Filling in the Blanks: Archiving Dynamically Generated Content

Justin F. Brunelle
Old Dominion University
Department of Computer Science
Norfolk, Virginia, 23508
jbrunelle@cs.odu.edu

ABSTRACT

Web 2.0 technologies are improving the average Web user's browsing experience by providing richer browsing. HTML5, Ajax, and other technologies allow the user to interact with – and sometimes generate – content on the client-side. Pages can be customized based on user preferences, user interaction, and other local events. These events and interactions have the potential to generate, change, and alter client-side content and representations. However, these events, interactions, and dynamically generated representations are as ephemeral (if not more) as the browsing session itself; dynamic content is lost when the browsing session ends and cannot be retrieved. This proposed body of work will provide the ability to capture, share, and archive the results of these ephemeral and fleeting interactions, effectively capturing a portion of the Web previously not captured by crawlers.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Data Sharing

General Terms

Design, Experimentation

Keywords

Web Architecture, HTTP, Web Archiving, Digital Preservation

1. MOTIVATION

Current archival efforts and information retrieval efforts cannot exploit an extremely important aspect of the Web. The most effective automatic digital preservation is being performed in the live Web by use of crawlers. These crawlers start at a particular Web resource and follow all of the links extending from that site. This allows these crawlers to automatically exploit the linked nature of the Web to discover and archive new resources in a recursive fashion. Crawlers, however, are limited in the resources they can access and render. They cannot access resources that are not linked from the seed sites or subsequent sites that are visited. Further, crawlers simply access the static representations of the sites. Some resources on the Web have representations that change as a result of user interaction. Such resources are cannot be captured by traditional methods, such as through crawlers and other automated methods. [12, 3, 18]. Additionally, these resources are not accessible with a single HTTP GET request. Crawlers do not have the ability to interact with a site to change the representation, or initiate client-side events that may spawn new content. This dynamic portion of the Web is comprised of several types of resources.

This research aims to bring this portion of the Web to the surface so that users can share dynamic content, and archivists can capture it. Current archival methods are able to archive and capture static resources, and single, independent resource representations. With the increased popularity of custom and dynamic resources, such as those implemented in Web 2.0 with Ajax and JavaScript, there is a large portion of the Web that is unable to be archived or shared. Current Web users cannot share a representation of a resource that is generated as a result of user interactions. For example, when a Web 2.0 resource is loaded, it is identified by a Universal Resource Identifier (URI) [5, 4, 13, 29]. When the Web user interacts with the page, data is transferred from the server back to the client in the form of Ajax requests. The client-side JavaScript interprets the data received from the server and modifies the client-side HTML Document Object Model (DOM) as a result. The HTML DOM is the hierarchy of tags generating the local markup, and normally is represented with a tree structure.

Such interactions can be observed in applications such as Google Maps, Facebook, Google News, and others. User interactions can initiate events that change the representation and appearance of the resource without changing the URI. For example, if a user wished to demonstrate the capabilities of Google Maps, a textual description may be possibly confusing, where an archived interaction and associated representation of a resource will be infinitely more instructive to the user. A user may be able to give an instruction, such as "open Google Maps, plug in the address, zoom in about 1/3 of the way, scroll a little to the left, and look under the second tree to find out where I've hidden my buried treasure." However, it would be much easier to record the interactions necessary to find this location, and share the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

interactions and representations with a canonical URI. Similarly, how does one explain what the dangers of Hurricane Irene were in the Tidewater area to family over seas? The items of importance were difficult to share in high fidelity. A screenshot, such as that seen in Figures 1(a) and 1(b) is insufficient to share and convey full effect of the situation. The URI is the same for both images, and an interaction was necessary to get the provided representation. Such resources can, and should, also be archived so that future accessors of the URI will see the same playback as the originating user.

This example is further investigated with a browsing session. Normal browsing sessions occur as depicted in Figure 2. The browser issues a HTTP request for a resource, and the server provides the resource to the browser. The browser then renders the representation. However, when requesting an interactive resource, the browsing session will make an initial HTTP Request for the main content, followed by several requests for embedded content. The page is then rendered in the browser. Then, the JavaScript embedded in the page makes additional requests to the server for additional content. This content is only retrieved after the main page content is rendered. Figure 3 illustrates this alternate sequence.

These steps can be replicated manually. This resource is obviously accessible via `cURL`:

```
curl -i http://hamptonroads.com/2011/08/
map-mandatoryevacuation-areas-norfolk

HTTP/1.1 200 OK
X-Static: vpapp1/p
Date: Thu, 12 Apr 2012 10:05:00 GMT
Server: Apache
Last-Modified: Thu, 12 Apr 2012 10:01:51 GMT
...
```

All constituent resources must be identified and downloaded. While these are all available, one observes that the Google Maps API is embedded in the page and makes calls back to the Google Maps servers:

```
<script type="text/javascript"
 src="http://maps.google.com/maps/
api/js?sensor=false">
</script><script type="text/javascript"
id="script">

var layer = 'full address';

var tableid = 1349029;

var center = new
google.maps.LatLng(36.901862,-76.267433);

var zoom = 12;

var map;

var geocoder = new google.maps.Geocoder();

....

// Use the geocoder to geocode the address
  geocoder.geocode(
{
'address': document.getElementById("address").value
},
function(results, status) {

    // If the status of the geocode is OK

    if (status == google.maps.GeocoderStatus.OK) {

      // Change the center and zoom of the map

      map.setCenter(results[0].geometry.location);

      map.setZoom(16);
```

