How to Follow Software Users

Lev Manovich
softwarestudies.com

March 2012

"Big data" is the new media of 2010s. Like previous waves of computer technologies, it changes *what* it means to know something and *how* we can generate this knowledge. Thanks to the efforts of Digital Humanities Office[1] at National Endowment of Humanities (U.S. National funding agency for humanities research) which together with other agencies organized Humanities High Performance Computing program in 2008, and Digging Into Data competitions in 2009 and 2011, as well as the few pioneering projects by individual humanities researchers and their labs, big data paradigm has now entered the humanities.

An article in *New York Times* from November 16, 2010 (part of their Humanities 2.0 series) boldly stated: "The next big idea in language, history and the arts? Data."[2] While digitization of historical cultural artifacts seemingly produces such data, representing *cultural activities* as data which can be analyzed with computers is a challenge in itself.

So far, all big data projects in digital humanities that I am aware of used *digitized cultural artifacts* from the past. If we want to apply the big data paradigm to the study of contemporary *interactive software-driven media*, we are facing fascinating theoretical questions and challenges. What exactly is "big data" in the case of interactive media? How do we study the interactive temporal experiences

[1] (http://www.neh.gov/odh/.
[2] http://www.nytimes.com/2010/11/17/arts/17digital.html.

of the users, as opposed to only analyzing the code of software programs and contents of media files?

This article provides possible answers to these questions. They are based on the research at Software Studies Initiative which I direct at University of California, San Diego (UCSD) and California Institute for Telecommunication and Information (Calit2), and our current NSF Eager funded project where we are analyzing the records of the interactions of large number of users with massive virtual world Scalable City.[3]

Software Studies Initiative was founded in 2007 to work on two interrelated programs: 1) study of software and its use in contemporary societies using the methods of humanities, social sciences and new media theory; 2) study of cultural contents and cultural processes using software-based methods. In other words, software for us is both a subject of investigation, and an intellectual technology which joins other existing humanities methods for reading culture. The first perspective is a part of "software studies" agenda as it is usually understood[4]; the second falls within "digital humanities."[5] (Since this term is currently used very broadly to include all kinds of digital work in humanities ranging, from curating online collections to computational analysis and visualization of big cultural data, we use a separate term "cultural analytics" to refer to the latter.)

[3] http://www.scalablecity.net/.
[4] See the introduction to Software Studies series at The MIT Press: http://mitpress.mit.edu/catalog/browse/browse.asp?btype=6&serid=179.
[5] http://lab.softwarestudies.com/2007/05/about-software-studies-ucsd.html.

Although during 20+ years of its existence, new media studies (and its more recently developed "software studies" part) generated thousands of books and conferences and tends of thousands of papers that already analyzed many dimensions of interactive media. (The MIT Press alone currently lists 368 books in its "new media" category[6]). However, if we want to use big data paradigm to study interactive media, we need to interrogate this type of media theoretically from additional perspectives. The first part of my article offers this analysis; the second uses the result of the analysis to offer a methodology for the study of interactive media as "big data."[7]

What is the "Data" in Interactive Media?

The use of software re-configures most basic social and cultural practices and makes us rethink the concepts and theories we developed to describe them. As one example of this, consider the modern "atom" of cultural creation, transmission, and memory: a "document", i.e. some content stored in a physical form, which is delivered to consumers via physical copies (books, films, audio record), or electronic transmission (television). In software culture, we no longer have "documents," "works," "messages" or "recordings" in 20th century terms. Instead of fixed documents whose contents and meaning could be determined by

6 http://mitpress.mit.edu/catalog/browse/browse.asp?cid=12&btype=1.

7 This article further develops the ideas which I first articulated in the proposal *Digging Into Born Digital Data: Methods For The Analysis Of Interactive Media*, prepared by me, Jeremy Douglass, Matthew Fuller, and Olga Goriounova for 2009 Digging Into Data Competition. The text of the article is based on my introduction to the book manuscipt *Software Takes Command* (revised 2012 version) currently under consideration by The MIT Press.

examining their structure and content (a typical move of the 20th century cultural analysis and theory, from Russian Formalism to Literary Darwinism[8]) we now interact with dynamic "software performances." I use the word "performance" because what we are experiencing is constructed by software in real time. So whether we are exploring a dynamic web site, play a video game, or use an app on a mobile phone to locate particular places or friends nearby, we are engaging not with pre-defined static documents but with the dynamic outputs of a real-time computation happening on our device and/or the server. Computer programs can use a variety of components to create these performances: design templates, files stored on a local machine, media from the databases on the network server, the real-time input from a mouse, touch screen, joystick, our moving bodies, or another interface, etc. Therefore, although some static documents may be involved, the final *media experience constructed by software usually does not correspond to any single static document stored in some media*. In other words, in contrast to paintings, literary works, music scores, films, industrail designs, or buildings, a critic can't simply consult a single "file" containing all of work's content.

Even in such seemingly simple cases such as viewing a PDF document or opening an photo in a media player, we are already dealing with "software performances" - since it is software which defines the options for navigating, editing and sharing the document, rather than the document itself. Therefore examining the PDF file or a JPEG file the way twentieth century critics would examine a novel, a movie, or a TV program will only tell us some things about the experience we get when we interact with this document via software – but not everything. This experience is equally shaped by the interface and the tools provided by software. This is why the examination of the tools, interfaces, assumptions, concepts, and the history of cultural software – including the

[8] http://en.wikipedia.org/wiki/Darwinian_literary_studies, acccessed March 14, 2012.

theories of its inventors who in the 1960s-1970s have defined most of these concepts – is essential if we are to make sense of contemporary media.

