Anatomy of an Early Social Networking Site

Alan Dix[1,2], Russell Beale[3], Nadeem Shabir[1], Justin Leavesley[1]

[1] Talis,
43 Temple Row
Birmingham, B2 5LS, UK

[2] Infolab21
Lancaster University
Lancaster, LA1 4AW,UK

[3] School of Computer Science
University of Birmingham
Birmingham, B15 2TT

{ alan.dix, nadeem.shabir, justin.leavesley }@talis.com *r.beale@cs.bham.ac.uk*

http://www.hcibook.com/alan/papers/hci2011-vfridge/

Social networking software is ubiquitous, from Facebook to Flickr, defining the internet for many users. However, this is a recent phenomenon. Is the timing due to socio-technical determinism, inspiration of individuals, or sheer chance? While much has been written about recent successful social networking sites, this paper takes a different approach and examines vfridge, a social networking application developed 10 years ago, well before the current explosion, which, despite a vision that now seems prescient, was unsuccessful. The reasons for failure are partly about timing and market conditions, but also yield valuable lessons for future innovative applications.

social network, adoption, web2.0, web architecture

1. INTRODUCTION

Social networking software is now ubiquitous from Facebook to Flickr, defining the internet for many users as the web did for computing 10 years earlier. However, this is a recent phenomenon, arising largely over the last 3-5 years, for example, Facebook grew from a million users at the end of 2004 to over 500 million today, see Figure 1.). Is this timing due to techno-social determinism, the inspiration of individuals, or sheer chance?

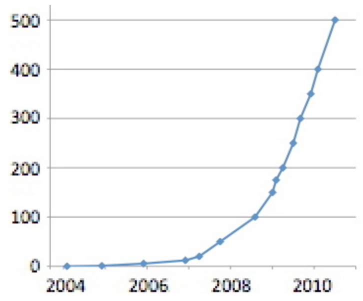


***Figure 1:** Facebook Growth 2004–2010 – millions of users (data from Facebook, 2011)*

Much has been written about recent successful social networking sites, examining many social and experiential aspects (e.g. Hart, 2008; Ferebee, 2009, Kirman, 2010). However, the analysis is largely of the successful systems attempting, to determine post-facto, the critical features of success in terms of user experience or other factors. This paper takes a different approach and examines vfridge, a social networking application developed 10 years ago, well before the current explosion. vfridge was ultimately unsuccessful, despite a vision that now seems prescient and a design that most who saw it found captivating. The reasons for failure are partly about timing and market conditions, but also yield lessons for future innovative applications. While management literature frequently studies business success and failure; this paper takes a broader view of market, social and technical effects.

Note, the term 'social networking' was initially used for sites focused on profiles, dating, or other forms of connecting (e.g. Linkedin for business). Over time social *sharing* (Twitter, Facebook wall, Flickr) has become more significant. vfridge was focused on the latter, sharing in a social group, rather than profiles or connecting.

vfridge attempted to create the 'message centre' of fridge magnets on the web, drawing partly on the analysis of Norman (1992) and partly on earlier 2D communication systems. Although there were short presentations at the time of its production (Dix, 1999, 2000), and occasional mentions in the literature, this is the first in-depth analysis.

The next section describes the commercial and technical background to the development of vfridge and the "*web sharer*" vision, which drove it. The web sharer concept now seems obvious in the light of web2.0 and the social networking explosion, but at the time was far from the expected evolution of the web. Section 3 describes vfridge itself both as a user experience and some implementation details. Section 4 then analyses various the technical, commercial and market-readiness issues, which contributed to its ultimate failure.

2. BACKGROUND AND VISION

2.1 Commercial background and vfridge history

Given massive changes in the web it is important to put the development of vfridge in a commercial and technical context. This started with aQtive a web start-up funded by venture capital company 3i just before the UK dot.com explosion in 1999. aQtive produced an intelligent internet agent onCue (Dix, 2000b), which was distributed on over half-a-million magazine cover disks and attracted substantial press and TV interest. However, it was a casualty of the 2000 dot.com crash after the infamous lastminute.com IPO (Wray, 2010; Goldfarb, 2006).