Only after this resource is retrieved and loaded does the JavaScript load the map and focus on the target area. Additionally, it is clear that the API is making calls to the server for additional data and content when investigating the content of `http://maps.google.com/maps/api/js`:

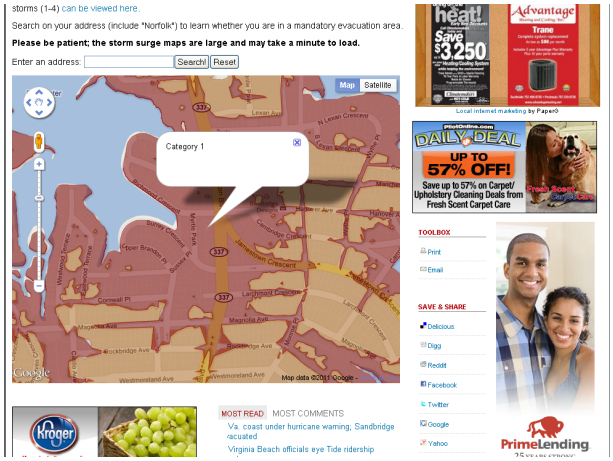
```
getScript("http://maps.gstatic.com/intl/en_us/
mapfiles/api-3/8/7/main.js");
...
function getScript(src) {
    document.write('<' + 'script src="' + src +
 '"' + ' type="text/javascript">
<' + '/script>');
  }
```

This function retrieves and loads a JavaScript library after the page load. When running a `wget -p -H` to download all prerequisite resources regardless of their host address, the `http://maps.gstatic.com/intl/en_us/mapfiles/api-3/8/7/main.js` file is omitted. This is an example of a page prerequisite that is not known unless the page is rendered.

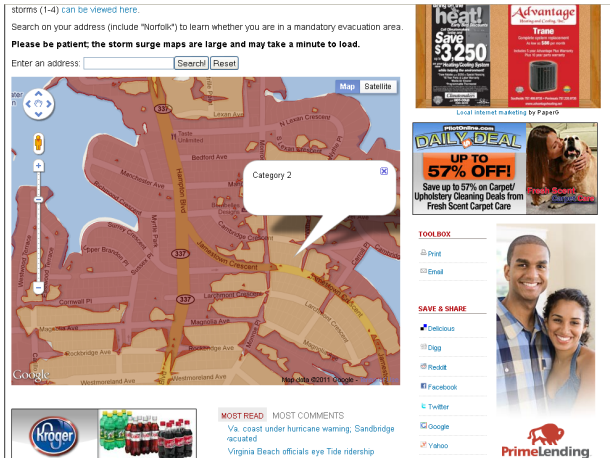
Further more, this example illustrates another problem that exists with resources relying on Web 2.0 technologies for their content. The goal of archival efforts is to preserve digital content. However, even when archived, this resource will reach out of the archive back to the server containing the data needed to render the map tiles. The entirety of the representation is never fully archived because the data cannot be capture with the client-side content. When the server containing the data is no longer accessible, the representation will no longer show the state of the resource when it was archived. This can be seen in the differences in a live and archived resource that includes an interactive map as shown in Figures 4(a) and 4(b), respectively.

2. ILLUSTRATIVE EXAMPLE

On August 27th, Hurricane Irene made landfall in the Outer Banks of North Carolina. Hurricane Irene affected much of the East Coast. However, it is the approach of Hurricane Irene that is most interesting for the purposes of this case study. Hurricane Irene approached the East Coast as a Category 4 hurricane. The panic created by this hurricane was hard to explain and describe, particularly to those disconnected from the situation. For demonstration purposes, this paper assumes a PhD student named Justin lives in Norfolk, VA. His parents were on a cruise in the Mediterranean



(a) Screenshot of an interactive flood map (http://hamptonroads.com/2011/08/map-mandatoryevacuation-areas-norfolk).



(b) Screenshot of an interactive flood map, at another location (http://hamptonroads.com/2011/08/map-mandatoryevacuation-areas-norfolk).

Figure 1: Different representations of the same resource generated by user interaction. URI does not change during client-side events, and state change is not represented.

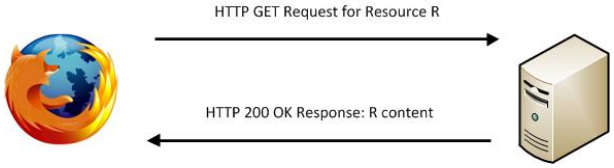


Figure 2: Normal request-response interaction between client and server for a resource R.

