Application of UriGraph to Uniform Resource Identifier Design

Hrvoje Šimić
University of Zagreb
Faculty of Electrical Engineering and Computing
Department of Telecommunications
Unska 3, HR-10000 Zagreb, Croatia
mailto:hrvoje.simic@fer.hr

Abstract

UriGraph is a Web site structure model that specifies identity, identifier, position and composition of each resource constituting the Web site. This paper presents the application of resources identifier model (a subset of UriGraph) to Uniform Resource Identifier (URI) design. The URI technology is essential to the Web, and exposure of URIs to the user makes it a major – although often overlooked – part of the Web user interface. The process of assigning specific URIs to resources on the Web site and thinking up rules for mapping URI subspaces to classes of resources has become known as the URI design. The paper lists requirements and gives recommendations for URI design in general, and for multilingual Web sites in particular. A case study illustrates the proposed approach.

Introduction

Uniform Resource Identifier (URI) design is lesser known discipline of the Web information architecture, although it is a very important aspect of Web usability and functionality. The goal of this paper is to contribute to the field, theoretically (attempting to formulate definitions and identify the use cases and requirements) as well as practically (giving recommendations and proposing a new model for designing URIs).

The paper describes the design only of some URIs, primarily those in "http" scheme, which are the most important for the Web. Similar principles can be extended to other of URI schemas, but URIs are so generally defined that it is hard to establish many universal principles.

The paper is organized as follows. The next section defines the cornerstone terms of the paper, especially "resource" and "URI", as well as the need for designing URI. Section titled "URI design" lists the use cases for URIs, followed by a list of requirements and recommendations for URI design. The following section describes the UriGraph model in brief, with more detailed look at the resources identifier model. A case study illustrating the use of UriGraph in URI design of author's own Web site follows. The paper ends with the conclusion and a list of references.

Web resources and identifiers

Resources

The current standard on Uniform Resource Identifiers [rfc2396] defines the term very widely: "A resource can be anything that has identity." Another important standard that has the word "resource" in its name, the Resource Description Framework [rdfms] says: "All things being described by RDF expressions are called resources." A little later, it specifies: a resource can be "any entity imaginable." Or, as Tim Berners-Lee says: "A 'resource' is a conceptual entity (a little like a Platonic ideal)." [tbl96c]

On the other hand, there has been a tendency to perceive the resource to be Web-specific. HTTP/1.1 specification [rfc2616] sees it as only "a network data object or service". This view reflects the older, traditional Web. Extending the term "resource" to cover almost everything is typical for the Semantic Web.

General definition of a resource

A better definition of (general) resource says: "resource is a source of aid or support that may be drawn upon when needed" [wordnet] [cha01]. Four essential qualities of every resource can be recognized from this definition.

First, a re*source* is a **source**. Generally speaking, it is the sea, not fish; the power plant, not energy; the book, not a sentence. On the other hand, almost any entity can be regarded as a source of something from a proper perspective.

Second, it is a source of something **useful**. That "something" can be matter, energy, information or whatever, but it has to have some value to the user, it should aid or support him. It should be noted that usefulness is inherently subjective: one civilization's pool of stinky mud is another's energy resource, and the starry sky is a navigational resource only to some.

Third, it is **available**. The user can access and use it at will. The resource is expected to be at disposal to the user, although the user may be other than the owner of the resource. However, resource's availability depends on its proper use: your checking account may not be available to you in any place on Earth at any time, and you cannot expect for the public transportation bus to just appear before you when you wish for it.

Four, it is **persistent**. Etymologically, "re-source" can be interpreted as "constant source", a source that never goes dry. But this "never" doesn't necessarily mean "for all eternity", because every real-world resource has its lifespan. However, a resource should be persistent; it should be around for some expected duration, so people could count on it during that time. Sometimes they can be expected to last for many years, other times they are needed only for a few hours (e.g. an improvised stage).

Although these four requirements can seem rather restricting, they all are flexible to subjective review. Almost anything can be presented in the way that it seems to be a source of useful stuff that's available and persistent enough. That flexibility is good, as long as the perspective of considering something a resource is an honest one.

Web resource

A Web resource is a real resource. It is a source of data, not the data itself. The idea is that the data is useful to the user of the resource, or the user wouldn't bother to use it. A Web resource is available through the underlying computer network (explicitly, the HTTP over TCP/IP or something equivalent). And finally, the Web resource should be persistent: it should be up and running for an appropriate period of time.

More specifically, Web resource is an information resource. Dereferencing a Web resource typically returns digital data. Web resource can also receive some data which may modify its state. In Hypertext Transfer Protocol these functions are implemented through the GET and POST methods, respectively.

It is useful to regard that a Web resource is representing some concept in the context of the Web site it resides. Since every Web site is owned by some authority, it could be said that it represents that authority's view on the concept.

For instance, "http://w3.org" identifies the resource "W3C's home page". That resource represents the concept of the World Wide Web Consortium and it's owned by the Consortium itself. The task of providing the information about the entire W3C is too much for one resource, so a large collection of resources has been included in the W3C's Web site. So the W3C's home page currently provides information in three main sections: (1) declaring its identity, (2) news listing, and (3) references to other resources providing detailed information on specific W3C's activities and issues. The last section is indirectly providing information on complex issues by referencing to other resources which usually deal with simpler issues – the basic principle that ties the whole Web together.