Whilst 1999 had been driven by a "Get Big Fast" business strategy (Goldfarb, 2006), the UK post-crash environment was very conservative with virtually no investment in hi-tech and a focus on showing early profit for existing companies. In retrospect, current valuations of companies that survived the crash (including lastminute.com itself) suggest, this was an over-reaction. Certainly business strategies today look strikingly similar to those pre-crash, but, of course, both technology and market conditions have changed in the interim.

In this context vfridge was envisaged to be a self-marketing product and one to aid the marketing of other aQtive products. As such, it was explicitly designed to be social, to engender regular usage and active sharing by its users, and was created with a clear goal to be viral and spread rapidly. This was reflected in the underlying architecture, which was created to cope with very many vfridges, and in the interface designs that were focussed on ease of use and approachability for a very wide section of the population. This is important, in that for some applications their popularity and spread is very welcome, but unexpected and undesigned; in this case, it was part of the plan.

Despite the harsh funding environment following the crash, aQtive was successful in obtaining just over £200K seed funding for a spin-out company vfridge.com, based around the vfridge web application. vfridge.com subsequently ran out of cash, was mothballed in 2002 and the company dissolved formally in 2004. However, the application continued to run for existing users albeit getting steadily slower over the years with newer versions of Java. In 2010 the application was rebuilt in facsimile (in PHP), and is available for public viewing and use (vfridge.com), albeit with the original, but now retro, 2000-web look and feel.

1999 –	first implementation – stand-alone Java app with bespoke server later first web version using applet
2000 –	dot.com crash but seed funding for vfridge.com
2002 –	vfridge mothballed
2010 –	‘facsimile’ version reproduced in PHP!

Figure 2: *Timeline of vfridge*

2.2 The web sharer vision

During the height of the dot.com era in 1999, the dominant view was there would eventually be some sort of shake out in the market; not the hi-tech market crash that actually happened, but more of a food chain where smaller companies would be bought by larger companies leading, as in other parts of the media industry, to a small number of major players dominating the market (e.g., AOL, Yahoo – Google was only recently out of the lab). In particular the homespun personal web page was expected to fade and instead a slick TV-style broadcast medium emerge.

Set against this backdrop, aQtive was formulating a counterview captured in a whitepaper entitled the "*web sharer vision*" (Dix, 1999b), which suggested that a potential (but not inevitable) path for the web was towards far more open future:

> “*The web/Internet is not just a medium for publishing, but a potential shared place.*
> *....*
> *Everyone may be a web sharer — not a publisher of formal public 'content', but personal or semi-private sharing of informal 'bits and pieces' with family, friends, local community and virtual communities ...*”

Today, post web2.0, this is passe: one of O'Reilly's web2.0 maxims is "Users Add Value" (O'Reilly, 2005) and end-user content is integral not only to social media sites such as YouTube, but also in reviews and lists in Amazon and other eCommerce sites. However, at the time this was counter to the popular wisdom and in hindsight seems prescient.

This vision statement led to the search for products that would be suitable for and help to bring about this web sharer future. It was in response to this that vfridge was designed.

2.3 Theoretical foundations

The design of vfridge was also built upon a more theoretical view of the diffusion of web products. One aspect of this was *market ecology*, which involved the tracing of networks of influences between different stakeholders or market groups. For example, if a teacher is using an educational application then potentially many children in her class might start to use a variant of the product targeted at home use; parents may then be influenced and so on (see Figure 3). Particularly important are the feedback loops, for example, where a parent influenced by one teacher then goes on to mention the application to a different teacher at a parents' evening. Of course, this is now familiar as viral marketing, but, at the time, this was only just beginning to emerge as a concept.