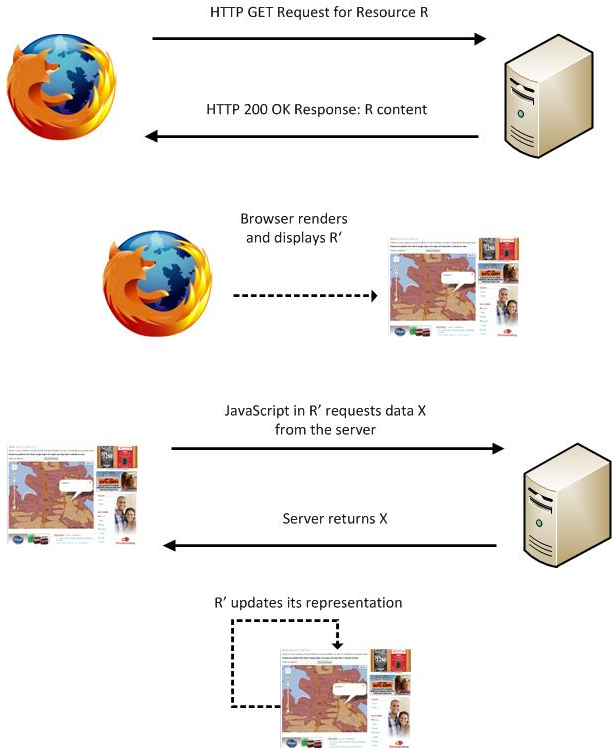
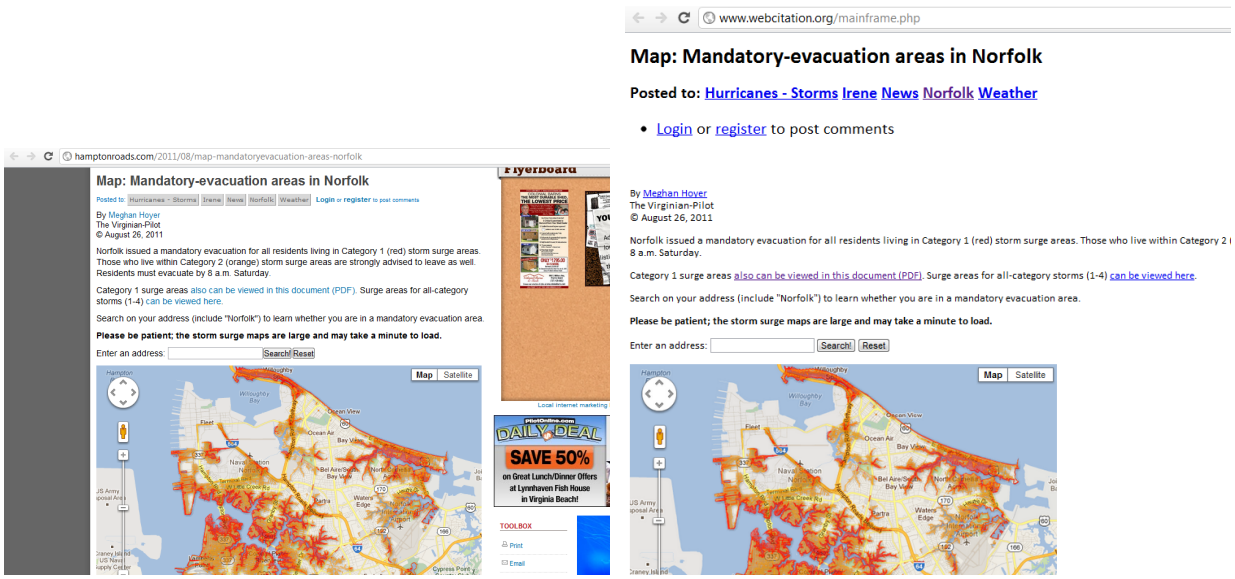


Figure 3: Request-response interactions between client and server and page and server for resource R and embedded content.



(a) Live resource

(b) Resource archived at WebCitation

Figure 4: The same resource as it exists in the live Web (Figure 4(a)) and the same resource as archived in WebCitation (Figure 4(b)).

in late August of 2011. As the Irene approached the East Coast and the storm became more dangerous and frightening, Justin tried to explain the situation to his parents. His parents were skeptical of the imminent danger of the storm and of the near-pandemonium it was creating all along the East Coast. Justin was unable to accurately share representations of web resources that explained the situation to with his parents, mainly because there was no single resources containing all of the information. What Justin needed was a way to navigate between resources, interact with them, and share the representation of an aggregate and interactive viewing session.

2.1 Evacuation

Justin lives in a low lying area in Norfolk, VA. The local news paper provided a list of mandatory evacuation areas on a web resource. The live shot from August is provided in Figure 4(a). The resource is still available as a live resource (at the time of this publication) at `http://hamptonroads.com/2011/08/map-mandatoryevacuation-areas-norfolk`. To identify whether or not Justin has to evacuate, one must interact with the map to determine under what evacuation level his house falls. Figure 1(a) demonstrates the result of zooming in to Justin's house and identifying his evacuation level. Justin must evacuate since the mandatory evacuation level is at category 1. Alternatively, to find out if Dr. Nelson must evacuate, we can pan to his neighborhood and find that he falls under a category 2 evacuation level and does not have to evacuate.

