people will agree with and like the resulting structure depends on the number of participants at the test, which basically means the more participants, the better. One procedure of card sorting will be done by one subject of the test. There should be more than one procedure. If more then one procedure is done, calculating an Inter-Rater-Agreement of the individual results, a value which can be interpreted from full agreement to lower than chance agreement, helps to filter out participants which mostly do not agree with all the others. An example for an Inter-Rater-Agreement is the value of Cohen's Kappa, which puts the relative observed agreement and chance agreement into relation. The result of Cohen's Kappa can be interpreted and used for identifying unwanted results of participants as *spam* (cp. Koerner [2011]). The subject of the test is an order to sort several terms which stand for the content of the website. The person should notice relations. It is possible to reject and to produce expressions. Each group should get a name which describes the members. Finally, the groups should be brought in a meaningful structure (cp. Koch, Karin and Hye, Florian [2011]).

Once the test is started it is important to notice each reaction of the test-participant as this is the way to gather information about the strength of members. The participant reads the provided cards and begins to sort and categorize them. The new order represents the structure of the content which the user expects or is easy to handle when he navigates though the webpages. If the result of the test-participant contains too much or not enough categories, simply tell the participant to build a hierarchy with several levels of the categories. The gained knowledge of the order is a starting point for the navigation design phase (cp. Koch, Karin and Hye, Florian [2011]).

Note that the card sorting test in principle outputs hierarchies, tree structures. That means that this test can not be directly used to create the whole graph-like model, however, for developing central taxonomies which are then part of the domain graph, it is useful.

4.4 Thinking Aloud Test

The first version of a website is usually full of mistakes. The same is true of the user interface: regardless of how careful it is created, there will always be some serious and unexpected problems. They might not occur when the developers itself use it, but when average users do. The Thinking Aloud technique therefore focuses on how users actually behave in concrete and realistic situations. The scenario is quite simple as the aim is to just record the test participant in audio and video and ask him to explain loudly what is thought and done during doing some tasks. Sometimes more than one video camera for recording is needed to film the test case and the mimic of the participant at the same time. If just one video camera is available then a mirror should be used to film both. A pre-test is important to check if the developed theoretical procedure is possible to solve and is possible to record. The TA test sometimes changes the test scenario, for instance a check which one of different layouts performs best with regard to time will not be comparable due to the talking slows down the participant's performance (cp. Molich [2007]). The most interesting thing is to gather information where exactly users are struggling and how they solve it.

Chapter 5

Case Studies

This chapter gives some examples, in which Domain Driven Design has been applied successfully. The first example will be a small one, just illustrating the main aspect of DDD. The second example will deal with DDD, as well as demonstrate how to use Semantic Web technologies to create comfortable user journeys through the Web.

5.1 A County Council's Website

This example from Ide-Smith [2011] deals with a website about Councillors and Council meetings. As discussed in the previous sections, a domain model should express the relationships between the key *things* of the domain. To identify the *things* and their relationships it is crucial to talk to a domain expert. For this Council's website the experts would be the officers in Democratic Services. Talking to users is necessary as well. Users often use simpler language to describe *things* the way they understand it, although some might struggle to actually give names to the *things* they describe. Again, for this example the users would be average persons. What comes out of this is a domain model, which expresses all relationships between *things*. However, to simplify this example, the following model does not label the relationships.