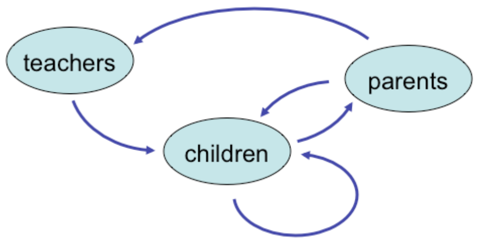


Figure 3: *Market ecology – tracing the network of influences between different stakeholders*

Market ecology was itself based on earlier work in CSCW, in particular Grudin's (1988) seminal "Why CSCW applications fail", which was first to highlight issues of *critical mass*, and Dix's (1997) development of this to analyse the potential for CSCW applications on the web. Also very influential was Cockburn's doctoral work (Cockburn, 1993a, 1993b). Cockburn's work was significant because it was the first time that Grudin's analysis was used *constructively* to guide the design of groupware systems.

In parallel to the study of CSCW systems the concepts of *network effects* was being developed within economics research (Liebowitz, 1998; Economides, 1996). Traditional economics assumed that the value of a good to one person was independent of its value to others. However, many digital products do not work this way. If one person is using Microsoft Word, then the value of Word increases for that person's colleagues if they wish to collaborate. These positive network effects can be found in non-digital products or standards, for example, the choice of gauge for railway tracks, but are much more pronounced in the digital domain. For example, Brynjolfsson and Kemerer (1996) applied this form of analysis to the early spreadsheet market to account for the growth of Lotus 1-2-3. Crucially it was the establishment of new platforms (DOS, and later Windows) that enabled 1-2-3 to emerge in the face of an existing competitor, VisiCalc, and later Excel to supplant it

At a similar time to the development of market ecology, Stallman (2000) at Sloan was applying *system dynamics* models to address business issues. The mathematical models of business dynamics are virtually the same as those of market ecology, both modelling the flows between groups as multiple linked differential equations. However, whilst both network effects and business dynamics were used *descriptively* to understand economic phenomena, in contrast, market ecology was used *prescriptively* to guide design, just as Cockburn did with Grudin's CSCW analysis.

3. THE VFRIDGE APPLICATION

3.1 Appearance: free layout and local structuring

As noted vfridge attempted to capture some of informality of the fridge door, but in a web setting. Registered users could have one or more 'fridges', each fridge was essentially a 2D canvas on which short text notes or images could be posted. Figure 4 shows a screen shot of the full interface (do recall styling c.2000!). As in many current applications such as mySpace (but interestingly not Facebook), users could choose colour scheme, background image or even create custom skins to integrate vfridge more closely into other sites.

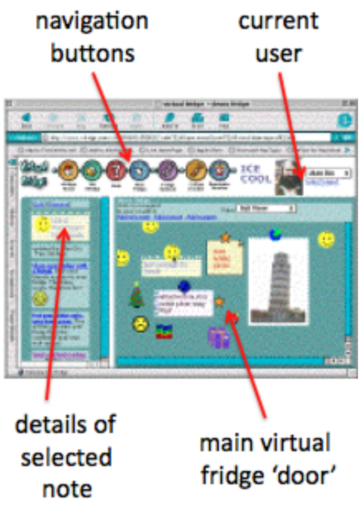


Figure 4: *vfridge screenshot*

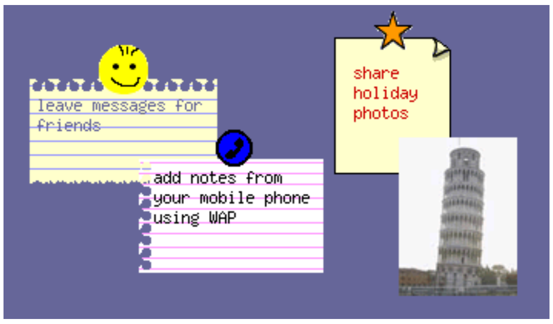

leave messages for friends
add notes from your mobile phone using WAP
share holiday photos