In order for Justin to share this with his parents overseas, he had to provide a URL to the resource and instructions on how to interact with it. (Zoom in, pan to to the north east, click on my house, etc.) This is an inaccurate way to share information, and a way to share these interactions and resulting representation would have been helpful. To take this a step further, how can this resource be shared across time? Let's say Justin wanted to archive this representation. He tries to push it to WebCitation to preserve the representation of the resource. The final result of the archived resource is seen in Figure 4(b). Aside the from the loss of the stylesheet, the resource is seemingly archived in tact. However, upon further examination, one can see that the map is generated live from the Google Maps API. If the data on the Google Maps server changes, so does the representation loaded with this archived resource. Also, the interactions aren't capture. It's worth noting that the Internet Archive has no memento of this resource.

2.2 Storm Surge

The evacuations in Norfolk were spurred by the threat of storm surge[1]. Again, Justin would like to share the predicted storm surge with his parents to show the impending danger for his neighborhood. The live, interactive resource showing the storm surge is shown in Figure 5. Again, the representation focusing on Justin's house requires user interaction to generate. Further, this resource is unarchivable. As seen in Figures 6 and 7, the Internet Archive and WebCitation cannot accurately capture this resource for archival purposes.

2.3 Panic

[1]http://www2.timesdispatch.com/news/2011/aug/27/tdmain01-irenes-dangers-on-virginias-doorstep-ar-1267221/

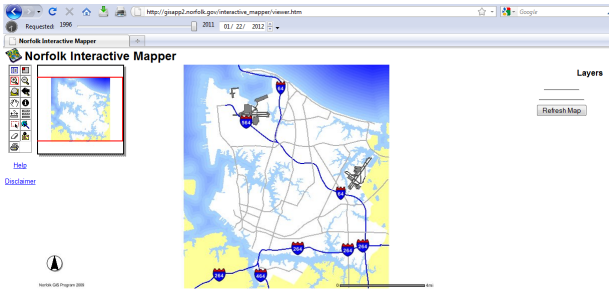


Figure 5: Screenshot of an interactive storm surge map.

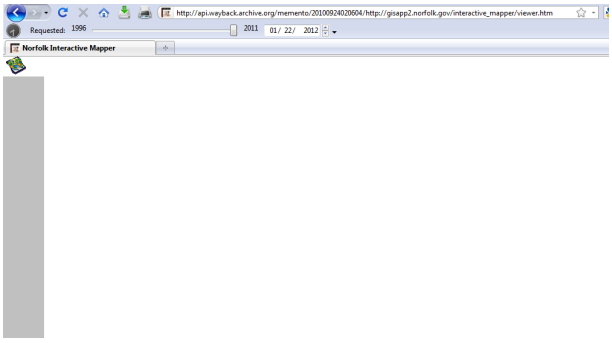


Figure 6: Screenshot of Internet Archive version of the storm surge map.

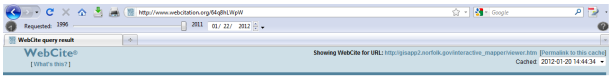


Figure 7: Screenshot of the Webcitation copy of the storm surge map.

The most difficult thing to share was the panic in the US over Hurricane Irene. Justin tried to communicate the severity of the situation with his parents. He mentioned that New York city had begun evacuations. These stories were met with disbelief and responses such as "They wouldn't evacuate NYC." The flights into JFK were being canceled, and Justin's parents were supposed to fly in to JFK the day the Hurricane was going to make landfall in New York. Justin mentioned that their flight would be canceled, and this story resulted in more disbelief. Justin's parents, without adequate understanding of the situation, make further plans to fly in to JFK and drive down the Eastern seaboard to their residence in Virginia Beach. Justin tried to explain that most of the east coast would be evacuated, but again,

his "fabrications" weren't believed. No single resource explained the entirety of the situation. All Justin could do was provide a set of links and a set of instructions for interacting with the resources. Additionally, some of the resources were mashups of things like Twitter feeds or interactive maps. For example, during the evacuation of the Outer Banks, the traffic at one point was backed up from Hampton, through Norfolk, Virginia Beach, and the rest of the OBX; North- and West-bound traffic was at a standstill for an entire day before the hurricane. This was temporal and dynamic information lost due to the inability to archive the dynamic representation, data, and content and share it across time and browsing sessions.

2.4 Aftermath

Hurricane Irene made landfall as a category 1 hurricane. This was much weaker than anticipated[2]. Justin's parents got home a few days after the hurricane left the East Coast. Due to the rapid weakening of the hurricane just before landfall, the damage and effects of the hurricane were much more mild than expected. This made Justin's concerns and the panic of the public during the approach of the hurricane seem unjustified. Without the ability to share the resources or recall archived mementos, Justin could not accurately demonstrate the actual panic that preceded the storm. Without the ability to archive his browsing sessions, Justin has lost the ability to recall the preceding events forever.

3. RELATED WORK

Other efforts have explored stateful Ajax and JavaScript applications, such as the HashBang URI schema of Google[3][4][5] and several research efforts to crawl this dynamic portion of the Web by discovering the states of JavaScript applications [21, 1, 10]. Still other research has tried to make Ajax applications stateful instead of allowing them to remain as stateless applications [16, 19]. These solutions are not suitable for archiving content, and are most certainly not able to archive and share a static representation of a resource that was generated as a result of client-side events. The HashBang URIs from Google make dynamic content shareable with URIs generated from client-side state. Twitter makes extensive use of HashBang URIs. For example, a user's list of Tweets is pulled from a server after the page is loaded. A HashBang URI designates the client-side state of Justin's Tweets List: `https://twitter.com/#!/justinfbrunelle`. Different technologies for synchronizing Ajax communications with a server have also been explored [23] and may prove useful to this research. Several projects have also focused on monitoring data that flows from the client to the server [20, 9, 31, 30]. Most of this research is for security purposes, but the concepts can be extended and implemented in this research.