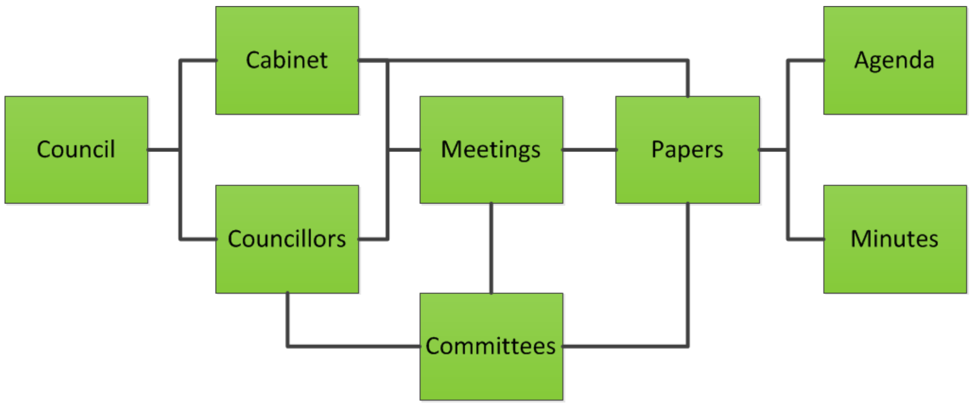


Figure 5.1: Councillors and Council domain model, redrawn from Ide-Smith [2011].

5.1.1 Search Engine Optimization

Most of the users nowadays enter a website via Google and not by typing in URLs. This means, users start visiting a site not from the homepage, but from a page deep within the site. A link to the homepage should therefore always be present on all subsites. Furthermore, one page should deal with only one concept, so search engines can point to it directly. In this example, that results in a page for a Councillor, a page for a Committee, a page for a Meeting, and so on.

5.1.2 URI Design

URIs are part of the user interface, so it is important to design them well. A good URI should be hackable, memorable and persistent. *Hackable* means that if parts of the URI are removed, the URI should point to a page further up in hierarchy. For example `http://www.mycouncil.gov.uk/meetings/agendas` and `http://www.mycouncil.gov.uk/meetings` should both work.

5.2 British Broadcasting Corporation

The British Broadcasting Corporation (BBC) is the largest[1] broadcasting organization in the world, operating numerous radio and television programs as well as an extensive website. The website consists of several domain-specific micro-sites, such as BBC News Online, Sport, Weather, TV or Radio. Micro-sites use the same data/domain model as the *parent* site, but with different branding and additional materials. Every BBC program is given its own page as part of a micro-site. Larger programs usually get their own micro-site.

The following example from Atherton [2011] deals with “The complex world of television shows” on BBC.

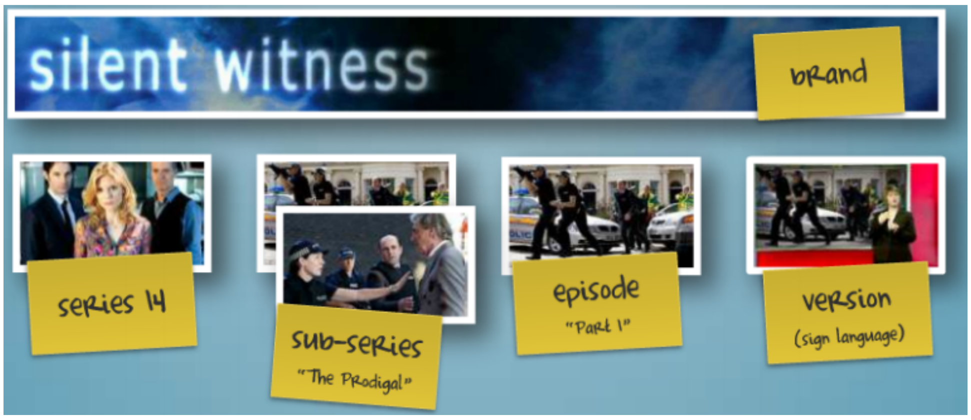


Figure 5.2: TV shows as seen by BBC programmes, from Atherton [2011].

Ever since, BBC used to design from the interface down instead from the domain model up. For many micro-sites however, this had to change. As one of the first sites, BBC programmes was redesigned, using a Domain Driven Design approach. Section 5.3 shows the corresponding domain model.

[1]And arguably best.