Identity of the resource (in the context of its Web site) corresponds to the intension of the representing concept. Identity reputation of the resource corresponds to the public perception of its identity.

Amongst the Web resources some are not directly exposed to the user, but are used (referenced) by other resources. The examples include decorative images or separate CSS definitions.

Uniform Resource Identifier

Uniform Resource Identifier (URI) is a standard [rfc2396] for resource identifiers on the Internet. URI is a sequence of characters, easily recognized by humans familiarized with Latin script.

A much more popular term similar to URI is URL, Uniform Resource Locator. Strictly speaking, URLs are subset of URIs, but for the purposes of this paper we don't need to make this distinction, and will always use the term "URI" [urcr].

URI syntax

General URI syntax is simple [rfc2396]. Every URI is composed of two parts separated by a column. The first part is the scheme name, and the syntax of the second part is defined specifically for that scheme. This paper will focus specifically on one URI scheme named "http", because it is the most relevant for the Web. That scheme defines http-URI as two slashes, followed by authority (host name with optional port, separated by a column), followed by an optional path, and followed by an optional query, beginning after a question mark. Path starts with a slash and consists of zero or more segments, each separated by a slash. Query consists of segments as well, separated by an ampersand. Each query segment contains a key and an optional value, separated with the equals sign.

Here is an http-URI which includes all optional parts:

```
http://example.org:81/a/bc?x=1&yz
```

Scheme is "`http`", host name is "`example.org`" and port is "`81`". Path consists of two segments named "`a`" and "`bc`". Query also consists of two segments, first with key "`x`" and value "`1`", second with just the key "`yz`".

The query issue

The query string controversy starts with the very document it is defined in [rfc2396]. It says: "The query component is a string of information to be interpreted by the resource." If the resource is identified before the query component is interpreted, why is the query a part of the identifier? This paper discards this definition as paradoxical and archaic, recognizing both parts of URI as identifying parts, although in different roles.

There are two main technical uses of queries: HTML forms and Web executables. When submitting data from a HTML form using a GET method, Web browsers construct URIs by adding query segments with the data in the form. Also, Web servers usually identify the executable file using only the path part, passing the query segments to the executable as parameters.

Although there are no formal distinctions between path and query segments, because of the practical uses query segments are often perceived to be:

- containing more detailed information about resource identity then path segment;
- optional, such that removing it from URI would result in "defaulting" some parameters;
- generally in random order, in contrast to path segments where the order is important.

URI is a part of the UI

The Web has its user interface, predominantly covered with user agents (usually browser). But one aspect of the interface common to all user agents is user exposure to URIs [nie99].

The need to consider URIs a part of the user interface on the Web comes from the basic designer's goal of **transcribability**. URI standard [rfc2396] specifies: "A URI may be transcribed from a non-network source [and] often needs to be remembered by people." URIs are one of the user's ways of dealing with the Web.

The extent to which URI should be exposed to the user and the need for making them more user-friendly is a subject of a debate. A closely related issue is the URI **opacity**: using the URI as a black box, without ever seeing (analyzing, caring about) the sequence of characters it is composed of. While it is useful for URIs to be opaque in some situations, others require that URI be analysed or constructed [bone02c].

URI Design

The process of assigning specific URIs to resources on the Web site and thinking up rules for mapping URI subspaces to classes of resources has become known as the URI design [nie00] [bone02b]. This section lists use cases, proposes requirements, and gives recommendations for URI design.

Use cases for URIs

When it comes to using URIs, there are basically two user roles: browsing and authoring. Browsing users use the URI to communicate with the resource it identifies. Authoring users include URIs into hypermedia documents, program code, configuration data etc.

There are three types of media where URIs may appear:

- hyperlink-enabled media – media supporting hyperlinks, such as hypermedia in browsers;
- character-manipulative media – media supporting copying and pasting URIs as a sequence of digitally encoded characters, but not supporting hyperlinks;
- other media.

Hyperlink-enabled media

This is the natural environment for URIs, and they are widely used in this type of media, usually hidden from the user. However, they may be exposed through the "Address" field on the browser or by other means (status bar, copying links, bookmarks, etc.).

Character-manipulative media

URI is passed through digital text-enabled, but not hyperlink-enabled media, available for human reading, but also available for copying into other programs or media. Examples include e-mail messages in plain text, PS files etc. Sometimes the media itself allows hyperlinks, but that feature is not used and the URI cannot be activated. This type of media exposes the URI as a textual fragment, but doesn't force the user to manually parse or transcribe the characters. Instead, the complete URI can be opaquely transferred, using features such as "copy & paste".

Reading and writing

URI is written on non-character-manipulative media for human reading. Common examples include newspapers, sides of the bus, napkins, TV screens, brochures, letters... URI information cannot be easily extracted without human processing of characters.

URI is read for many purposes, such as transcribing, memorizing, analysing the contents etc. Often, an URI is read aloud or spelled out over voice-communication media such as telephone, radio or TV.

Memorizing

URI is remembered, stored either in short or long-term memory, depending on how long and how often we need to remember them.