This shift in the nature of what constitutes a media "document" also calls into question well-established cultural theories that depend on this concept. Consider the intellectual paradigm that dominated the study of media since the 1950s – "transmission" view of culture developed in Communication Studies. Communication scholars have taken the model of information transmission formulated by Claude Shannon in his 1948 article *A Mathematical Theory of Communication* (1948)[9] and the subsequent book published with Warren Weaver in 1949,[10] and applied its basic model of communication to mass media. The paradigm described mass communication (and sometimes culture in general) as a communication process between the authors who create and send "messages" and audiences that "receive" them. According to this paradigm, the messages were not always fully decoded by the audiences for technical reasons (noise in transmission) or semantic reasons (they misunderstood the intended meanings.)

Classical communication theory and media industries considered such partial reception a problem; in contrast, in his influential 1980 article "Encoding/decoding"[11] the founder of British Cultural Studies Stuart Hall argued that the same phenomenon is positive. Hall proposed that the audiences construct their own meanings from the information they receive. Rather than being a communication failure, the new meanings are active acts of intentional reinterpetation of the sent messages. But both the classical communication

[9] C.E. Shannon, "A Mathematical Theory of Communication", *Bell System Technical Journal*, vol. 27, pp. 379–423, 623-656, July, October, 1948.
[10] Claude E. Shannon, Warren Weaver. *The Mathematical Theory of Communication.* Univ of Illinois Press, 1949.
[11] Hall, Stuart (1980): 'Encoding/decoding'. In Centre for Contemporary Cultural Studies (Ed.): *Culture, Media, Language*. London: Hutchinson.

studie and cultural studues implicitly took for granted that the message was something complete and definite – regardless of whether it was stored in physical media (e.g., magnetic tape) or created in real time by a sender (a live TV broadcast). Thus, the receiver of communication was assumed to read all of advertising copy, see a whole movie, or listen to the whole song and only after that s/he would interpret it, misinterpret it, assign her own meanings, appropriate it, remix it, etc.

While this assumption has already been challenged by the introduction of DVR (digital video recorder) in 1999 that led to the phenomenon of *time shifting*, it simply does not apply to interactive software-driven media. The interfaces of media access applications, such as web browsers and search engines, the hyperlinked architecture of world wide web, and the interfaces of particular online media services which offer large numbers of media artifacts for playing, preview and/or purchase (Amazon, Google Play, iTunes, Rhapsody, Netflix, etc.) encourage people to "browse", quickly moving instantly both *horizontally* between media (from one seach result to the next, from one song to another sond, etc.) and vertically, *though* the media artifacts (e.g., from the contents listing of a music CD to a particular track). They also made it easy to start playing/viewing media at an arbitrary point, and leave it at any point. In other words, the "message" which the user "receives" is not just actively "constructed" by her (through a cognitive interpretation) but also actively "managed" (defining what information she is receiving and how.)

It is at least as important that when a user interacts with a software application that presents media content, this content often does not have any definite finite boundaries. For instance, a user of Google Earth is likely to experience a different "earth" every time she is accessing the application. Google could have updated some of the satellite photographs or added new Street Views; new 3D buildings, new layers and new information on already existing layers were also likely to be added. Moreover, at any time a user of the application can load more geospatial data created by other users and companies by either selecting one of the options in Add menu (Google Earth 6.2.1 interface), or directly opening a KLM file. Google Earth is a typical example of a new type of media enabled by the web – an interactive “document” which does not have all of its content pre-defined. Its content changes and grows over time.

In some cases this may not affect in any significant way the larger “messages” “communicated” by the software application, web service, a game, or other type of interactive media. For example, in the case of Google Earth, regardless of which layers are turned on and which new content has been added by users since your last visit, this does not affect one of its built-in conventions – representation of the earth using the General Perspective Projection (a particular map projection method of cartography[12]).

However, since the user of Google Earth can also add her own media and information to the base representation provided by the application, creating complex and media rich projects on top of existing geoinformation, Google Earth is not just a “message.” It is a *platform* for users to build on. And while we can find some continity here with the users creative reworking of commercial media in the 20th century – pop art and appropriation, music remixes, slash fiction and video[13], and so on, the differences are larger than the similarities.

This shift from messages to platforms was in the center of the web transformation around 2004-2006, called Web 2.0. The 1990s web sites

[12] http://en.wikipedia.org/wiki/Google_earth#Technical_specifications, accessed March 14, 2012.

[13] See, for instance, Constance Penley, "Feminism, Psychoanalysis,and the Study of Popular Culture." In Grossberg, Lawrence, ed., *Cultural Studies* (Rutledge, 1992).

presenting particular content created by others (and thus, communicating "messages") were supplemented by social networks and social media sites where the users can share, comment on, and tag their own media. Wikipedia article on Web 2.0 describes these differences as follows: "A Web 2.0 site allows users to interact and collaborate with each other in a social media dialogue as creators (prosumers) of user-generated content in a virtual community, in contrast to websites where users (consumers) are limited to the passive viewing of content that was created for them. Examples of Web 2.0 include social networking sites, blogs, wikis, video sharing sites, hosted services, web applications, mashups and folksonomies."[14] For example, to continue with Google Earth cases, users added many types of global awareness information, including fair trade certification, Greenpeace data, and United Nations Millennium Development Goals Monitor.[15] In another example, you can incorporate Google Maps, Wikipedia, or content provided by most other large web 2.0 sites directly in your web mashup – an even more direct way of taking the content provided by web services and using it to craft your own custom platforms.