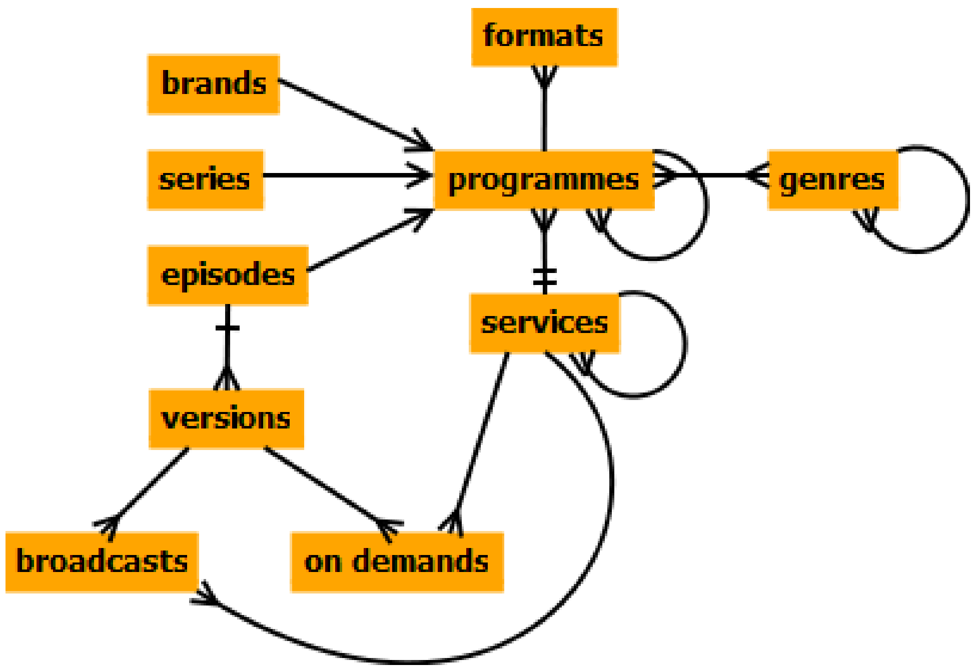


Figure 5.3: BBC programmes domain model, redrawn, original in Atherton [2011].

5.2.1 Problem of Cross-domain Navigation

Domain Driven Design alone, however, did not solve all the problems BBC had with its site. One other problem was, that the micro-sites, each providing large amounts of data, such as text, audio or video, were separate and standalone HTML sites which could indeed be accessed through own URLs, but rarely linked to each other. There wasn't a convenient way for users to get for example from a recipe on the food-specific site to an ingredient of that recipe in the nature section by simply clicking on a link. Since most of the micro-sites were maintained from different teams of programmers, interlinking was not easy to coordinate. Furthermore, data was not stored in a context-independent way, so it was hard for programmers to reuse it for a second purpose or visualize content in more than one way — cp. Raimond et al. [2010] and Kobilarov et al. [2009b].

Storytelling through Web content nowadays does not work like it used to for years. Today, users do not want to be guided straightforward from one article to the next. They want to decide for themselves where to go. A user journey therefore can go for instance from Google to a BBC site to Wikipedia, then back to a BBC site. Figure 5.4 visualizes such an example. However, precisely targeted links can only be provided if those links exist at data level according to Scott and Smethurst [2008].