Guessing

People usually guess URIs of popular companies' Web sites. For instance, although I can't distinctly remember seeing the URI for The Coca-Cola Company, I guess it is `http://www.coca-cola.com` (it turns out I was right). And if there is a John Doe employed at my faculty, I guess its e-mail address URI would be `mailto:john.doe@fer.hr`. Both are good guesses, but both could be wrong. The described ways to form an URI are strong conventions, but far from a standard.

This method of guessing uses **replacement** of critical part of URI, hoping to produce the desired result. In another example, dictionary.com's definition of the word "reference" is at the URI http://www.dictionary.com/search?q=reference. It would be a good guess that its definition of the word "resource" is at http://www.dictionary.com/search?q=resource.

Another method of guessing is by "**trimming**" or "hacking" parts of the URI. If you know the URI of an Alertbox article is http://www.useit.com/alertbox/990321.html and you want to see more Alertbox articles, in the absence of an appropriate link you can resort to guessing the page with the list of all articles would be at http://www.useit.com/alertbox/ – "move to higher levels of the information architecture by hacking off the end of the URL" [nie99]. That method is implemented as a navigation tool inside the Google Toolbar [gtb].

Generally speaking, guessing is usually performed by observing existent URIs, noticing a pattern in their structure and applying the pattern to the new information resulting in a new URI.

Constructing

URI may be constructed via a predefined schema. Schema is somehow authorised by the site owner, assuring the user in the validity of constructed URIs. Note that dereferencing a valid URI may result in "404 Not Found" response status code [rfc2616] if the concept is not represented at the site. In practice, there's a fine line between guessing and constructing. Constructing can be regarded as taking an educated guess on the mapping between the URI and the resource it identifies.

URI design requirements

Based on cases of using URIs, there can be recognized seven important design requirements. These are: meaningfulness, persistence, good structure, shortness, readability, memorizability, and pronounceability.

Meaningfulness

Meaningfulness is the only design requirement recommended by [rfc2396], concerning logical connections with the referenced concept. Logic can be sought in many contexts, especially cultural, organizational, and technical. Meaningful URIs facilitate all uses, especially memorizing and guessing.

For example, take The Coca-Cola Company's Web site and URI http://www.coca-cola.com. The "http" URI scheme is technical requirement for a Web site homepage URI. "Coca-cola" is the name for the company in popular culture. A strong convention for such companies' Web sites is to have Internet domains in the form of "www.*popular-company-name*.com". Another strong Web convention is that homepages don't have any path or query parts. These four facts combined make a strong logical connection between the URI http://www.coca-cola.com (reference) and The Coca-Cola Company's Web site homepage (referenced concept).

Persistence

Persistence is one of the most mentioned requirements for URIs. The usual formulation is: "URI should consistently reference to the same thing" [tbl96b] [nie98]. In fact, there are three kinds of persistence regarding Web resources:

- resource persistence – keeping the resource operational;

- resource identifier persistence – keeping the URI identifying the same resource;
- resource identity persistence – making the resource constantly supplying information consistent to its identity (or identity reputation).

Strictly speaking, none of these three kinds of persistence is URI design requirement, but they are often associated with URIs, since the URI is exposed to user during persistence problems. The resource identifier persistence is the most common problem and, unlike the other two, it can be prevented by good design.

Good structure

Good URI structure means designing URIs that can be effectively trimmed and its parts can be replaced to form another meaningful URI, thus automatically facilitating guessing and constructing. It is not recommended for URIs to blindly follow site's navigation layout, because it can frequently change. However, if reorganizing the site implicates restructuring the URIs, requests on old URIs should be consistently redirected.

Shortness

Short URI has obvious advantages. It can be typed in more quickly and accurately. It is less likely to break to the following line in text, which is particularly important in e-mails, but also in printed media, such as newspapers with narrow columns. Short URI is generally easier to work with in systems involving smaller digital devices, such as cellular phones and PDAs.

Readability

Readability refers to the human ability to visually perceive the written URI correctly so it can be memorized, transcribed or pronounced.

Memorizability

URI memorizability refers to human ability to more efficiently remember the URI for future uses. Long-term memory as a storage area for URIs is popular because of its constant availability (even in non-digital environment) and speed. The short-term memory is used in transcription.

Pronounceability

Pronounceability is obviously required for any use of URI involving reading it out aloud.

Interdependence of requirements

These design requirements for good URIs are not entirely independent. URI shortness often supports requirements of memorizability, readability, and (sometimes) pronounceability. However, it can be in conflict with meaningfulness and good structure, since they sometimes ask for more verbose expressions. Memorizability and persistence largely depend on meaningfulness. Readability and pronounceability are often closely interdependent.

URI design recommendations

Based on the design requirements, the paper suggests some restrictions and guidelines for URI design.

Lexical limitations

Further limiting the set of characters used in URI design can be an easy way to improve their readability, memorizability and pronounceability.

The strongest recommendation is not to use reserved characters as escape sequences (for example, space is escaped as "`%20`"). URIs with escaped sequences are longer and even less readable, memorizable, and pronounceable.

Path part of http-URI is best restricted to the limited set of characters, consisting of:

- letters of the English alphabet in both cases (however, uppercase characters should be used very sparingly);
- digits from '`0`' to '`9`';
- slash ('`/`') – for separating path segments;
- hyphen ('`-`') – for separating words and other sub-segment parts;
- dot ('`.`') – for special formatting.