The wide adoption of Web 2.0 services along with various web-based communication tools (online discussion forums about every popular software, collaborative editing on Wikipedia, Twitter, etc.) enables quick identifications of omissions, selections, censorship and other types of "bad behavior" by software publishers – another feature which separates content distributed by web-based companies from mass media of the 20th century. For example, every article on Wikipedia about a Web 2.0 service includes a special section about controversies, criticism, or errors.

In many cases, people can also use alternative open source equivalents of paid and locked applications. Open source and/or free software (not all free software

[14] http://en.wikipedia.org/wiki/Web_2.0, accessed March 14, 2012.
[15] http://en.wikipedia.org/wiki/Google_earth, accessed March 14, 2012.

is open source) often allow for additional ways of creating, remixing and sharing both content and new software additions. (This does not mean that open source software always uses different assumptions and key technologies than the commercial software.) For exampe, one can choose to use a number of alternatives to Google Maps and Google Earth - OpenStreetMap, Geocommons, WorldMap, and others which all have open source or free software licenses.[16] (Interestingly, commercial companies also often use data from such free collaboratively created systems because they contain more information than the companies' own systems. For example, OpenStreet Map, which by early 2011 had 340,000 contributors[17], is used by Flickr and Foursquare.[18]) A user can also examine the code of open-source software to fully understand its assumptions and key technologies.

Continuosly changing and growing content of web services and sites; variety of mechanism for navigation and interaction; the abilities to add one own's content and mashup content from various sources together; architectures for colloborative authoring and editing; mechanisms for monitoring the providers – all these mechanisms clearly separate interactive networked software-driven media from the 20th century media documents. But even when a user is working with a single local media document that is stored in a single computer file (a rather rare situation these days), such a document mediated through software interface has different identity than a 20th century media document. The user's experience is still only partly defined by the file's content and its organization. The user is free to navigate the document, choosing both what information to see and the sequence in which she is seeing it. And while "old media" (with the exception of

---

[16] http://geocommons.com/, http://www.openstreetmap.org, worldmap.harvard.edu, accessed March 14, 2012.
[17] http://en.wikipedia.org/wiki/Counter-mapping#OpenStreetMap, accessed March 27, 2012.
[18] http://en.wikipedia.org/wiki/OpenStreetMap#Derivations_of_OpenStreetMap_Data, accessed March 27, 2012.

20th century broadcasting) also provided this random access, the interfaces of software-drivem media players/viewers provide many additional ways for browsing media and selecting what and how to access.

For example, Adobe Acrobat can display thumbnails of every page in a PDF document; Google Earth can quickly zoom in and out from the current view; online digital libraries, databases and repositories containing scientific articles and abstracts such as the ACM Digital Library, IEEE Xplore, PubMed, Science Direct, SciVerse Scopus, and Web of Science show articles which contain references to the one you currently selected.[19] Most importantly, these new tools and interfaces are *not hard- wired* to the media documents themselves (such as a random access capacity of a printed book) or media access machines (such as a radio); instead they are part of the separate *software* layer. This media architecture enables easy addition of new navigation and management tools without any change to the documents themselves. For instance, with a single click, I can add sharing buttons to my blog, thus enabling new ways of circulation for its content. When I open a text document in Mac OS Preview media viewer, I can highlight, add comments and links, draw and add thought bubbles. Photoshop allows me to save my edits on separate "adjustment layers," without modifying the original image. And so on.

How to Follow Software Users

These new properties of interactive software-driven networked media require new methods and new theoretical concepts for its study:

19 http://en.wikipedia.org/wiki/List_of_academic_databases_and_search_engines, accessed March 14, 2012.

1. We need to be able to record and analyze *interactive experiences*, i.e. concrete temporal interactions of particular users with the software – as opposed to only analyzing media "documents" (i.e., the elements which are used to construct these experiences). For example, we should follow the users as they navigate though a web site – as opposed to limiting ourselves to studying the content of this site. We should follow different players as they progress through the video game – as opposed to using only our own gameplay as the basis for the analysis. We should follow the visitors of an interactive installation as they explore the space of possibilities defined by the designer – the possibilities which only become actual events when the visitors act on them.

(Here I can make a parallel with the famous shift in science studies created by Bruno Latour and his colleagues when they turned attention from studying scientific documents to following scientists in the laboratories – articulated in a number of Latour's books such as *Science in Action: How to Follow Scientists and Engineers through Society.*[20] In a parallel fashion, we need to follow users as they interact with software, rather than analysing media documents by themselves.)