Mapping software has long suffered from some of the same research questions that this research will investigate. For example, online mapping systems have tried to cache the map tiles to make them more quickly retrieved from the server. These tiles were stored locally, and able to be transferred between browsers [22]. There has also been work done in the area of caching JavaScript code [15, 24]. More interestingly to this work, an effort to cache the actual data being sent between the client and server through Ajax has been studied [6, 2]. This caching can be replicated in this research in the form of encoded data that is captured and shared with the client-side representations.

Walden's Path [26, 14, 27, 11] is a teaching tool that directs students through a series of resources, allows them to interact with the resource, even navigate away from the target page, and then return to the "path" at the click of a button. The goal is to provide a user a replay of user interactions and resource representations, allow the viewing user to possibly interact with the archived version, but also allow the user to see what the originating author saw in high fidelity.

4. FUTURE EXPERIMENTS AND PLANS

The problem at hand is observable by all Web users. However, measuring the prevalence of the problem remains to be performed. To capture this measurement, an experiment is being conducted to measure the "archiveability" of bit.ly targets being shared over Twitter. The study downloads the live DOM (after the page has loaded), a static image of the live site, and downloads the full page (`wget -pk <URL>`) to the local drive. The page is then observed from the local drive and compared to the live version. Differences in DOM and image are noted. Additionally, the local page is observed with all Internet connectivity removed, essentially mimicking the behavior that would be seen if the page existed in an archive and all live constituent resources had been lost. This experiment is currently in progress.

The main challenge of this experiment is defining a vector of metrics that will reliably define a difference in live vs. non-live representations. It has the benefit of providing a metric measuring "What can we archive?" among inherently popular resources. It will also provide a measurement of the difference between client-side capture and the archiving performed by the Internet Archive and WebCitation. Preliminary results can be observed in Figure 8. This figure demonstrates how resource availability and download mechanism can change the representation.

An unanswered question is to measure how dynamic content is generated, how it can be captured and replayed, and how much content is lost when this sort of interaction is not archived. An experiment must be designed to capture this metric. Essentially, one must understand how client-side inputs effect the representations (as seen in Figure 9).

Measuring the differences between archived and live Web 2.0 resources will help measure the prevalence and effect of this problem. After this experiment is completed, it will be important to understand how much archives are missing. Specifically, an experiment should be performed to determine how much content is missing from an archived copy of a resources versus a live resource, as well as determine how many archived resources reach into the live Web. The papers resulting from this future work could be suitable for conferences such as SIGIR, JCDL, or TPDL.

The eventual culmination of this body of work is to allow users to record interactions with a Web 2.0 resource, archive the data, events, and representations of the session, and pro-

[2]http://hamptonroads.com/2011/08/storm-surge-now-worry-irene-hampton-roads

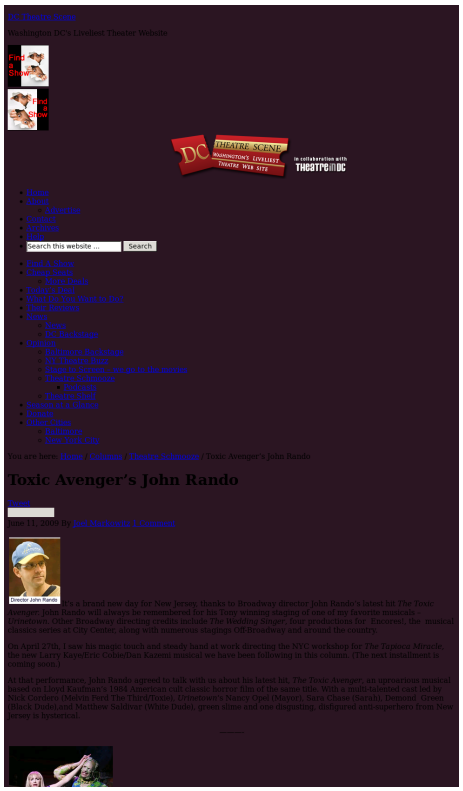
[3]http://www.seomoz.org/blog/how-to-allow-google-to-crawl-Ajax-content,

[4]http://mtrpcic.net/2011/02/fragment-uris-theyre-not-as-bad-as-you-think-really/