Some characters are not recommended:

- underscore ('`_`') is lexically uncommon in popular use, similar to the hyphen, and its use in the path may be inconsistent, because it is not allowed in the host name;
- comma ('`,`') is similar to the dot and unusual in URIs.

Query part must be lexically more flexible, because it can carry any string of characters (for example, from a Web form). However, query segments that are more likely to be exposed should follow the same guidelines on lexical limitations as the path segments.

Case sensitivity

Case sensitivity of the URI is a serious obstacle to its memorizability and pronounceability. Some parts of URI like schema name and host name are required to be case insensitive and it is recommended to apply the same insensitivity to the other parts. Again, this principle may be ignored in some parts of the query where case sensitivity is necessary.

Host name

Host names shouldn't be prefixed by "www". "`example.org`" is shorter and more pronounceable then "`www.example.org`". Also, the acronym "WWW" is being replaced with the simpler and more human-friendly "Web". The main argument for including the "www" prefix is that it is part of the outer message – in other words, that it suggests that the whole string is an URI, not something else [nie99]. But there are other clues specific to host names in URIs: lowercase characters separated by dots and familiar suffixes (top level domains, like "com"). That format is at least as recognizable as the telephone number format. Additional context can be provided by a simple clue, like:

```
Tel: +1 234  5678901
Web: example.com
```

The other issue is separation of words in host names. One popular practice is not to separate them (e.g. "`barnesandnoble.com`", "`rottentomatoes.com`"). However, separating the words with hyphens greatly improves readability, and somewhat pronounceability, only slightly lengthening the URI.

On the other side, adding "www" prefix and running words together is somewhat of a standard in Web publishing. Therefore I recommend providing aliases to host names for all combinations ("`example-one.org`", "`www.example-one.org`", "`exampleone.org`", "`www.exampleone.org`") if possible, but publishing only the recommended version (in the example, only the first host name).

Contents

URI should contain only information about resource identity, not more and not less. There are three types of error in designing the URI contents:

- irrelevant information;
- redundant information;
- lack of information.

To introduce unwarranted, irrelevant information in an identifier is the most common and most painful designer sin, usually leading to short-lived, lengthy and generally inferior URIs. The designer should not include:

- technical details, such as file-type suffixes (e.g. "`.html`", "`.asp`"), technology-specific path segments (e.g. "`cgi-bin`", "`plsql`") for they are subject to change when the technology changes;
- internal organization information, like the name of the department or the person maintaining the resource;
- any changeable property of the resource, like "new", "confidential", etc.

Tim Berners-Lee [tbl92] [tbl96a] postulates that every piece of information included in the URI is compromising its persistence, because the more information the URI contains, the more likely is that some of that information will change – thus making the URI brittle. However, if the words used in the identifier express only the attributes of the represented concept itself, that identifier is not burdened with irrelevant information.

Redundant information in an URI means that the part of resource identity is included more then once. For example, the path "`/1998/log/27.05.98`" contains redundant information about the year. Better solution would be "`/log/1998/5/27`" which is shorter and well structured.

Finally, the lack of necessary information in the identifier can also be a serious source of its shot life. For example, the imaginary university has hosted a WebConf conference in 2001 and for its Web site chose "`http://university.tld/WebConf`". Next year it became obvious that the better URI would have been "`http://university.tld/WebConf/2001`", because they had to place "WebConf 2002" somewhere.

Query design

Some experts caution against the use of query in URI design, receiving data from HTML forms using the GET method being an exception [bone02a]. Restricting the issue only to URI design, it is recommended that query parts in URIs be used sparingly. Characters used in query ("`?`",

"&" and "=") are less readable and pronounceable, so exposing such URIs should be done cautiously.

Multilingualism

The natural language in which URI is written is essential to its quality. If people understand the words that make up URI, it is easier for them to read, remember and pronounce it. It is only natural that an URI identifying a resource serving content in English be written in English as well.

When multiple resources residing on the same site vary only in language of the content served, it is appropriate that they URIs be only translations of the same idea, for example:

```
/dalmatia/hvar/accomodation?currency=EUR
/dalmacija/hvar/smjestaj?valuta=EUR
```

The content of second relative URI has the same meaning as the first, only in Croatian. As a result, the URI itself suggests the requested language, without its explicit mention. Although differences in language-specific URIs may suggest separate Web resources (differences in host name may even suggest separate Web sites) it is useful to regard them as a single resource providing English and Croatian translations of a single content. This way, the site's structure is coherent and the design and maintenance is easier.

Of course, there can be resources and URIs that are not related to one specific language – in those cases, the choice of natural language for the URI is left to author's best judgement.

UriGraph

The model mentioned in this paper's title can be used to visually describe the design of URIs in the Web site, its URI subspace organization. The model is named "UriGraph", because it uses URI analysis through a directed graph to construct information about the resource identity, position and composition. Special graphic representation and an XML grammar for describing site structure are devised. For a formal definition and a more detailed description of UriGraph, see the author's Master's thesis [sim02].