Why this is so important? What is the difference between capturing the process of user interaction with traditional media and using such records for analysis, versus doing the same with software-driven media? After all, we can use various technologies to capture traces of person's emotional and cognitive states while she is reading a book, watching a movie, moving through a built environment, or interacting with other older media forms. We can capture indications of brain activity using EEG or MRI; record eye movements; track pulse; capture galvanic skin response and use other biometric techniques. (By 2012, a number of

[20] Bruno Latour. *Science In Action: How to Follow Scientists and Engineers Through Society* (Harvard University Press, 1987).

neuromaketing companies were already working with Hollywood studios to record such information from test viewers, helping directors to refine editing of feature films and product placement[21] - a new service which originated in the academic research on "neurocinematics."[22])

When we record and analyze movements of a cursor on the screen, spatial trajectory of user-controlled game character, or other elements of an interactive experience with a software-driven media, there is a qualitative difference. What we are capturing and analyzing in all these cases is the actual "work" as it is *co-created by the software and user's interactions* – the work which only emerges during this interaction and which is different from session to session and from user to user. This is different from capturing biometric data of test audiences of a film which was created beforehand.

While all cultural experiences can be described as forms of interaction, software-driven interactive media represents a new form of human culture. Certainly, a recipient of a non-interactive text can be also said to cognitively construct her own version of the text. However, the actual material «input» (i.e., a text on the page, a movie shown in a movie theatre, an interior of a building, the shape of a product, etc.) remains the same for all the users (at least, currently). With software-driven interaction, this "input" is likely to change every time are you accessing the application. It is co-created by the software and my behavior at the moment of the interaction.

[21] http://www.fastcompany.com/1731055/oscars-avatar-neurocinema-neuromarketing, accessed March 14, 2012.
[22] Uri Hasson, Ohad Landesman, Barbara Knappmeyer, Ignacio Vallines, Nava Rubin, and David J. Heeger. "Neurocinematics: The Neuroscience of Film." *Projections*, Volume 2, Issue 1, Summer 2008: 1–26. http://www.cns.nyu.edu/~nava/MyPubs/Hasson-etal_NeuroCinematics2008.pdf.

For example, a video game which you are will be playing is likely to be different from the game I will be playing. Why? Because the probability that another player follows exactly the same spatial trajectory through a level, solving all puzzles in exactly the same sequence, and duplicating all my other actions is close to 0. (And since the game is likely to use a random number generator to control generation and actions of objects and enemies, this adds another level of variability.) Similarly, the contents of amazon.com home page will certainly be different every time I visit it, because the site software automatically customizes parts of page's content (”Your Recent History” area of the page, recommendations, etc.) based on the data about my previous browsing and purchasing behavior, and behaviors of people who made similar choices. And while at the moment of this writing the interfaces of computers and mobile devices do not change from one session to another – programs do not re-arrange themselves on the desktop, and document icons do not modify their color to reflect how often they have been accessed before – it is certainly feasible to give them more “intelligence”. (Researchers in the field called «augmented cognition» are developing interfaces and learning environments which would continuously change based on the input from sensors monitoring human bodily and brain activity.[23])

Any analysis of software-driven interactive media need to take into account its fundamental co-creation (by users and software) geneology, and its built-in variability. But if we can’t simply analyze media document, how to we go about capturing multiple “software performances” and user experiences?[24] In other words: *how do we follow software users*?

[23] http://www.augmentedcognition.org/, accessed March 12, 2012.
[24] Some of the points in the following analysis were originally developed in a grant proposal I submitted with Mathhew Fuller to 2009 Digging Into Data competition. Lev Manovich and Mathew Fuller, PIs. *Digging Into Born Digital Data: Methods For The Analysis Of Interactive Media*, 2009.

Culture always involved interactions between audiences and cultural objects, or between performers and participants. However, until recently recording and analyzing these interactions was difficult and expensive. Only Hollywood movie studies and big advertising agencies could afford special facilities where simple emotion responses of selected viewers to commercials or movie endings could be recorded and combined with other focus group techniques such as interviews and questioners. As already mentioned above, now capturing emotional responses can be done with EEG, MRI and other biometric techniques, but most of them are still rather expensive, and their use and interpretation requires special training. However, tracking users interaction with software itself is trivial.

Why? Because the very phenomenon of interaction is made possible by software continuously monitoring the interface inputs. Without this monitoring, there is no interaction. A software application is continuously capturing key presses, mouse movements, menu selections, finger gestures over a touch surface, voice commands, and other types of inputs. These recorded inputs trigger various actions provided by the application – zooming into a location on a map, firing a weapon in a game, altering an image with a paintbrush tool, finding the web pages which match user query, and so on.

Graphical User Interfaces (GUI) used today by all application software also require capturing many kinds of user input for another reason. If you use UNIX, you know that after you enter a command, often you don't get any feedback – unless you did not use command syntax correctly and then you get an error message. Similarly, the edits in HTML code do not become visible until you preview HTNL document in a web browser. But in programs which use GUI, all user input is explicitly acknowledged. Computer provides immediate textual, visual and/or sound feedback. If I am editing a document, it immediately shows my changes. In fact, as I am writing this sentence, every time I press a key on the keyboard, Microsoft Word captures this key press and immediately updates the document shown on the screen. (This interface can be said to simulate 19th century typewriter interface where pressing a key resulted in a physical movement which printed the corresponding letter on the paper.) Another examples of how interfaces captures and stores a sequence of user inputs is a History window (or *history* command in UNIX) which displays the sequence of all entered commands during a user session.