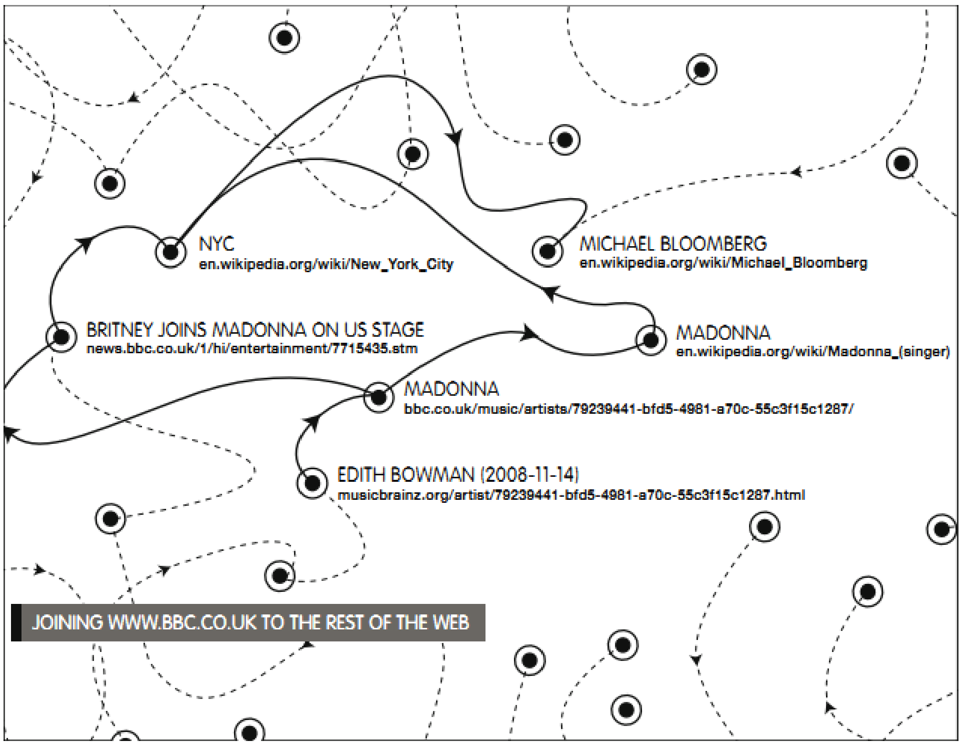
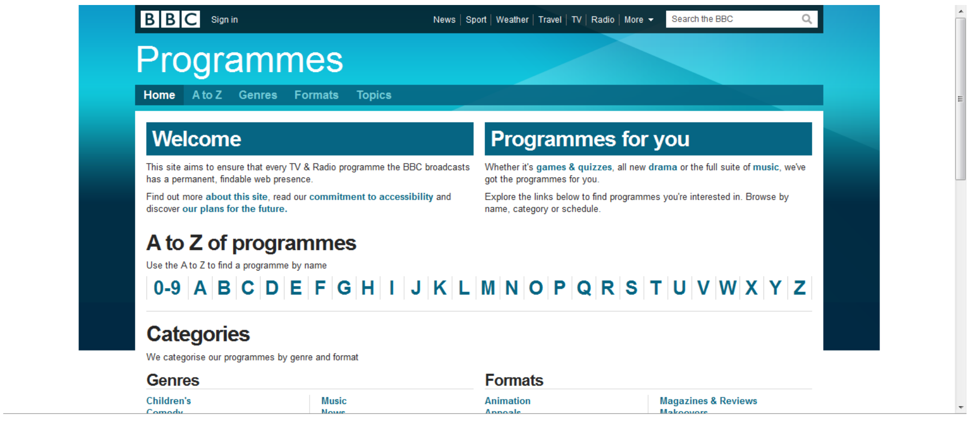

NYC
en.wikipedia.org/wiki/New_York_City
MICHAEL BLOOMBERG
en.wikipedia.org/wiki/Michael_Bloomberg
BRITNEY JOINS MADONNA ON US STAGE
news.bbc.co.uk/1/hi/entertainment/7715435.stm
MADONNA
en.wikipedia.org/wiki/Madonna_(singer)
MADONNA
bbc.co.uk/music/artists/79239441-bfd5-4981-a70c-55c3f15c1287/
EDITH BOWMAN (2008-11-14)
musicbrainz.org/artist/79239441-bfd5-4981-a70c-55c3f15c1287.html
JOINING WWW.BBC.CO.UK TO THE REST OF THE WEB

Figure 5.4: How a user's journey might look, from Scott and Smethurst [2008].

5.2.2 Solution Using Semantic Web Technologies

As solution to the problem described before two projects, *BBC Programmes* and *BBC Music*, were launched. They provide unique identifiers for programmes and artists featured in BBC music programmes which allows referring from other pages. Moreover, they make use of and link to related content in the Linked Data cloud.


BBC Sign in News | Sport | Weather | Travel | TV | Radio | More Search the BBC

Programmes

Home A to Z Genres Formats Topics

Welcome

This site aims to ensure that every TV & Radio programme the BBC broadcasts has a permanent, findable web presence.

Find out more about this site, read our commitment to accessibility and discover our plans for the future.

Programmes for you

Whether it's games & quizzes, all new drama or the full suite of music, we've got the programmes for you.

Explore the links below to find programmes you're interested in. Browse by name, category or schedule.