[5]http://www.jenitennison.com/blog/node/154

(a) Live resource

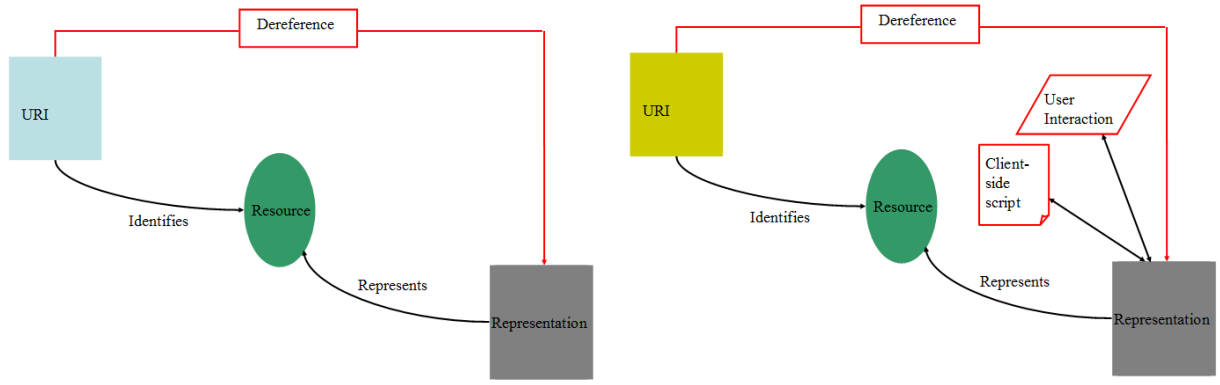
(b) Local resource with Internet



(c) Local resource without Internet

(d) Local resource at WebCitation

Figure 8: The same resource (http://dctheatrescene.com/2009/06/11/toxic-avengers-john-rando/) as it exists on the live Web (8(a)), locally downloaded with access to the Internet (8(b)), locally downloaded with no Internet access (8(c)), and the resource archived at WebCitation (8(d))



(a) Traditional dereferencing leads to a representation

(b) Interactive representation generation

Figure 9: Dereferencing URIs is no longer the only thing that generates a final representation – client-side events modify the representation, as well.

vide access to the interactions and changes in the DOM that are incurred as a result of the user-induced events. This VCR-like activity will record user browsing sessions to an archive on command, and allow for sharing and replay.

This proposed research will design a system that can extract a representation of a resource, capture the user interactions with the resource, store the content transferred between server and client, and store the changed representation as a function of the user interaction and original state. The contributions of this research will be threefold: the first known way to record, share and archive user interactions with a Web 2.0 resource, the the crowd-sourced the capture of an entire portion of the Web, and the capability to exchange persistent Web 2.0 representations during an entire browsing session. Essentially, this research will create a VCR for Web sessions in which a user can record, share, and play back interactions that are previously lost in an ephemeral medium.

This system will be embodied first as a Mozilla browser add-on. The add-on will fit in between the server and the browser, and listen for changes on the client. This is shown in Figure 10. The add-on will have VCR-like controls: play, stop, and record. The play button will start recording a session. This will start with reading and storing the original client-side representation and code. Subsequent events will be captured and recorded, along with the results of each event. For example, a user may click a button that retrieves server-side data, and then displays the retrieved data as a form on the client-side page. The add-on will capture and record the event and store the data retrieved from the server. The new representation will be recorded, as well. With this information and data, the tool will be able to replay the interaction on command. Feasibly, a user can even navigate between resources and record the change in pages.

When the user stops recording the session, the collected data, HTML, and events will be packaged and sent to an archive to be stored. The archive will handle the change detection between representations of resources, and provide a single, canonical URI for representing the newly archive content. Upon accessing this content, a user can use the Play button to play back the representation that was recorded by the original Web user.

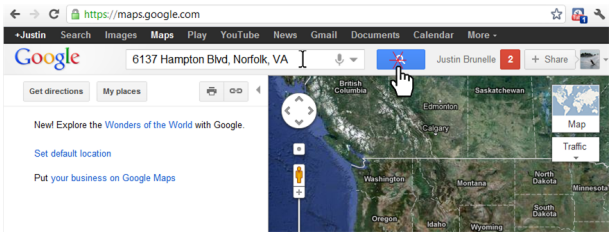
Capturing the initial representation and subsequent events, data, and changes in HTML DOM will be performed with jQuery, JavaScript, and listening to HTTP Headers [28] being sent to and from the server and client. The archival packaging will be performed using OAI-ORE [17, 25], which is a standard format to represent an archived version of a page that is made of multiple constituent parts (such as images, CSS, etc.) needed for the representation. The playback will be performed by displaying the original representation, and artificially kicking off the original user's events through JavaScript. The data and resulting changes to the representation caused by these events will be fed to the client. That is, if an event normally kicks off an Ajax call to a server to ask for data, the call will instead retrieve data from the archive. The rest of the changes in the representation should happen organically, meaning the code already on the client should handle the code as if it were still live.

To share the resource, a URI will be provided by the archive that points to the archived session. Eventually, the goal is to have a URI that a user can access from any browser or computer. However, initial attempts may require that replay be performed from a computer and browser equipped with the Mozilla add-on.

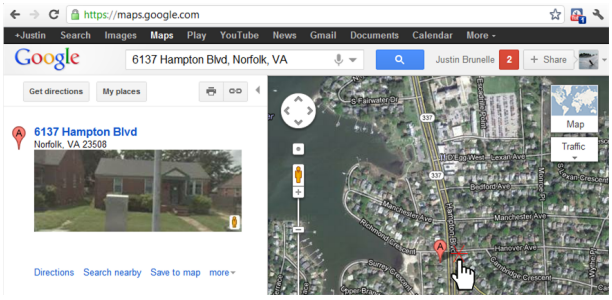
This solution should be studied and compared to the previous research. Specifically, the ability to capture previously unarchiveable content and to prevent archived resources from reaching into the live Web for content should be measured. Since a goal of the tool is to provide a way to archive Web 2.0 resources, the content produced by the tool should be studied to ensure fidelity; the produced content should be compared to live content for completeness, correctness, and acceptance by the popular archives.