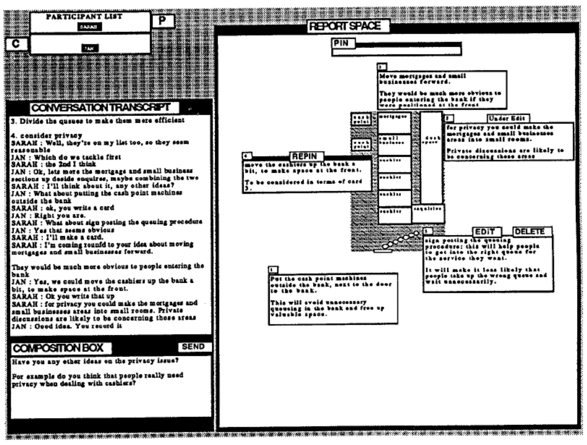
Figure 5: *Close up of notes on a fridge*

There are various navigation and sidebar elements, but the main area is the large frame on the bottom right. This is the area where user notes and photos are displayed. Figure 5. shows a close up of a few notes on a vfridge. Notes could have different kinds of 'notepaper', which resized to the text and the user could also post pictures. Each note has a small icon at the top, its 'magnet'. As well as a decorative element, this was where the user could grab the note to move it around. Using this, notes could be repositioned, allowing users to create ad hoc groupings. Fridges could be private or public, and, crucially for adoption, could be shared.

As already noted, the choice of a fridge door was not arbitrary, but partly due to the inspiration of Don Norman's analysis of fridge doors as 'message centre' (Norman, 1992). Later, more rigorous ethnographies of domestic life have corroborated the importance of these informal message centres in homes, from kitchen cork boards to piles on the hall table (Crabtree, 2004).

Although this was crucial to the choice of the fridge door metaphor, more central to the core design issues was the work on 2D layout interfaces. While 2D folder layout was a feature for file icons in the Star and Macintosh interfaces (albeit only partially implemented on Windows), the Xerox Whiteboards interfaces, built on top of the Cedar programming environment, made this a far more central part of the interface (Donahue, 1986). In particular 2D virtual 'whiteboards' could be shared so that they became a means of collaboration.

Whiteboards was also one of the inspirations for the experimental Conferencer system (McCarthy, 1990). In common with many CSCW systems of the time, Conferencer had a synchronous chat area (left-hand side, Figure 6), but in addition had a 2D area (right-hand side, Figure 6), which users could use to post shared messages.



3. Divide the queues to make them more efficient

4. consider privacy
SARAH : Well, they're on my list too, so they seem reasonable
JAN : Which do we tackle first
SARAH : the 2nd I think
JAN : Ok, lets more the mortgage and small business sections up deside enquires, maybe combining the two
SARAH : I'll think about it, any other ideas?
JAN : What about putting the cash point machines outside the bank
SARAH : ok, you write a card
JAN : Right you are.
SARAH : What about sign posting the queuing procedure
JAN : Yes that seems obvious
SARAH : I'll make a card.
SARAH : I'm coming roundd to your idea about moving mortgages and small businesses forward.

They would be much more obvious to people entering the bank
JAN : Yes, we could move the cashiers up the bank a bit, to make space at the front.
SARAH : Ok you write that up
SARAH : for privacy you could make the mortgages and small businesses areas into small rooms. Private discussions are likely to be concerning those areas
JAN : Good idea. You record it

Have you any other ideas on the privacy issue?

For example do you think that people really need privacy when dealing with cashiers?

2
Move mortgages and small businesses forward.

They would be much more obvious to people entering the bank if they were positioned at the front

5
for privacy you could make the mortgages and small businesses areas into small rooms.

Private discussions are likely to be concerning those areas

4
move the cashiers up the bank a bit, to make space at the front.

To be considered in terms of card 3.

3
sign posting the queuing procedure: this will help people to get into the right queue for the service they want.

It will make it less likely that people take up the wrong queue and wait unnecessarily.

1
Put the cash point machines outside the bank, next to the door to the bank.

This will avoid unnecessary queueing in the bank and free up valuable space.

