Consequences of Software Design Decisions for Low-Income Communities: A Case Study

Fritz Meissner, Edwin Blake
Department of Computer Science
University of Cape Town, Private Bag X3, Cape Town 7701
Tel: +27 21 6502663, Fax: +27 21 6503551
email: {fritz.meissner }@gmail.com; {edwin} @cs.uct.ac.za

This paper discusses how software design decisions can have a positive effect on broad contextual issues which affect Information and Communication Technology for Development (ICT4D) projects. We present five decisions which we made during the design of a web application intended for use by members of low-income communities in Cape Town. Our decisions were based on our knowledge of the context of deployment but increased the effort required of the development team. We use the Bridges.org Real Access / Real Impact criteria to categorise the benefits of our decisions and justify our prioritisation of concerns other than minimising the development effort.

Index Terms—ICT4D, Software Design

I. INTRODUCTION

This paper shows how software design decisions in the initial stages of Information and Communication Technology for Development (ICT4D) projects can have a profound effect on their outcome. We present five choices made during the development of a web application that were decided based on our knowledge of the intended users: an NGO working with low-income communities in Cape Town and the members of those communities. The Bridges.org Real Access / Real Impact criteria [2] are used to categorise the consequences of our decisions.

II. BACKGROUND

A. The Warehouse and Link Programme

We are building a web application in collaboration with an NGO called The Warehouse [5]. The application will be deployed as part of a developmental programme called Link which provides career guidance to people from low-income communities. The Warehouse will capture relevant content and support the site after deployment.

B. Bridges.org Real Access / Real Impact Criteria

The Real Access / Real Impact (RA / RI) Criteria [2] is a framework created to provide a holistic overview of the factors which influence the success of ICT4D interventions. Of the twelve factors, we single out five because they relate to this work: physical access to technology, appropriateness of technology, affordability of technology, human capacity and training, locally relevant content and services.

III. CASE STUDY: LINK WEBSITE

In this section we present the design decisions we made.

A. Limit Unique Implementation Code

We were ethically required to ensure that our work could be understood and modified by other developers with as little cost to the NGO as possible. Our observation was that code with which future developers were already familiar would be better understood than our own, and hence we took a decision to use a framework which had an existing development community. This would reduce the cost of maintenance. Although we lost the flexibility of architecting our own solution specific to our problem domain, this decision positively impacted the RA / RI criteria affordability of technology and technology use.

B. Choose technologies which utilise existing capacity

While the NGO does not have staff with software development skills, the staff is computer literate, managing their operations with email and office software. On the Link programme, existing career guidance data was organised in spreadsheets. We knew that to make best use of the programme's existing human capacity, the addition of new content to the site had to be a task that could be completed by the same people who maintained the spreadsheets, i.e. without any programming knowledge.

The NGO already had a website for informing funders and volunteers of their activity. Content on that website was being maintained by non-technical users. It was built on top of a proprietary Content-Management System (CMS) called ExpressionEngine (EE) [3], which we determined could be used for our application. The possibility of skills transfer from staff and volunteers who worked on the other site (creating content and providing technical support) was attractive due to the reduced need for external support.

The choice of CMS dictated our choice of programming language, as EE was written in PHP, as would be any modifications we made. We were comfortable that even without EE experience, volunteers with a knowledge of PHP would be able to transfer their skills due to the popularity of other similar PHP tools such as Joomla, Wordpress and Drupal (Drupal has also been used by The Warehouse, but the staff and volunteers indicated a preference for EE).

Using EE and PHP increased our development time, as we were not familiar with either before the project began. However, the benefits for long-term maintenance and support will positively impact affordability of technology and technology use as well as human capacity and training. The availability of locally relevant content is also positively affected as it relates to availability of up to date content. This which would have been impeded had we required content administrators to learn a new data entry paradigm or work with a structured language such as HTML.

C. *Use conventional paradigms for custom code*

Two principles applied when we needed to modify standard EE functionality. First, we used EE modules written by the development community where possible, and second, where no modules were available we would package our own modifications as modules instead of modifying the "core" code. Examples of the former principle were tagging articles and rich text editing modules supported by the EE community. An example of the second is a search module which we implemented ourselves that offered support for ranking by relevance not provided by EE. Adopting the standard EE module system gave the code a structure which other developers familiar with EE should be able to understand.