4.1 Comparison Metrics

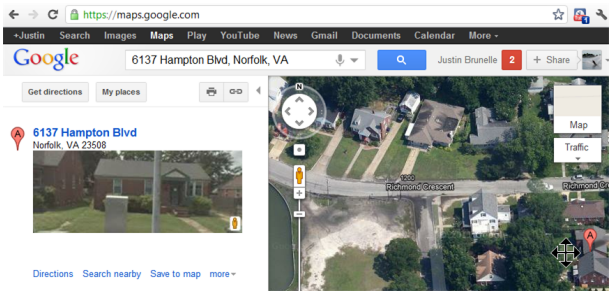
Without proper experimentation, baselines, and metrics, this research is little more than a systems engineering so-



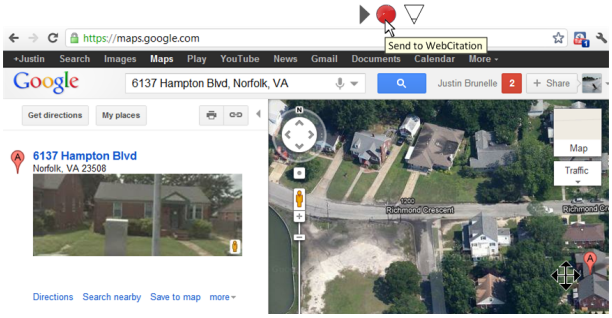
(a) User interacts with the representation: Typing and button clicks



(b) User double-clicks on the map to zoom in.



(c) User clicks and drags to pan West on the map.



(d) Add-on will allow the user to submit the final representation to the archive.

Figure 10: The add-on will listen to and record changes on the client-side of the architecture, and allow the user to replay client-side events.

lution. Along with the aforementioned experiments, there must be a set of metrics for comparing systems, contributions of the systems, and the improved coverage of the digital repositories. This is the current effort of the author. This investigation will provide a set of metrics for future experimentation, including comparing live, archived, and modified representations of resources.

5. BENEFITS OF THE RESEARCH

The benefits of this research are many. For example, companies always gain new members midway through the life cycle of the project. To get the team members caught up, a series of resources and instructions on interactions can be constructed and exchanged. The team member can then follow the instructions provided, and try to absorb the content. This research can make it possible to share a single, canonical URI that the new member can play, interact with, and explore to gain a richer, more beneficial, and faster introduction to the necessary project-specific material. Further, this research can create bookmarks for Web 2.0 representations, regardless of the interactions that were needed to create this particular representation. This can fix some of the anomalies seen in social bookmarking tools. For example, a user can interact with a dynamic representation, but cannot bookmark the representation that she saw during her browsing session; she can only bookmark the target URI, which currently does not give state to the representation. A future effort of this work could be to plug this tool into Onomi (MITRE's corporate social bookmarking tool) [7, 8], Deli.cio.us, or other tool for sharing and preserving Web and digital content.

With the increasing popularity of social bookmarking, such as Delicious and Onomi, this research can be implemented as a way to create more permanent bookmarks, and also bookmark and share resource representations. Currently, if a bookmarked resource had a representation generated through user interactions, a bookmark cannot accurately link to or describe the representation the user saw during the action of bookmarking. Further, the user cannot share the bookmark with another user. Since the representation exists as as a function of the user interactions, client and server state, this representation could never be recreated.

Finally, this research will allow for richer communications, archives, and usability of Web 2.0 and dynamic resources. Justin will be able to share his concerns about future hurricanes, and archive static representations of news-worthy events. This will provide a way to capture dynamic content.

6. CONCLUSIONS

In conclusion, this research will provide a study of a growing portion of the Web: dynamically generated content. This will provide the ability to capture the representations of dynamic resources, share them across browsing sessions, and archive previously unarchivable content. It will be general enough to be applied to any resources, and powerful enough to capture all data required to generate an accurate and faithful representation. Before this end solution is achieved, comprehensive studies of dynamic resources will be conducted. This will allow researchers to understand how resources change, how users interact with content to generate additional content, and what steps can be taken to capture dynamic representations for archival, information sharing, and information retrieval purposes.

7. ACKNOWLEDGMENTS

Justin F. Brunelle's advisor is Michael L. Nelson. This work is supported in part by NSF grant 1009392.

8. REFERENCES

[1] K. Benjamin, G. von Bochmann, M. Dincturk, G.-V. Jourdan, and I. Onut. A strategy for efficient crawling of rich internet applications. In S. Auer, O. Diaz, and G. Papadopoulos, editors, *Web Engineering*, volume 6757 of *Lecture Notes in Computer Science*, pages 74–89. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-22233-7_6.

[2] E. Benson, A. Marcus, D. Karger, and S. Madden. Sync kit: a persistent client-side database caching toolkit for data intensive websites. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 121–130, New York, NY, USA, 2010. ACM.

[3] M. K. Bergman. White paper: Deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1), September 2001.

[4] T. Berners-Lee. What do HTTP URIs Identify? http://www.w3.org/DesignIssues/HTTP-URI.html.