A to Z of programmes

Use the A to Z to find a programme by name

0-9 | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Categories

We categorise our programmes by genre and format

Genres

Children's Music

Formats

Animation Magazines & Reviews

Figure 5.5: Screenshot from BBC Programmes, `http://www.bbc.co.uk/programmes`

To allow other sites to make use of BBC content as well as the programmes domain model was converted into a RDF ontology and RDF feeds were added. Since the domain model was well thought through, conversion then was not much work to do. It took one BBC programmer a week according to Scott and Smethurst [2008]. In this article it also says

"Our hope in making BBC data available as RDF is that we will make it as generative as possible — helping others to do interesting things with our data."

5.2.3 The Web as Content Management System

Today, many sites provide information (content) for the Internet. *MusicBrainz*[2], for instance, provides web identifiers for artists or music tracks which BBC can use for its music section. However, concepts can be related to more than one domain. For example, Kate Winslet is both an artist (belongs to the music domain) and an actress (belongs to the movies domain). BBC developed a mechanism for interlinking concepts and content about concepts using *DBpedia*[3]. *DBpedia* provides Linked Data URIs for a broad range of concepts, as well as structured data about those concepts and their relationships.

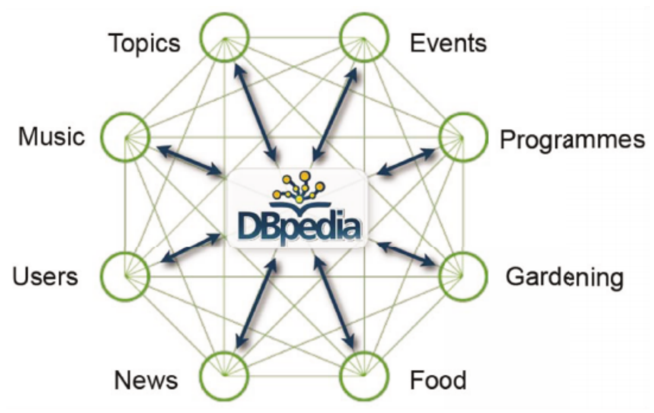


Figure 5.6: Linking BBC domains, by Kobilarov et al. [2009a].

5.2.4 URI Design

The following is based on Atherton [2011]. For BBC it is important to have unique and persistent URIs. Going back to our example with the BBC programmes domain model, suppose there is a series called *Sherlock* looking like that:

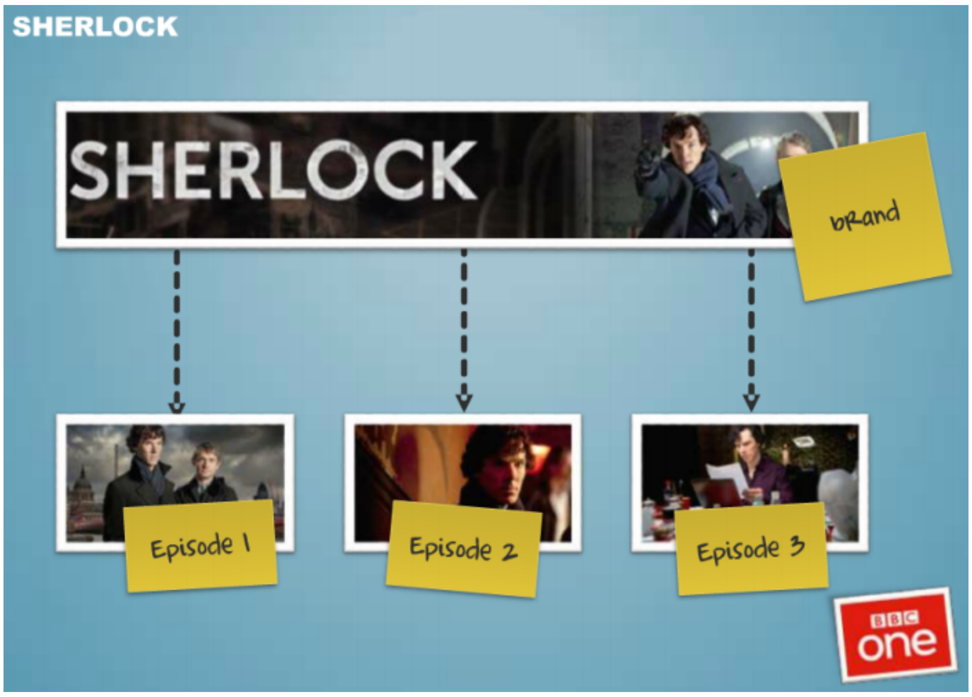


Figure 5.7: Concept of television series *Sherlock*, by Atherton [2011].