Web site structure model

UriGraph is the model of Web site structure. The Web site structure is defined as the collective information about the four qualities of every Web resource contained in the site. These four qualities are:

- **identity**, site owner's notion of what a resource is representing;
- **identifier** in the form of URI;
- **position** relative to other resources in the site, providing a navigational backbone to the site;
- **composition** of the resource – every resource in the site may be regarded as consisting of modular components, which are relatively independent and may be repeatedly used in various resources on the site.

The idea of UriGraph is to establish the framework for mapping the identifiers of the Web site's resources to the other qualities, primarily identity and composition. The first part of the process based on this mapping is to analyse the request (basically, the path and query parts of the URI in the HTTP request) and the second part is synthesis, where the information extracted from the request is incorporated in the appropriate response.

For the purposes of this paper, only a part of the UriGraph model concerning the analysis of the identifiers is described in detail.

Resource identifier model

The restricted UriGraph model described here is used to define all URIs valid for a specific Web site.

Topology

The structure of the Web site modelled with UriGraph is represented as a directed graph with two distinct types of nodes: **places** and **transitions**. There is one prominent place called the **root** node. Any two nodes may be connected with at most one directed edge in each direction, except for following restrictions:

- a node cannot be directly connected to itself;
- two places cannot be directly connected;
- there can be at most one edge leading from any transition to a place.

Places are depicted with small circles and transitions with wide shapes described later. Edges are arrows. The root node is marked with a symbol of a house (alluding to "home page").

Places represent classes of resources in the site. The root node represents the home page of the site. Transitions define what segments should be added to the URI for them to identify another class of resources or how to supply additional resource identity information to the same class of resources.

A walk is defined as a sequence of nodes in which any two consecutive nodes in the sequence are connected by an edge in the same direction. A **passage** through the graph is defined as a walk that begins with the root node and ends with a place. For example, in Figure 2 we can find these passages: (p_1), (p_1, t_1, p_1), (p_1, t_2, p_2), (p_1, t_1, p_1, t_2, p_2), etc. Total number of unique passages is infinite, due to the loop containing transition t_1.

Analysis

The UriGraph request consists of the path and the query (sequences of path and query segments, respectively). The request is empty if both the path and the query are empty sequences.

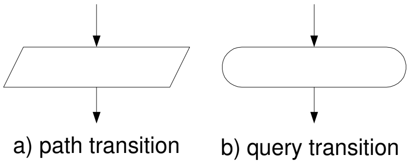
The analysis starts at the root node with a full request and regularly finishes at some place with an empty request, thus constructing a passage through the graph. If the analysis finishes at some transition, the request is said to be incomplete at that node of the graph, and the analysis is considered unsuccessful.

Each node in the constructing walk is processed, starting at the root node. To **process** a node means to: (1) trim the request by one path or query segment, and (2) find the next node, append it to the walk and continue the analysis by further processing it.

Segment trimming

Step 1, trimming, happens only in transitions. There are two types of transitions: **path transitions** which trim path segments and **query transitions** which trim query segments. Only the first segment in the path transition may be trimmed – in query transition, segments are tested sequentially from the first to the last, and the first appropriate segment is trimmed.

Graphical representation of a path transition is a parallelogram, side lines resembling the slashes delimiting the path segment. Analogously, a query transition has curved side lines resembling parentheses, as in the (name, value) pair. Both are shown on Figure 1.



a) path transition b) query transition

Figure 1: Two types of transition nodes

Figure 2 displays a simple graph with both types of transitions. The graph contains a loop (a transition connected back to the same node) and a cascade (a transition "bridging" two nodes).

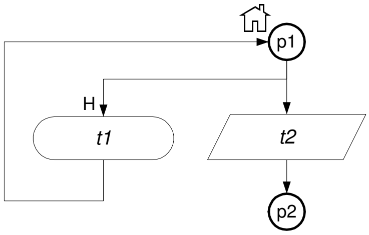


Figure 2: An example of UriGraph topology.

XML grammar used to describe UriGraph consists of element `structure` enclosing the list of elements `place`, `pathTransition` and `queryTransition` representing various types of nodes. Each node has an identifier used for connecting in sub-element `connectTo`.

```
<structure rootId="p1">
   <place id="p1">
      <connectTo id="t1" priority="H"/>
      <connectTo id="t2"/>
   </place>
   <queryTransition id="t1">
      <connectTo id="p1"/>
   </queryTransition>
   <pathTransition id="t2">
      <connectTo id="p2"/>
   </pathTransition>
   <place id="p2"/>
</structure>
```

Listing 1: XML description of the example graph in Figure 2.

Passes

To find the next node in the passage (step 2) one has to establish which of current node's destination nodes are passable. If there is only one passable node, that node is taken to be the next node. If there are no passable nodes, the request is said to be unprocessable. If there are more then one passable node, the request is said to be ambiguous at that node. If the request is unprocessable or ambiguous, the analysis is considered unsuccessful and is halted.

Every place is defined to be passable. To determine if the transition is passable, special logical function called **pass** is introduced. A pass evaluates to true (passable) or false (not passable) depending on the part of the request being tested (either the top path segment or any query segment) and sometimes on the state of the analysis (context information).

Passes are located in transitions. Each transition directly contains exactly one pass, but a pass may be composite (composed of other passes). An example of an atomic (non-composite) pass is the fixed pass, testing if the segment equals a constant string value. Example of a composite pass is conjunctive pass, returning true (passable) only if all its subpasses evaluate to true.