Since capturing user inputs is already a fundamental principle of modern interactive software interfaces, the software can be easily modified to store values of these inputs – rather than erasing them as soon as the action is triggered. (This can be as simple as writing each new value of a variable to a file – an action which can be added with a single line of code.) While in principle recording and storing users inputs was always possible from the early days of software culture, the shift from desktop to web computing in the 1990s have turned this possibility into a fundamental component of a “software-media complex.” Since dynamic web sites and services are operated by software residing on company's servers, it is easy to log the details of their interactions. For example, each web server keeps detailed information on all visits to a given web site. A separate category of software and services exemplified by Google Analytics emerged to help a user analyze this information so it can be used to fine tune the web design.

Following initial example of Google Analytics introduced in 2005, today social media companies make available to the users some of the recorded information about visitors’ interactions with their site, blog, or account; the companies even provide interactive visualizations to help them figure out patterns and make their web offerings more successful. I can study this personal analytics for my web site and also for our lab blog softwarestudies.com, our Tube and Flickr accounts, and our Facebook page, seeing the impact of each separate blog post, photo, video, and update. The companies also make available to public some of this information for all *web* pages or accounts. For example, YouTube displays statistics below every video, showing graphically the number of views of over time, the important referrals, and basic demographics of the visitors. However, other captured data remains private and is used to improve algorithms, generate recommendations, and serve ads.[25]

Game companies capture many details of all gameplay in their massively multiplayer online games (MMOs); this information is then used internally by the companies to refine game design. During development of new games, the companies are also systematically record and analyze the game play of dozens of testers (this practice is called “game analytics.”) Another example of how recording and using information about users' interactions with software is at the core of contemporary web and media industries is provided by Google search algorithms.[26] While the algorithms use hundreds of variables to calculate which web pages are most relevant to a particular query, one of the inputs is the information about which of the pages previously returned to the same query other users have clicked on.

Social media and e-commerce sites such as YouTube and Amazon also capture and analyze user inputs, feeding this data into their recommendation engines. As Frank Kessler and Mirko Tobias Schaffer point out in the case of YouTube, the operations which for a user appear as being marginal to the primary activity of watching videos – tagging, commenting, and reporting some videos as inappropriate - are central to the service operations in general: “All these

[25] See http://en.wikipedia.org/wiki/Social_search, http://en.wikipedia.org/wiki/Recommendation_system, http://en.wikipedia.org/wiki/Semantic_targeting, accessed March 14, 2012.
[26] http://www.google.com/competition/howgooglesearchworks.html, accessed March 27, 2012.

operations that YouTube offers its users – or rather which must be used for YouTube to generate metadata necessary for its functioning – are at first sight ancillary options and additional services. Quite on the contrary, however, they actually provide the indispensable basis of the database's information management… In fact, every single click on one of these links to a clip, however random or accidental this choice may be, does feed into a database as well.[27]

Since YouTube site not explain its technologies, and the papers presented by its engineers at conferences only go into details about some of them, I can't verify if all details in this analysis is correct. [28] We also don't know whether all types of user captured and stored by YouTube are actually used in its algorithms. Finally, what ever is captured certainly does not get stored in a single database. (Part of Software Studies mission is to encourage scholars to be more precise in their analysis of software, because all these details matter.) However, the general argument by Kessler and Schaffer is correct - web and media companies do capture and parse user inputs in order to refine their products and offerings, and this captured data often is very important to their business, even though for users this may not be fully visible.

[27] Frank Kessler and Mirko Tobias Schaffer, «Navigating YouTube: Constituting a Hybrid Information Managment System.» Jean Burgess, Patricia G. Lange and Mirko Tobias Schafer, eds. *The YouTube Reader* (National Library of Sweden, 2009), 284-285.

[28] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems* (RecSys '10). ACM, New York, NY, USA, 293-296. DOI=10.1145/1864708.1864770. http://doi.acm.org/10.1145/1864708.1864770; Renjie Zhou, Samamon Khemmarat, and Lixin Gao. 2010. The impact of YouTube recommendation system on video views. In *Proceedings of the 10th annual conference on Internet measurement* (IMC '10). ACM, New York, NY, USA, 404-410. DOI=10.1145/1879141.1879193 http://doi.acm.org/10.1145/1879141.1879193.

Cultural disciplines should also take advantage of these kinds of data (and yes, of course you should ask users first if its OK to record it and analyze it). Too often researchers rely only on their intuitions and personal experience with these products. Having to face the actual facts about the actual behavior of many concrete users can be quite refreshing. In practice, this means that in addition to theorizing about an imaginary abstract user (typical strategy in humanities), or using surveys to ask people to report on their cultural experiences, or doing ethnography (communication studies), we can also capture and subsequently analyze data about people's actual interactions with software artifacts.

In doing this, we will align cultural analysis with the processes of culture production in software society - using the same methods, but for different purposes. For example, the designers of software applications and content capture data about experiences of *test* users, and then use this data to refine their offerings. As cultural critics, we may capture the data about the experiences of the *actual* users in order to better understand what these experiences are, map their variability, and – most importantly - compare them across many products to make visible larger cultural patterns. For example, if an analytics team at a particular game company usually only analyzes the games currently being developed only at this company, we can take a broader look - analyzing gameplay across many games (within a single genre or multiple genres), or the different versions of game during decades of their evolution. [29] (If you are getting scared reading this proposal for adopting the software-based techniques used by

[29] Ben Middler looks at the history of gameplay recordings and its uses in his 2009 article "Generations of Game Analytics, Achievements and High Scores", *Eludamos. Journal for Computer Game Culture*, Vol 3, No 2 (2009).http://journals.sfu.ca/eludamos/index.php/eludamos/article/viewArticle/66.

companies to create media for its analysis by humanists, I am sorry for dragging you into the 21st century..)

