

NeoVictorian Computing, with a Twist

Simon Harper

Human Centred Web Laboratory, School of Computer Science,
University of Manchester, Oxford Road,
Manchester, UK.

simon.harper@manchester.ac.uk — <http://hcw.cs.manchester.ac.uk>

ABSTRACT

Experience in World Wide Web (Web) accessibility has taught us: to think about small bespoke solutions; to tailor interaction and requirements to the user and job at hand; to value high data interoperability; to realise that large enterprise systems, become unmanageable and unable to change at the speed required by both users and technology; and finally, to value heterogeneity. Indeed, with the advent of Workflows, Remote Procedure Calls (RPCs), Representational state transfer (RESTful) services, and Cloud Computing we can also see these viewpoints becoming more common in mainstream thought. Here, we use Bernstein's concept of 'NeoVictorian Computing' as a counterfoil to Andriole's new viewpoint of 21st century software development. We extrapolate from the Web development model into corporate and enterprise systems and propose an architecture of client based heterogeneous applications each tailored to a specific user, and their job, with highly interoperable data, controlled by workflows that are transferred with the data itself. We discount the new client-computing fad, as this really means centrally controlled, sometimes unavailable, old style enterprise systems. We suggest that by moving toward user centred agile systems we follow the conceptual, if not the technological, underpinnings of the Web. In this case we realise that Web developers are in a privileged position to shape and push forward this new kind of software architecture and the 'craft' based approaches which will drive it.

Categories and Subject Descriptors

H.1.2 [Models and Principles]: User/ Machine Systems—*human factors, human information processing*; H.5.4 [Information Interfaces and Presentation]: Hypertext/ Hypermedia—*user issues, navigation*; H.5.3 [Group and Organization Interfaces]: *Web-based interaction*; K.6.3 [Software Management]: *Software development*

General Terms

Human Factors, Design, Theory

Keywords

Accessibility, Human Factors, World Wide Web, Cloud Computing, Agile Systems

1. INTRODUCTION

Accessibility, and particularly Web accessibility, is *the* constrained operating modality. Indeed, we believe that accessibility for disabled users is a natural pre-cursor to wider human factors research and indeed computer science research in general. By understanding disabled-users' interaction we enhance our understanding of all users operating in constrained modalities where the user is handicapped by both environment and technology.

One of the key ideas of the Web, as articulated by Tim Berners-Lee, is exactly this idea of unconstrained interaction suggesting many different kinds of Web pages viewable on many different devices. Indeed this vision so pervades the Web ethos that it also extends into the real world by suggesting that every Web page should be designed so that not just all devices, but all people, can access content regardless of any sensory or cognitive impairments. We assert that this view can be expanded from its focus on human factors to software development in general. This means we need to think about small bespoke solutions; to tailor interaction and requirements to the user and job at hand; to value high data interoperability; to realise that large enterprise systems, become unmanageable and unable to change at the speed required by both users and technology; and finally, to value heterogeneity.

“So what can mainstream software development learn from Web development and from Web accessibility, and what resources created to support Web accessibility can be used by software developers?”

To cross—pollinate we need to rethink our view of accessibility and begin to realise that 'accessibility' is in effect a misnomer; what we really mean is 'completeness' of the data to enable seamless data and data-flow between applications. Indeed, Web accessibility rests on the twin pillars of completeness and standardisation and it is these concepts that are key to how accessibility functions. In this keynote we will rationalise our viewpoint by discussing it in the context of Bernstein's 'NeoVictorian Computing' as a counterfoil to Andriole's new viewpoint of 21st century software development. We see these viewpoints becoming more common in mainstream thought, and examine Cloud Computing along with Workflows, Remote Procedure Calls (RPCs), RESTful services. Finally, we extrapolate from the Web development model into corporate and enterprise systems and propose a architecture of client based heterogeneous applications each tailored to a specific user, and their job, with highly interoperable data, controlled by workflows that are transferred with the data itself; and conclude that Web developers are in a privileged position to shape and push forward this new kind of software architecture and the 'craft' based approaches which will drive it.