Edge priorities

In some cases it is necessary to establish priority relation on the set of destination nodes for some node *n*, defining the order of testing passability and selecting the next node in the analysis. Of course, there still remains the possibility that no nodes are passable (unprocessable request) or that two or more passable nodes have the highest priority in the set (ambiguous request).

UriGraph uses so-called HNL model of priority markings when assigning priorities to edges. Each marking is a string of letters 'H' (representing high value of priority), 'N' (normal) and 'L' (low priority). Marking has a higher priority then some other marking, if it has a higher-priority letter in the first place the two markings differ.

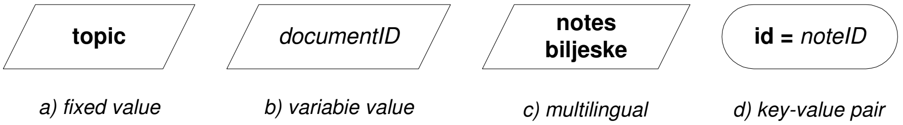
In graphical representation, priority marking is placed near the arrow end of the edge it refers to in a visually unambiguous way (Figure 2). No marking indicates default (normal) priority.

Implementations of passes

To implement the standard patterns of URI segment design, several classes of passes in UriGraph are described. The pass of the transition is graphically represented by a text or other symbols inside the shape representing a node – specific representation depends on the definition of the pass.

Fixed string value pass

The simplest case is when a segment should be some fixed string value. Fixed values in transition's passes are denoted with bold text, like in Figure 3a.



a) fixed value *b) variabie value* *c) multilingual* *d) key-value pair*

Figure 3: Examples of basic conventions for graphical representations of passes.

Variable string value pass

Sometimes the value of the segment or one of its parts is not known from the structure itself, but from the content of the Web site. In these cases special passes should be constructed to evaluate whether the string from the segment is from a set of permissible values, e.g. one of the identification codes stored in the database. Italic text represents variable value of the segment, with a set of permissible values hinted by the variable name, like in Figure 3b.

Multilingual pass

UriGraph's implementation of multilingual URIs consists of a composite language pass and a built-in languages clue. A languages clue containing the set of all the languages that the site uses is added to the response at the start of analysis. Whenever passing a transition node depends on the language of the segment, a language pass is used to filter through only the language(s) that apply. If the current languages clue does not contain any of the specified languages, that pass evaluates to false (not passable). Pass is represented as text in multiple lines, in the consistent globally defined language order (Figure 3c). Example XML description is shown in Listing 2.

Some resources have the same identifier in many languages: a typical example is the home page. This situation may be remedied with the inclusion of a simple query node, like in Figure 4.

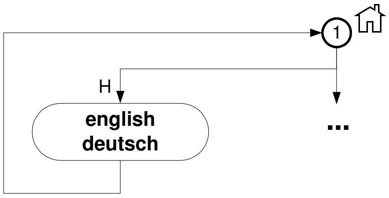


Figure 4: Query node for determining the language (English or German) of the resource.

This query node does not add any clues to the response, but its existence can identify the requested language. This construction can be used to sort out any other language ambiguities.

```
<pass class="LangPass">
   <translation lang="EN" segment="english"/>
   <translation lang="DE" segment="deutsch"/>
</pass>
```

Listing 2: Pass from Figure 4 in XML.

Pass for multipart segments

Sometimes the content of a particular segment consists of distinct parts. For example, the first part may be fixed string "image" and the second part may be a variable image identifier, separated by a dash. The multipart pass is developed for the purpose of dissecting the segment into parts delimited by the predefined character, and then applying the subpasses to each of the parts in sequence. Some parts later in sequence may be omitted, if the pass is marked as having optional parts.

Multipart pass is depicted using combination of bold and italic formats, signifying fixed and variable parts of the segment. The delimiting character is also displayed in bold formatting.

```
<pass class="MultipartPass" delimiter="-">
   <pass class="LangPass">
      <translation lang="EN" text="image"/>
      <translation lang="HR" text="slika"/>
   </pass>
   <pass class="ImageIdPass"/>
</pass>
```

Listing 3: Example of pass for multipart segments in XML.

Traversing limit pass

Sometimes loops in graph's topology may allow unlimited number of appearances for some segments. A pass has been constructed to limit traversing through a transition or a group of transitions.

Each time a transition is processed and traversed, a counter assigned to the traversing limit pass at that node is decreased by one. When the counter reaches zero, the pass remains impassable. Starting value of the counter (the limit) is defined at design time.

Traversing limit passes have identifiers and multiple passes at different nodes may share the same identifier. When the counter in one of them is decreased, the same applies to all of them – in other words, they share the same counter. This construct may be useful in cases when several transitions are alternative.

Case study: Author's personal Web site

To illustrate the use of UriGraph in designing URIs, the design of author's personal Web site was chosen for a case study. This site was chosen for two reasons: presenting a realistic design process and having complete control over requirements. In the time of writing this paper, the site is not operational due to technical issues, but hopefully it will be at the time of paper's publishing. Even then the site will probably be lacking some features described in the case study, until the platform capable of deploying described designs is made practical enough for public use.