[2]http://musicbrainz.org/
[3]http://dbpedia.org/

In the future it might look like that:

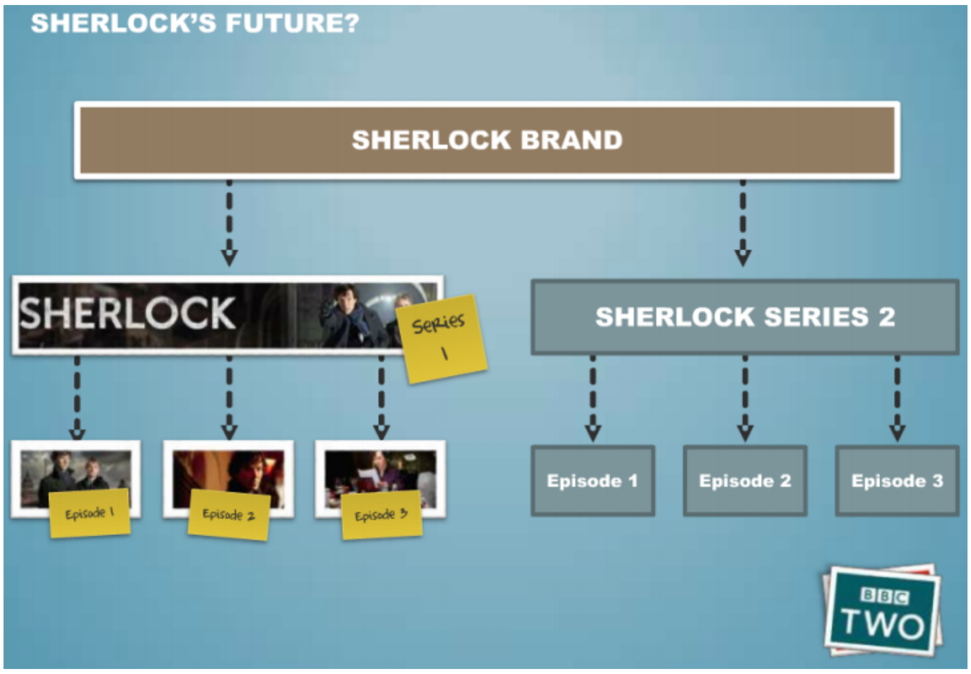


Figure 5.8: Possible future of television series *Sherlock*, by Atherton [2011].

Nevertheless, how can anyone know? So how should the URI be designed?

1. `http://www.bbc.co.uk/bbcone/sherlock/series1/episode1`**`.shtml`**
   No. Do not expose the technology. No one cares and it might change anyway.
2. `http://www.bbc.co.uk/bbcone/sherlock/`**`series1`**`/episode1`
   No. Is there really a series 1 until there is a series 2? Will it be the same overseas?
3. `http://www.bbc.co.uk/`**`bbcone`**`/sherlock/episode1`
   No. What if this show moves to BBC TWO or another network later?
4. `http://www.bbc.co.uk/`**`sherlock`**`/episode1`
   No. What if Radio 4 decide to do a program called 'Sherlock' in the future?
5. `http://www.bbc.co.uk/episode1`
   No. Hmm, well now that does not make much sense at all!
6. `http://www.bbc.co.uk/programmes/b00t8wp0`
   Yes. Now the unique identifier BBC gave to that program is used. That may not be human-readable, but it is an acceptable compromise to get persistence. HTTP 301 redirects can be used for a URL which is more easy to remember.

The concrete decision will depend on a number of factors, and it is up to the developers and experts to estimate the chances of certain things being extended or changed.