Since we usually don't have acess to the code of commercial media products, we can use other methods for capturing interactions. One method that is already been systematically used in my Software Studies lab since 2008 is capturing video of a gameplay.[30] Our approach is turn each game session into an equivalent of a film by recording a video of a gamer's screen during the gameplay. (This process is similar to the use of screen capture software to produce training video for software applications.) Once we create such a recording, we can use it to analyze a game experience as a process of changes over time. We can graphically represent rhythms and patterns in a single game play session. We can also aggregate the data from many sessions to visualize a space of possible experiences enabled by a particular game. We can further create visual "landscapes" of larger areas of video game culture, with sets of data representing different gameplay sessions of single game, and supersets of data representing the comparison of many games. Finally, by including even more data, the historical patterns in game culture can be also visualized. (Since many important historical games are available today in software emulation, this method works well as a way to understand games history.)

Note once again the fundamental difference between capturing an interactive media experience vs. the recording the experience of a static media document (for example, doing eye movements recording while a person is watching a video ad). Recording the video game player's screen captures not just *how you experience* but also the concrete media artifact *constructed by a user and*

[30] For the examples of our projects using this method, see: http://lab.softwarestudies.com/2008/06/videogameplayviz-analyzing-temporal.html; http://lab.softwarestudies.com/2012/02/kingdom-hearts-game-play-visualizations.html.

software during a particular session – in other words, the *object of the experience*.

While at present researchers in game studies usually base their analysis on their own engagement with interactive media, there are good reasons to propose that a collection of such recordings - representing many sessions by many users - should be taken as a basic "unit" of cultural analysis of this type of media. This idea can also be extended to all other types of interactive media. Thus, along with using her own experience of a particular interactive artifact, a researcher can record a set of sessions (ideally, representing as many diverse users as possible) which together will form a representative sample of the whole space of possibilities which constitute the “interactive text.”

2. The specificity of interactive media also challenges the meaning of the concept “big data” as applied to the study of culture. The already mentioned grant programs that were established in 2008-2009 by U.S. National Endowment for Humanities' Office for Digital Humanities have called for humanists to start applying the computational techniques for large-scale data analysis already standard in many areas of science. The description of the second joint NEH/NSF Digging into Data Competition (2011) opened with these questions: “Now that we have massive databases of materials used by scholars in the humanities and social sciences -- ranging from digitized books, newspapers, and music to transactional data like web searches, sensor data or cell phone records -- what new, computationally-based research methods might we apply?[31])

The big data paradigm is particularly relevant to the study of interactive media. Because of the digitization efforts of libraries, museums, cultural heritage organization and companies, scholars now have access to large volumes of

[31] http://www.diggingintodata.org/.

cultural artifacts in “old media” that can be analyzed using computational methods – millions of newspaper pages, books, maps, historical records, films, and television programs. However, the numbers of digital cultural artifacts available for research makes “big data” of traditional cultural archives rather small in comparison. As an example, compare the following numbers. By 2010, Google Books scanned over 15 million books[32], Europeana provided access to 10 million digital cultural objects[33], and ARTstor.org (a standard online resources for teaching of art history today) hosted over a million digital images of artworks contributed by many museums and collections. But with interactive media, the numbers quickly run into billions: by the same 2010, archive.org hosted 150 billion web pages collected since 1995, while YouTube users were uploading 24 hours of new video every minute. And by early 2012, Facebok was reporting that its users were uploading *7 billion photos every month*.

But this is not all. Software-driven interactive culture does not only dramatically scales up the numbers of media documents being created. It also makes each of these documents potentially infinite in size. In other words, *the “data” of digital experiences not just big – it is infinite*. Let me explain.

For non-interactive media artifacts, “big data” refers to the size - both the size of physical or digital storage required by the media artifacts and the time required by a human observer to “process” these artifacts (read the books, watch the films, etc.) Consequently, while digitization created massive cultural datasets that call for the use of computational methods to analyze them, these data sets are finite. In the case of interactive digital media, “bigness” acquires a new meaning. For example, let's say we want to study a particular video game. As we already saw, such a game does not correspond to any singular text – instead, every time a player engages with the game, this produces a different gameplay. Multiply this

[32] “On the Future of Books.” Google. Accessed October 16, 2010.
[33] http://version1.europeana.edu/”, accessed April 2, 2010.

by many players, and the data expands dramatically. A study of one game now involves analysis of thousands of hours of video that represent complete gameplay sessions by multiple players. (Considering that a typical single player game is designed to take approximately 40 hours to complete, if we capture only 10 complete sessions for each of 10 players, this already would result in 4000 hours of video.) Thus, while the game understood as *text* (or as a *work*) may be quite small (the game engine and media assets adding up to a few GB), the same game understood as *interactive experience* (or as a *generative process*) is potentially *infinite* – since every game play session is unique. This means that we start systematically analyzing interactive media in terms of actual user engagements, we no longer dealing with simply *big data* – instead, we have to embrace *infinite data*.