Requirements

Site's purpose is to provide basic information about the author and his work accurately, straightforwardly, and timely.

User profiles

Several profiles of site's user are targeted:

- general population (anybody who may be interested in basic information);
- potential employers or partners, who would want a short overview of my work, biography, and contact info;
- professionals interested in my work;
- friends, colleagues and acquaintances.

Features

Based on the user profiles, the site should have the following features:

- news – short bulletins about recent events in author's life;
- quick facts – concise elementary information about the author;
- biography;
- collection of author's publications, with descriptions and downloadable full text documents, if available;

- notes section – an area of the site in which various documents containing author's professional notes can be stored and organized in arbitrary tree-like structure;
- content-only display – displaying only the content of the page, in a custom XML grammar;
- printer-friendly display – displaying the page for easy printing (stripped of navigation controls, all content on one page, etc.);
- multilingual content, with Croatian and English translations of the most important resources.

Design details

Some interesting and characteristic details of URI design using UriGraph are shown.

Domain name

Croatian DNS registration service offers a package for personal use containing three domain names *xyz*`.from.hr`, *xyz*`.iz.hr` ("iz" means "from") and *xyz*`.name.hr`, where *xyz* is a person-specific identifier, usually the person's name. So the two domain names were registered as English and Croatian versions of the site: `http://hrvoje-simic.from.hr` and `http://hrvoje-simic.iz.hr`.

Cascading transitions for publications

A typical pattern in UriGraph designing is a cascade of path transitions. A simple example in the case study is design of publications.

For example, we can have "`/publication`" representing author's publication in general, "`/publication/cuc2002`" for this paper etc. The choice of singular for noun "publication" may seem strange, but it is compatible with the idea of resources representing concepts, so the resource representing the concept of Hrvoje Simic's publication should be at the URI "`http://hrvoje-simic.from.hr/publication`". The resource may display a list of concrete publications, but should not be regarded as a resource representing only a list of publications.

Section of the site concerning publications is implemented with one multilingual pass and one variable value pass (Listing 4). The last pass in the listing is upgraded to include format info, as explained below.

```
<place id="p-root"/>
   <connectTo id=" t-publication"/>
</place>
<pathTransition id="t-publication">
   <connectTo id="p-publication"/>
   <pass class="LangPass">
      <translation lang="EN" segment="publication"/>
      <translation lang="HR" segment="publikacija"/>
   </pass>
</pathTransition>
<place id="p-publication">
   <connectTo id=" t-publication-id"/>
</place>
<pathTransition id="t-publication-id">
   <connectTo id="p-specific-publication"/>
   <pass class="PublicationPass"/>
</pathTransition>
```

```
<place id="p-specific-publication"/>
```

Listing 4: Use of cascading transitions for publications section of the site in XML.

Qualifiers "content" and "printable"

For the features of content-only and printer-friendly displays, a pair of query transitions loops is connected to the root node (Figure 5). For example, to get only the contents of the home page, agent should dereference the URI "`http://hrvoje-simic.from.hr/?content`". This enables the use of the each feature for every resource on site. Where the use of the feature is not applicable, the content/printable query segment is ignored.

Since the two features are mutually exclusive, a special traversing limit pass is constructed to prevent both "switches" to be included in the same request. Both passes use the same counter ID, and both are limited to one traverse, so there is no passage through the graph containing both transitions.

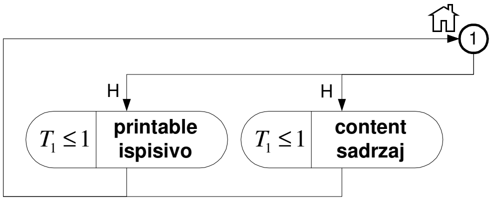


Figure 5: Query transition loops implementing printable and content-only displays.

```
<queryTransition id="t-content">
   <connectTo id="p-root"/>
   <pass class="TraversePass" key="content-or-printable" limit="1"/>
   <pass class="LangPass">
      <translation lang="EN" segment="content"/>
      <translation lang="HR" segment="sadrzaj"/>
   </pass>
</queryTransition>
<queryTransition id="t-printable">
   <connectTo id="p-root"/>
   <pass class="TraversePass" key="content-or-printable" limit="1"/>
   <pass class="LangPass">
      <translation lang="EN" segment="printable"/>
      <translation lang="HR" segment="ispisivo"/>
   </pass>
</queryTransition>
```

Listing 5: Transitions with traversing limit passes implementing printable and content-only displays in XML.

Topic structure

For the notes section of the site URI designer should provide a construction enabling the content author to construct simple tree-like structure of topics, for example "web", "web/resource" or "web/resource/definition". Each topic can be associated with a variable set of documents. Resource corresponding to particular topic should provide current information about the topic and associated documents, while the documents (notes) itself are more static.

Unknown URI structure consisting of several path segments is implemented as a loop path transition with variable pass of topic identifiers.