Chapter 6

Discussion and Concluding Remarks

Domain Driven Design is a philosophy which allows the developer and domain expert to develop highly informative models of a subject area which will be reflected in the entire project. A very good knowledge about the domain is required. Although this new approach is especially for larger projects, even the smallest projects can benefit from this way of thinking. A world of all connected things is the main idea, Eric Evan‘s Domain Driven Design book has characterised it. The first step is to design a ubiquitous language, due to talking to experts is essential for modeling and domain related concepts can be discussed. They will help to understand the complex things of the domain. Another point is to talk to the user, what they want is important. For the developer it is essential to capture the mental model. To break down complex domains into manageable parts a bounded context should be used.

Summary of the new information architect, according to Atherton [2011]:

- Thinks about real-world things
- Thinks ’user-experience’ from the ground up
- Designs for mobile first
- Wrestles data from around the Web, and publishes their own data back out

Domain Driven Design is a challenge, but a good way to start developing complex systems and discover a new development and design. The BBC and the Council‘s website are just the beginning. To conclude the report, a set of general principles based on Rissen et al. [2011] is recalled:

- Investigate and understand the domain of the product
- A common language between users, developers, designers, product managers
- A domain model shows the most important things, and how they’re connected
- Everything should flow from the model — datastore, designs, URL schemas
- One URL per thing — must be permanent, nice if it’s readable & hackable
- Search engines are the least able user — design for them & accessibility will result
- Design for the Web first, then make appropriate representations for each platform
- Test iteratively over and over again with users
- Don’t hesitate to throw away
- Linking instead of re-inventing

References

Abel, Avram and Marinescu Floyd [2006]. *Domain-Driven Design Quickly*. C4Media Inc., 1–96 pages. `http://portal.acm.org/citation.cfm?id=1557346`.

Andrews, Keith [2011]. *Information Architecture and Web Usability*. `http://courses.iicm.tugraz.at/iaweb/iaweb.pdf`.

Atherton, Mike [2011]. *Beyond the polar bear*. `http://www.slideshare.net/reduxd/beyond-the-polar-bear`.

Barker G., and Rich S. [1992]. *Influences on adolescent sexuahty in Nigeda and Kenya: findings from recent focus-group discussions. Studies in Family Planning*, 23, pages 199–210.

Bizer, Christian, Heath Tom, and Berners-Lee Tim [2009]. *Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems*, 5(3), pages 1–22. `http://www.citeulike.org/user/omunoz/article/5008761`.

Bonnie, Lida, Spring Hull, and Katie Pilcher [2003]. *Breadcrumb Navigation : An Exploratory Study of Usage. Search*, 5(1).

Brickley, Dan and Miller, Libby [2011]. *The Friend of a Friend (FOAF) project*. `http://www.foaf-project.org/`.

Cyganiak, Richard and Jentzsch, Anja [2011]. *Linking Open Data cloud diagram*. `http://lod-cloud.net/`.

Evans, Eric [2004]. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.

Hruby, Pavel [2005]. *Ontology-Based Domain-Driven Design. Development*, 37, pages 1–9. `http://www.softmetaware.com/oopsla2005/hruby.pdf`.

Ide-Smith, Michele [2011]. *Using domain models in IA*. `http://www.ide-smith.co.uk/?p=866`.

Kaiser, Rene, Claudia Wagner, Martin Hoeffernig, and Harald Mayer [2011]. *The interaction ontology model: supporting the virtual director orchestrating real-time group interaction*, pages 263–273. `http://portal.acm.org/citation.cfm?id=1950054.1950086`.

Khan, Aslam [2009]. *Getting Started with Domain-Driven Design. I Can*, pages 1–6.

Kobilarov, Georgi, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee [2009a]. *Media Meets Semantic Web - How the BBC Uses DBpedia and Linked Data to Make Connections*.

Kobilarov, Georgi, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee [2009b]. *Media Meets Semantic Web - How the BBC Uses DBpedia and Linked Data to Make Connections. The Semantic Web Research and Applications*, 5554(3), pages 723–737. `http://www.springerlink.com/index/P53T728027254485.pdf`.