3. Since "software-driven media" is a particular case of software in general – that is, it corresponds to one or more computer programs written in some standard computer or scripting language which typically use a collection of media assets (texts, 3D models, images, graphics, sounds) and user inpus to generate an interactive experience – it is logical to ask if we can also analyze the code of these programs. However, while the existence of code separate from the actual «interactive text» experiences by the users is certainly one of the key defining features of software culture, the actual study of software programs is not that easy. It generates its own set of theoretical and methodological challenges.

Early software programs such as 1970s video games were relatively short. They form the ideal objects for the code studies approach. However, in the case of any contemporary commercial interactive media project, media application, or an operating system (OS), the program code will simply be too long and complex to allow a meaningful reading - plus you will have to examine all the code libraries it uses.

Consider the following numbers. While Windows NT 3.1 (1993) is estimated to contain 4-5 million source lines of code, Windows XP (2003) already contained 40 million.[34] Praised for its elegance and simplicity in comparison with Windows, MAC OS turns out even bigger, with OS X 10.4 (2006) containing 86 million lines. And what about Adobe Creative Suite? The estimated number of lines for Adobe CS 3 is 80 millions.[35] (Gmail has faired better – it only has 443,000 lines of Javascript.[36])

But the size of software code is not the only problem. While the applications and operating systems running on a single machine already have staggering complexity, the gradual move of application software to the web brings with it a new set of considerations. Let's say we want to analyze ta web service, a web app such as Gmai, or a dynamic web site. (In contrast to static web sites which dominated early years of the web, in dynamic web sites the content of a page can change depending on user' actions. The examples of technologies used today to create dynamic web sites are Javascript, Flash, PHP, Perl, and CGI.[37]) Web services, web apps and dynamic web sites often use *multi-tier software architecture* where a number of separate software modules (for example, a web client, application server, and a database[38]) work together. Especially in the case of large-scale commercial dynamic web site such as amazon.com, what the user experiences as a single web page may involve continuos interactions between dozens or even hundreds of separate software processes. (In 2009 Google started development of a new programming language Go specifically targeted for

[34] http://en.wikipedia.org/wiki/Source_lines_of_code, accessed September 7, 2009.
[35] http://www.craigfergusonimages.com/2007/03/80million-lines-of-code-adobe-cs3/, accessed September 7, 2009.
[36] http://www.infoworld.com/d/developer-world/google-executive-frustrated-java-c-complexity-375, accessed March 29, 2012.
[37] http://en.wikipedia.org/wiki/Dynamic_web_site, accessed September 7, 2009.
[38] http://en.wikipedia.org/wiki/Three-tier_(computing), accessed September 3, 2008.

"big software" – ""large programs written by many developers, growing over time to support networked services in the cloud: in short, server software."[39])

The complexity and distributed architecture of contemporary large-scale software poses a serious challenge to the idea that to study interactive media is to study the code of its software. However, even if a program is relatively short and a critic understands exactly what the program is supposed to do by examining the code, this understanding of the logical structure of the program can't be translated into envisioning the actual user experience. If it could, the process of extensive testing with the actual users which all software and media companies go through before they release new products (anything from a new software application to a new game) would not be required, and the field of Human-Computer Interaction would not need to exist.

The attraction of "reading the code" approach for humanities is that it creates an illusion that we have a static and definite text that we can study – i.e., a program listing. But this is an illusion, and we have to accept the fundamental variability of the actual "software performance." So rather than analyzing the code as an abstract entity, we may instead trace how it is executed, or "performed," in particular user sessions. (Of course, this recording can be combined with other recordings of the sensorial dimensions of interactive experience such as video screen capture of gameplay I discussed earlier.) To use the terms from linguistics, rather than thinking of the code as language, we may want to study it as speech.

[39] Rob Pike (co-designer of Go), quoted in Joe Brockmeier, "Google's Go Programming Language Grows Up: Now What?" http://www.readwriteweb.com/cloud/2012/03/googles-go-programming-languag.php.

(Mark Marino and others working in “critical code studies” paradigm have been promoting more nuanced, theoretically rigorous and rich ideas about what it means to “read the code,” so this critique is only aimed at a naïve version of this idea which I sometimes encounter in humanities. As Marino makes it clear in his original presentation of this paradigm, “code” does not only refer to a program listing but also includes software architecture[40], which is particularly crucial in large-scale software systems.)

What about another conceptual approach - comparing computer code to a music score which gets interpreted during the performance (which suggests that music theory can be used to understand software culture). While it appears promising, is also limited since it can’t address the most important dimension of software-driven media experience – interactivity.

The development of methods for the study of contemporary software, which reduce its complexity to some more abstract representations, which can be then discussed in articles, conferences, and public debates by non-programmers, is certainly a key task for software studies. However, given both the complexity of real life software systems and the fact that, at least at present, only a very small number of media and cultural researchers are trained in software engineering, I don't expect that we can solve this problem any time soon. (While the field of software engineering developed a number of techniques to represent algorithms and programs using high-level representations such as pseudo-code, flowcharts and UML diagrams, understanding these representations does require basic knowledge of programming.)

[40] Mark Marino. “Critical Code Studies.” *Electronic Book Review*, 2006-12-04. http://www.electronicbookreview.com/thread/electropoetics/codology.