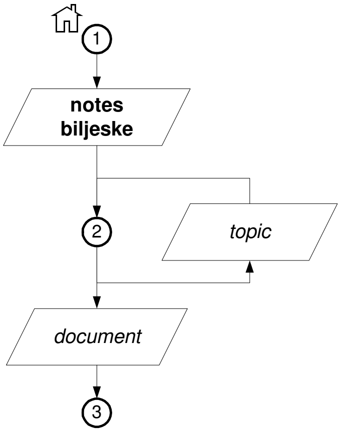


Figure 6: Topic structure graph fragment.

```
<place id="p-notes">
   <connectTo id="t-topic" priority="H"/>
   <connectTo id="t-notes-doc"/>
</place>
<pathTransition id="t-topic">
   <connectTo id="p-notes"/>
   <pass class="TopicPass"/>
</pathTransition>
<pathTransition id="t-notes-doc">
   <connectTo id="p-notes-doc"/>
   <pass class="NotesDocPass"/>
</pathTransition>
<place id="p-notes-doc"/>
```

Listing 6: Topic structure in XML (partial description of graph in Figure 6).

Composition of this part of the URI is more defined by the content then the structure of the Web site. This phenomenon could be called semi-structured site, and the content of such a site is often semi-structured data itself.

File type extensions in URIs

Sometimes URIs contain file type extensions borrowed from the file system, such as "`.html`" or "`.gif`". In recommendations for URI content, we cautioned against including this information when unwarranted, especially since technologies and formats change. However, if the format type is an integral part of the resource identity, the URI should include that extra information, and a good way is adding popular file type extensions. So, we can have URI `/publication/masters-thesis` for a general page representing the thesis (as with a simple cascading transition designed explained before) and `/publication/masters-thesis.pdf` for the thesis in PDF format. The designer should be careful not to commit to specific formats very often, because they are bound to change, compromising the persistence of designed URIs.

```
<pathTransition id="t-publication-id">
   <connectTo id="p-specific-publication"/>
   <pass class="MultipartPass" delimiter="." optional="true">
      <pass class="PublicationPass"/>
      <pass class="FormatTypePass"/>
   </pass>
</pathTransition>
```

Listing 7: Multipart pass used for file type extensions in XML.

Conclusion

This paper attempted to identify cases for using URIs, as well as URI design requirements. To achieve those requirements, a set of recommendations is given, as well as a model for describing URI subspace design for a Web site. The model is general enough to describe any design, but hopefully also straightforward enough to be useful in designing even the simplest sites.

The application of UriGraph goes beyond URI design. Based on the URI analysis, information about resource identity is collected, as well as its position and composition. This idea is in accordance with the approach in which designing URIs is at the heart of designing Web site as a whole, combining structure with content, look and functionality.

References

[bone02a] Bone, Jeff: "*Query Strings Considered Harmful*", http://internet.conveyor.com/RESTwiki/moin.cgi/QueryStringsConsideredHarmful

[bone02b] Bone, Jeff: "*Generative Naming*", http://internet.conveyor.com/RESTwiki/moin.cgi/GenerativeNaming

[bone02c] Bone, Jeff: "*Opacity Reconsidered*", http://internet.conveyor.com/RESTwiki/moin.cgi/OpacityReconsidered

[cha01] Champin, Pierre-Antoine; Jérôme Euzenat; Alain Mille: "*Why URLs are good URIs, and why they are not*", May 2001, http://www710.univ-lyon1.fr/~champin/urls/

[gtb] Google Toolbar, http://toolbar.google.com/

[nie00] Nielsen, Jakob: "*Designing Web Usability: The Practice of Simplicity*", New Riders Publishing, Indianapolis, 2000, ISBN 1-56205-810-X, http://useit.com/jakob/webusability/

[nie98] Nielsen, Jakob: "*Fighting linkrot*", Alertbox, June 1999, http://www.useit.com/alertbox/980614.html

[nie99] Nielsen, Jakob: "*URL as UI*", Alertbox, March 1999, http://www.useit.com/alertbox/990321.html

[rdfms] Lassila, Ora; Ralph R. Swick (editors): "*Resource Description Framework (RDF) Model and Syntax Specification*", W3C Recommendation, February 1999, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[rfc2396] Berners-Lee, T.; R. Fielding; L. Masinter: "*Uniform Resource Identifiers (URI): Generic Syntax and Semantics*", RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt

[rfc2616] Fielding, R.; J. C. Mogul; H. Frystyk; L. Masinter; P. Leach; T. Berners-Lee: "*Hypertext Transfer Protocol -- HTTP/1.1*", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[sim02] Simic, Hrvoje: "*Modelling of WWW site structure*", Master's Thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, May 2002 (In Croatian)

[tbl92] Berners-Lee, T.: "*Style guide for online hypertext*", http://www.w3.org/Provider/Style/

[tbl96a] Berners-Lee, T.: "*The Myth of Names and Addresses*", December 1996, http://www.w3.org/DesignIssues/NameMyth.html

[tbl96b] Berners-Lee, T.: "*Universal Resource Identifiers – Axioms of Web Architecture*", *W3C Design Issues*, December 1996, http://www.w3.org/DesignIssues/Axioms.html

[tbl96c] Berners-Lee, T.: "*Generic Resources*", 1996, http://www.w3.org/DesignIssues/Generic

[urcr] "*URIs, URLs, and URNs: Clarifications and Recommendations 1*.0", Report from the joint W3C/IETF URI Planning Interest Group, W3C Note, September 2001, http://www.w3.org/TR/uri-clarification

[wordnet] Wordnet, http://www.cogsci.princeton.edu/~wn/