This approach is a reflection of our prioritisation of the time of developers who would need to be paid to modify our work. By making their work easier (at the expense of the time we spent learning how the EE module system worked), we avoided a negative impact on RA/RI criterion three, affordability of technology.

D. *Avoid complex data relationships*

Early data modelling efforts were very relational. Job adverts and study courses were related by field, for instance. Using this information, the site could display jobs available for potential graduates if a user was viewing a course, or courses necessary to work in a field when viewing a job. The data model grew more complicated as we considered the need for other relationships such as sector and career. Upon reflection we realised that requiring the data capturers to correctly differentiate between relationships would lead to frequent mistakes.

Our solution was to provide the ability to tag entries. This would allow the site to suggest to users that an unread article might be related to one they were currently reading based on tags in common, but without having to identify which sort of relationship applied. New relationships could emerge and not every relationship need apply between every possible pair of entries. This freed both data capturer and end users from the time consuming task of distinguishing between relationships. Although some time spent on our relational data model design was discarded, we knew that this decreased our need for new human capacity and training.

E. *Reduce infrastructure barrier*

The web technologies used to build the site were chosen based on existing infrastructure. We consider two areas: hosting platform and end user computer access.

The choice of CMS dictated the use of PHP, but we did consider a deviation from the language, in order to use Apache Solr, an Java full-text search server [1]. However, the NGO's existing web host refused to support Java. When we considered that we would be adding to the workload of the volunteer who provided technical support by requiring him to manage an account at a new host, we decided against the change. Fortunately we were able to compromise between prioitising existing capacity and using the library we wanted by using a PHP port of the Java code which underpins Solr.

Household access to computers is unusual in the communities with whom we are working [6], hence reliance on telecentre infrastructure such as Smartcape [4]. A consequence of not having control of the computers on which our website is viewed is that our site has to work on low-end computers. From a bandwidth perspective, this meant not offering a graphic intensive or AJAX heavy site. From a client machine perspective, the site will avoid Javascript or other client-side technology where possible.

By setting aside our own technology preferences because of infrastructure considerations we positively affected four RA / RI criteria: physical access to technology, appropriateness of technology, affordability of technology and finally human capacity and training.

IV. Conclusion

We have presented five choices made during development of the Link web application: to use existing code as much as possible, to use technologies that require as little re-training as possible, to code in the standard paradigm of our chosen CMS, to avoid complex data relationships and to leverage existing infrastructure. Our decisions had a negative impact on the initial development effort due to time spent learning new technologies and discarding early data model design work. However, we have shown that these decisions positively impacted five RA / RI criteria: physical access to technology, appropriateness of technology, affordability of technology, human capacity and training, locally relevant content and services. In so doing we have avoided many ICT4D pitfalls of which the RA / RI criteria warn.

V. References

[1] Apache Software Foundation: Apache Solr, 2007. Retrieved June 14, 2011 from Apache Software Foundation: http://lucene.apache.org/solr/

[2] Bridges.org: Real Access / Real Impact Criteria | bridges.org, 2005. Retrieved September 3, 2010 from Bridges.org: http://www.bridges.org/Real_Access

[3] EllisLabs: ExpressionEngine – Publish Your Universe. Retrieved June 14, 2011 from ExpressionEngine: http://www.expressionengine.com

[4] Smartcape: About – Smartcape, 2010. Retrieved June 14, 2011 from Smartcape: http://www.smartcape.org.za/about.html

[5] The Warehouse: Welcome to the warehouse. Retrieved June 14, 2011 from The Warehouse: http://warehouse.org.za

[6] Walton, M. 2010. Mobile literacies & South African teens: Leisure reading, writing, and MXit chatting for teens in Langa and Gugulethu. Research report prepared for the Shuttleworth Foundation m4Lit project. http://m4lit.files.wordpress.com/2010/03/m4lit_mobile_literacies_mwalton_20101.pdf

Fritz Meissner received his Computer Science Honours degree in 2006 from UCT before working as a software developer for three years. In 2010 he began working towards his M.Sc., also at UCT. His research interests include software design and the use of technologies in the developing world.