2. NEOVICTORIAN COMPUTING & 21ST CENTURY DEVELOPMENT

NeoVictorian Computing is characterised by its heterogeneity, its flexibility, but most of all the way that it is built. Bernstein proposes / predicts a return to craft based approaches to making quality software. Routed in the arts and crafts movement of the Victorian era (hence the name) Bernstein forecasts and evangelises a return to quality, to small software projects created for a tight purpose and with a definite outcome in mind. A move away from large scale enterprise systems to a microeconomic model of software development and customer care. To quote directly Bernstein states:

“By NeoVictorian Computing, I mean systems that are: Built for people; Built by people; Crafted in workshops; Irregular; Inspired. When I say that software is ‘built for people’, I don’t mean some fuzzy notion that the software is intuitive or ‘friendly’ or that it can be sold to millions of consumers. I mean, simply, that it offers some specific people three specific virtues: commodity, firmness, and delight. It helps to get stuff done: not filling out forms or filing pictures or retrieving records, but the endlessly difficult, challenging, everyday stuff of understanding what is going on around us.” [2]

It seems that he has some support for his view if we can correctly interpret Eric Roberts who suggest that in the modern economy: entertainment; education; science; engineering; medicine; and economics, requires innovation and quality in their software products [1]. Indeed, Roberts suggests that for these sectors, software is far from becoming a commodity product, leading us to conclude that Bernstein’s ‘irregular and inspired’ view of software, has traction. Unfortunately, the same cannot be said for Stephen Andriole’s viewpoint [1]. Here we see a view of the software economy routed in the corporate world and characterised my monolithic software projects. Projects so large that companies such as Oracle in the guise of ‘PeopleSoft’ offer an ‘of-the-peg’ solution with a ‘nip-and-tuck’ here and there. The rationale here seems to be that bespoke corporate software is so difficult and prone to failure, and software requirements so transferrable between companies, that there need be only one solution with minor customisation. If this does not fit, then better change your corporate processes. Andriole suggests, that corporations need less and less development, due to these off the shelf customisations and technology outsourcing, and to some degree he may be right, although, his main point is that the need for software development is of so little concern to corporations, we need not train software engineers to code. His view seems to be indirectly at odds with that of Bernstein and Roberts. However, the principles of NeoVictorian Computing seem to also be at odds with enterprise software development ‘... not filling out forms or filing pictures or retrieving records...’, which if interpreted correctly, is exactly what Andriole is suggesting corporations need most.

3. CUMULUS HUMILIS

What do we mean by ‘Cloud Computing’¹? Well maybe it is easier to define what we don’t mean – or at least what we see as being something else. We don’t mean ‘Utility Computing’¹; rented resources, the packaging of computing resources (computation, storage, etc.), as a metered service. Neither, do we mean

¹Thanks to Wikipedia for confirming most of these definitions.

‘Edge Computing’¹; Web based application caches, which provides application processing and load balancing capacity to corporate and other large-scale web servers. Finally, we don’t mean ‘Grid Computing’¹; a network of loosely-coupled computers, acting in concert to perform very large tasks. By cloud computing we mean the ability to store and retrieve data and request computational services from Web infrastructure from any device.

Cumulus humilis is what is commonly referred to as ‘fair weather cumulus’ and is characterised by noticeable vertical development and clearly defined edges. They occur as solitary clouds, or in lines, or clusters and are often puffy or cotton wool like in appearance, and are in general disconnected. The Nimbostratus on the other hand is characterised by a formless layer that is almost uniformly dark grey and covers the sky to a great depth and in one homogeneous connect sheet [3].

At present we can see that the current reality of cloud computing espoused in some form or another by many large corporations, such as Microsoft, Yahoo, Amazon, and Google, is in fact more like Cumulus Humilis Computing than Nimbostratus Computing. Each cloud can contain very similar data types and retrieve information to many of the same machines, however the layer is still not joined. Duplication in functionality occurs through each cloud and instead of focusing on certain strengths the clouds try to replicate each others ability, providing a commensurate but universally mediocre service portfolio. Indeed, the current system seems to be disjoint and shallow as opposed to conjoined and deep.