4. While the study and interpetation of software code is a challenging project, we can take advantage of the more general property of software-created and software-driven media which I would like to call "self-description." (A different way of expressing this idea is to say that the digital media “texts” contain statements in both the object language and the metalanguage.)

What do I meant by this “self-describing” characteristic? With practically all kinds of digital media, some aspects of media structure, dynamic behavior, the possible range of user interactive experiences, and the semantics are specified in its "software layer.” This software layer can take a number of forms – HTML code of a web page, XML document, After Effects’ “project file” describing the details of a motion graphics project, or the actual programing code (such as the code of a video game). Note that the program “code” is only one of these forms – and in fact, its harder to analyze it because of the reasons I just went through then other types of “software layers,” such markup languages (such as HTML, XHTML or wiki markup), or blog templates.

Certainly, many “old media” formats also have self-describing elements, which ease its automatic analysis. For exampe, the text of the play explictly names each speaking character. An article is divided into sections, and if the names of these sections are explit and clear, they provide a summary of the structure and contents of the article. However, in digital media, such self-describing elements are more systematic, more detailed, and they have to be present. Thus, a HTML code of a web oage explictly specifies different types of media contained in this page. Fhe following examples taken from a page on softwarestudies.com show the code for including an image and a video file:

```
<a href="http://www.flickr.com/photos/culturevis/5109394222/" title="Manga Style
Space by culturevis, on Flickr"><img
src="http://farm2.staticflickr.com/1187/5109394222_2ce37492ae.jpg"
width="500" height="500" alt="Manga Style Space"></a>
```

```
<iframe width="560" height="315"
src="http://www.youtube.com/embed/hd3HMrAIxw4?rel=0" frameborder="0"
allowfullscreen></iframe>
```

Here is another even more dramatic example of self-descibing characteristic of digital media– a piece from a Postcript file which describes a line:

```
0.1 setlinewidth
2 2 newpath moveto
3 3 lineto
3 4 lineto
2 4 lineto
0 setgray
stroke
```

if we know Postscript language, we can understand that this part of the file specifies a few connected strait lines – without having to see the actual rendered image.

What is the reason for this precision and explictness of the codified descriptions of content in digital media files? These descriptions are not written for a human reader. (This, however, does not mean that we have to agree with the the famous opening lines of Friedrich Kittler's 1985 article *There is No Software*: "The bulk of written texts - including this text - do not exist anymore in perceivable time and space but in a computer memory's transistor cells.") Instead, they are read by a software program which renders the "media" (what we see, hear, navigate, and interact with). And computer programs need explicit instructions to do anything – be it displaying images and video in a web page (first example), or simply drawing a vector line (second example).

(This consideration can be used to define digital media as follows: *a media object consist from data and a set of formalized instructions which describe this data and tells media applications how to render them*.)

Focusing on this fundamental self-describing property of all types of digital media opens it up to an exiting "big data" analysis. Rather than thinking about interpeting the particular instances of programming code (which is usually not available for commercial software anyway), we can instead use computers to analyze patterns in the use of self-description elements of large sets of digital media documents. For example, Mathew Fuller suggested that we could map the evolution of web design by analyzing frequencies HTML tags of a large number of the historical snapshots of web pages available on archive.org.[41] I imagine a study of the patterns in architectural design using commands histories maintained by all design software. For example, we can ask architecture students to make available the saved histories of their commands as they work on their project, and then analyze the patterns in the use of these commands. This idea can be also extended to all other types of design and authorship. (Recall here the dream of Semantic Web to add semantic markup to every document on the web – and if we ever get even remotely close, we will have another amazing resource for analyzing cultural patterns.)

This method can be also applied to the study of computer code itself. Rather than analyzing particular code listings manually, we can use big data approach, automatically "reading code" and extracting patterns. For example, Jeremy Douglass proposed that we can follow dissemination of the media techniques and conventions by tracking the use of software libraries across software systems and projects.[42] Obviously, this brilliant idea would only work if we have access to enough program listings. Luckily, because of the open-source

[41] We developed this idea in our (unfunded) proposal to 2009 Digging Into Data competition: Jeremy Douglass, Mathew Fuller, Olga Goriounova, Lev Manovich, *Digging Into Born Digital Data: Methods For The Analysis Of Interactive Media*, 2009.
[42] Jeremy Douglass, "Include Genre", presentation at Softwhere 2008 workshop, University of California, San Diego. http://emerge.softwarestudies.com/files/11_Jeremy_Douglass.mov.

movement, in every part of the software universe we now have widely used equivalents of commercial products which are accessible to such analysis – for example, Firefox browser (%25 market share as of 1/2012[43]), or Blender 3D software (ranked 5th in terms of usage in 2010[44]). Other software type where this “big code” (to use an analogy with “big data”) method would work include scripts written in Perl, Python, MEL, Javascript and other languages, Blogger and Wordpress templates, and any other software which is distributed in human-readable (as opposed to binary) form.

In summary, the self-describing property of all digital media opens another avenue for “big data” cultural analysis. Rather than only analyzing the visible surfaces of interactive artifacts (for example, graphic and interactive designs of a web site, its text and any other media elements), we can also analyze, visualize the structures and patterns in "software layers" of these artifacts.

[43] http://en.wikipedia.org/wiki/Firefox, accessed March 29, 2012.
[44] http://www.blendernation.com/2010/04/18/cg-survey-initial-results-in/, accessed March 29, 2012.