Koch, Karin and Hye, Florian [2011]. *E-Business 2 - Usability.* `http://dmt.fh-joanneum.at/projects/ebus2/objects/application_pdf/CHI-Paper_v1.pdf`.

Koerner, Christian [2011]. *Evaluation Strategies and Methods.* `http://kmi.tugraz.at/staff/markus/courses/WS2011-12/707.009_knowledge-management/slides/week-evaluation.pdf`.

Krug, Steve [2005]. *Do not Make Me Think! A Common Sense Approach to Web Usability.* Sage Publications. ISBN 0321344758.

Laribee, David [2009]. *An Introduction to Domain Driven Design. MSDN Magazine*, (February). `http://msdn.microsoft.com/en-us/magazine/dd419654.aspx`.

Merton R., and Fisk M., and Kendall, P. [1956]. *The focused interview: a report of the bureau of applied social research.*

Molich, Rolf [2007]. *Usable Web Design.* Nyt Teknisk Forlag. ISBN 978-87-571-2526-9.

Nielsen, Jakob [2007]. *Breadcrumb Navigation Increasingly Useful.* `http://www.useit.com/alertbox/breadcrumbs.html`.

Pammer, Viktoria [2010]. *Automatic Support for Ontology Evaluation - Review of Entailed Statements and Assertional Effects for OWL Ontologies.* Ph.D. thesis. `http://ebookbrowse.com/dissertation-viktoria-pammer-pdf-d54636999`.

Raimond, Yves, Tom Scott, Patrick Sinclair, Libby Miller, Stephen Betts, and Frances McNamara [2010]. *Case Study: Use of Semantic Web Technologies on the BBC Web Sites.* `http://www.w3.org/2001/sw/sweo/public/UseCases/BBC/`.

Rissen, Paul, James Howard, and Silver Oliver [2011]. *Domain Driven Design and the Semantic Web at the BBC.* `http://www.slideshare.net/r4isstatic/an-introduction-to-domain-driven-design-for-product-managers`.

Scott, Tom and Michael Smethurst [2008]. *Building coherence at bbc.co.uk. Nodalities The magazine of the semantic web*, (December). `http://www.talis.com/nodalities/pdf/nodalities_issue5.pdf`.

Smethurst, Michael [2009a]. *Designing for your least able user.* `http://www.slideshare.net/fantasticlife/designing-for-your-least-able-user`.

Smethurst, Michael [2009b]. *Designing for your least able user.* `http://www.bbc.co.uk/blogs/radiolabs/2009/03/designing_for_your_least_able.shtml`.

Smethurst, Michael [2009c]. *How we make websites.* `http://www.slideshare.net/fantasticlife/how-we-make-websites-presentation`.

Smethurst, Michael [2009]. *How We Make Websites.* `http://www.bbc.co.uk/blogs/radiolabs/2009/01/how_we_make_websites.shtml`.

Smethurst, Michael [2009]. *Designing for Your Least Able User.* `http://www.bbc.co.uk/blogs/radiolabs/2009/03/designing_for_your_least_able.shtml`.

Steward, David W., Prem N. Shamdasani, and Dennis W. Rook [2007]. *Focus groups: Theory and practice.* Sage Publications. ISBN 0-7619-2583-X.

TED Conferences, LLC [2011a]. *Tim Berners-Lee on the next Web.* `http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html`.

TED Conferences, LLC [2011b]. *Tim Berners-Lee: The year open data went worldwide*. `http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide.html`.

Thibodeau Jr., Ted [2011]. *DBpedia*. `http://dbpedia.org/`.

Tullis, Thomas S. [1998]. *A Method for Evaluating Web Page Design Concepts*. `http://www.cc.gatech.edu/projects/PageSleuth/references/p323-tullis.pdf`.

W3C [2011a]. *Linking Open Data*. `http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`.

W3C [2011b]. *Resource Description Framework*. `http://www.w3.org/RDF/`.

W3C [2011c]. *SPARQL Query Language for RDF*. `http://www.w3.org/TR/rdf-sparql-query/`.