4. DISCUSSION

Web Service workflow engines, such as ‘Taverna’, may suggest a possible answer to the disconnected nature of cloud computing which mirrors the ‘all in one basket’ view of enterprise systems. Taverna allows users to integrate many different software tools, including web services, to move data through a process using different resources to execute set tasks [4]. These kinds of workflow engines use RPC-style² to orchestrate the flow and call different services, these are mostly SOAP and WSDL style services, with a small handful of RESTful services included. The work flow is orchestrated by a workflow enactor [5], which calls many different kinds of services, most of which are the SOAP+WSDL variety. However, each service and application must be designed with high degree of data interoperability in mind. By understanding the nature of the flow of Web services data, the question is raised,

“Can we move this knowledge coupled with our understanding of the strengths of NeoVictorian Computing onto the desktop?”

We believe that by capturing data, schema, and workflow, in one package we can add value to NeoVictorian computing resources, encourage specialised islands of activity in the cloud computing resources, and remove the need to know in advance what RESTful or RPC resources are required. This approach is supported by researchers in the Computer Human Interaction field who have long expounded the principle of ‘Universal Usability’. A universality which suggests to most designers and engineers that the solutions they come up with must best fit most of the population most of the time. Indeed, many practitioners follow the viewpoint that universal usability means designing to support all users and devices. To create universal packages by including complete data, schema, and

²An Inter-process communication technology that allows a computer program to cause a subroutine or procedure to execute in another address space.¹

workflows involves making no generalisations regarding the application needed to run them. Indeed, it is precisely because previous enterprise solutions made so many assumptions that some applications not fitting these assumptions have been excluded.

Therefore, we see a confluence of thought and technology in this area summarised by the principles underlying RESTful services; In effect, the transmission of domain-specific data over Hypertext Transfer Protocol (HTTP) without an additional messaging layer or session tracking. Something that as Web Developers we are all familiar with and use on a daily basis. These principles maintain that the application state and functionality are divided into resources and that every resource is uniquely addressable using a universal syntax; and that, all resources share a uniform interface for the transfer of state between client and resource, consisting of constrained set of well-defined operations and constrained set of content types, optionally supporting code on demand. So what does this sort of architecture mean for the desktop?

We can now see a future characterised by many NeoVictorian applications each built for a very specific purpose, and therefore very heterogeneous, but with the ability to process specific objects based on the workflow. Application state (achieved by the workflow), functionality (of the application), and interface (specific to the user) are now driven in a RESTful manner in which each object is uniquely identifiable.

In this way we build a push based peer-to-peer system controlled by the enterprise user and the task they need to accomplish. Universal Data Objects can be split and joined, new generated data added and state tracking logs updated by either user controlled applications or automated ones. The process model is therefore humanised as it is the employee controlling the dataflow not the reverse. Indeed, this level of information processing leads to craft based and quality aware approaches when using compute resources, in a similar way to NeoVictorian computing for developers. Additional benefits abound, but the most significant of these is the ability for very small teams to build very tailored human, and user/use, centred applications creating adding and maintaining certain data elements while ignoring other elements not required.

5. CONCLUSIONS

There is a Web characterised by very different operating modalities. It doesn't work properly just yet but it is a lot further along the path to universality than we may imagine. It is called the 'Accessible Web' and it has had its own share of setbacks and successes that we can all learn from. The disabled user serves as a reminder that Web accessibility is a truly individual experience and that by understanding the flexible and personalisation required by disabled users we can understand that at some-point this same flexibility and personalisation will be required, not just by people, but by the software agents we create help, assist, and act for them.

By using these lessons, and techniques we extend the concept of NeoVictorian computing and place the agile, bespoke, craft based software it represents into a network of discrete applications creating a unified enterprise application.

6. ACKNOWLEDGEMENTS

This work is part of the UK EPSRC RIAM Research Project (EP/E002218/1). I would also like to thank the organising committee of WebMedia 2008 for inviting me to give this Keynote.

7. REFERENCES

- [1] S. J. Andriole and E. Roberts. Point/counterpoint technology curriculum for the early 21st century. *Commun. ACM*, 51(7):27–32, 2008.
- [2] M. Bernstein. Neovictorian Computing. Web Page <http://www.markbernstein.org/NeoVictorian.html>, July 2008.
- [3] R. Houze. Cloud dynamics. 1993.
- [4] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucl Acids Research*, 34(2):729–732, 2006.
- [5] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.