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

[6] P. Cao, J. Zhang, and K. Beach. Active cache: caching dynamic contents on the web. In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, Middleware '98, pages 373–388, London, UK, 1998. Springer-Verlag.

[7] L. Damianos, J. Griffith, and D. Cuomo. Onomi: Social bookmarking on a corporate intranet. Technical Report 06-0352, The MITRE Corporation, 2006.

[8] L. E. Damianos, D. Cuomo, J. Griffith, D. M. Hirst, and J. Smallwood. Exploring the adoption, utility, and social influences of social bookmarking in a corporate environment. *Hawaii International Conference on System Sciences*, 0:86b, 2007.

[9] M. Dhawan and V. Ganapathy. Analyzing information flow in JavaScript-based browser extensions. *Annual Computer Security Applications Conference*, pages 382 – 391, 2009.

[10] C. Duda, G. Frey, D. Kossmann, and C. Zhou. AjaxSearch: crawling, indexing and searching web 2.0 applications. *Proc. VLDB Endow.*, 1:1440–1443, August 2008.

[11] R. Furuta, F. M. Shipman, III, C. C. Marshall, D. Brenner, and H.-w. Hsieh. Hypertext paths and the World-Wide Web: experiences with Walden's Paths. In *Proceedings of the eighth ACM conference on Hypertext*, HYPERTEXT '97, pages 167–176, New York, NY, USA, 1997. ACM.

[12] K. Hagedorn and J. Sentelli. Google still not indexing hidden web URLs. *D-Lib Magazine*, 14(7), August 2008. http://dlib.org/dlib/july08/hagedorn/07hagedorn.html.

[13] IANA. Permanent URI Schemes. https://www.iana.org/assignments/uri-schemes.html.

[14] U. Karadkar, L. Francisco-Revilla, R. Furuta, H. wei Hsieh, and F. Shipman. Evolution of the walden's paths authoring tools. In *WebNet'00*, pages 299–304, 2000.

[15] R. Karri. Client-side page element web-caching. December 2009. http://scholarworks.sjsu.edu/etd_projects/137.

[16] E. Kiciman and B. Livshits. Ajaxscope: A platform for remotely monitoring the client-side behavior of web 2.0 applications. In *the 21st ACM Symposium on Operating Systems Principles (SOSP'07)*, SOSP '07, 2007.

[17] C. Lagoze, H. Van de Sompel, P. Johnston, M. Nelson, R. Sanderson, and S. Warner. Open Archives Initiative Object Reuse and Exchange. 2008. http://www.openarchives.org/ore.

[18] F. McCown, X. Liu, M. L. Nelson, and M. Zubair. Search engine coverage of the OAI-PMH corpus. *IEEE Internet Computing*, 10:66–73, 2006.

[19] A. Mesbah, E. Bozdag, and A. van Deursen. Crawling Ajax by inferring user interface state changes. In *Web Engineering, 2008. ICWE '08. Eighth International Conference on*, pages 122 –134, july 2008.

[20] L. A. Meyerovich and B. Livshits. Conscript: Specifying and enforcing fine-grained security policies for JavaScript in the browser. *Security and Privacy, IEEE Symposium on*, 0:481–496, 2010.

[21] J. Mickens, J. Elson, and J. Howell. Mugshot: deterministic capture and replay for JavaScript applications. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 11–11, Berkeley, CA, USA, 2010. USENIX Association.

[22] D. S. Myers, J. N. Carlisle, J. A. Cowling, and B. H. Liskov. Mapjax: data structure abstractions for asynchronous web applications. In *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, pages 8:1–8:14, Berkeley, CA, USA, 2007. USENIX Association.

[23] @NesbittBrian. Play framework sample application with JWebUnit and synchronous ajax. http://nesbot.com/2011/10/16/play-framework-sample-app-JWebUnit-synchronous-ajax.

[24] S. Periyapatna. Total recall for Ajax applications - firefox extension. 2009. http://scholarworks.sjsu.edu/etd_projects/139/.

[25] D. Reynolds. OAI-ORE model for sample article 1. 2009. https://wiki.library.jhu.edu/display/DATAPUB/OAI-ORE+Model+for+Sample+Article+1.

[26] F. Shipman, C. Marshall, R. Furuta, D. Brenner, H. wei Hsieh, and V. Kumar. Creating educational guided paths over the World-Wide Web. *Ed-Telecom 96*, pages 326–331, 1996.

[27] F. M. Shipman, III, R. Furuta, and C. C. Marshall. Generating web-based presentations in spatial hypertext. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, IUI '97, pages 71–78, New York, NY, USA, 1997. ACM.

[28] W3C. HTTP request fields. http://www.w3.org/Protocols/HTTP/HTRQ_Headers.html.

[29] W3C. UriSchemes. http://www.w3.org/wiki/UriSchemes.

[30] D. Yu, A. Chander, N. Islam, and I. Serikov. JavaScript instrumentation for browser security. In *Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '07, pages 237–249, New York, NY, USA, 2007. ACM.

[31] P. Zeng, J. Sun, and H. Chen. Insecure JavaScript detection and analysis with browser-enforced embedded rules. *International Conference on Parallel and Distributed Computing Applications and Technologies*, 0:393–398, 2010.