Figure 6: *Conferencer*

Both Whiteboards and Conferencer highlighted the importance of *local* vs. *global structuring*. Designers determine many aspects of the way interfaces are organised; they are globally structured, the same for everyone. However, the 2D layout allows users to choose their own arrangements (local structuring), assigning certain areas for decisions made, another for new ideas, etc.; that is *appropriating* the interface. Similar observations have been made in the study of user desktops, where users assign different screen areas for different purposes (Katifori, 2008). In all of these systems, it is usually the case that users have not pre-decided, "this area is for X, that are for Y", but instead these distinctions typically emerge over time and are only later articulated.

3.2 In every situation

vfridge was targeted at sharing between family and friends. Corporate systems can often rely on a common technical infrastructure in terms of hardware, operating system and web browser – it is still common to see extranets that are Windows/IE only. However, when designing for open groups it is crucial that the application is available in some form to everyone irrespective of platform.

The main view of vfridge used DOM manipulation and JavaScript. This was still very new technology, highly browser specific, supported on only the newest versions of browsers, and often buggy. vfridge was therefore deliberately constructed with a number of 'fall back' options.

The 'List View' was most important, a simple multicolumn layout showing notes in reverse order of posting (newest first), rather like Facebook wall today. The edged 'notepaper' was preserved in this view and the magnets also displayed above information about who posted the note (see Figures 7 and 8). Of course this view did not show the 2D layout, so a miniature image of the 2D fridge was also displayed (generated server-side) and the user could also chose to view and interact with a full size static image of the fridge, although the interaction was somewhat more clumsy than the DOM-based view requiring frequent page transactions.

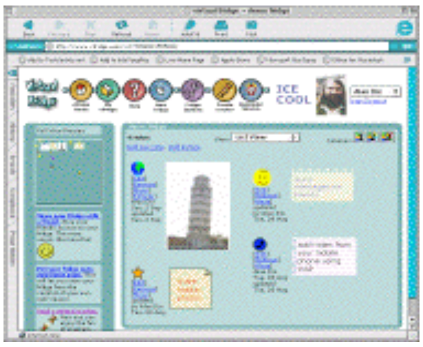
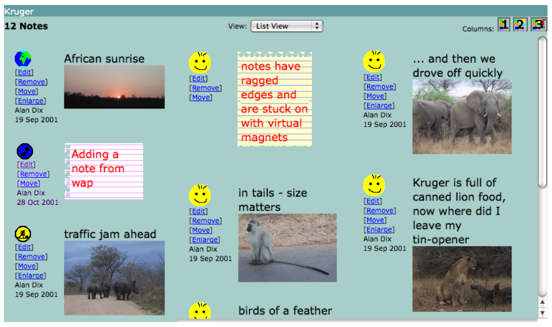


Figure 7: *'List View' – multi-column layout*



12 Notes

African sunrise
Alan Dix
19 Sep 2001

Adding a note from wap
Alan Dix
28 Oct 2001

traffic jam ahead
Alan Dix
19 Sep 2001

notes have ragged edges and are stuck on with virtual magnets

in tails - size matters
Alan Dix
19 Sep 2001

birds of a feather

... and then we drove off quickly
Alan Dix
19 Sep 2001

Kruger is full of canned lion food, now where did I leave my tin-opener
Alan Dix
19 Sep 2001

Figure 8: *List view close-up (three columns)*

As well as catering for low-end browsers, vfridge was one of the early applications to work across mobile devices. In figure 5 one of the notes has a phone magnet; this note had been submitted by a phone. It was also possible to view a fridge using a WAP phone (the early mobile web standard). At first the graphic 2D layout of the fridge seemed at odds with the small and at that point very low resolution of a phone. However, the small notes suggested by the 2D layout were ideal for adding or viewing by phone, prefiguring micro-blogging.

Technology changes over time, but the basic lesson does not. Now-a-days DOM manipulation is standardised, but browser differences remain (especially IE). Furthermore with many different devices (iPhone, Android, internet enabled TV and games consoles) collaborative applications have to consider how to work at some level with all. It is no good sharing photos using a iPhone-only app if you are the only person in your family who owns one.

3.3 Architecture and implementation issues

Many web frameworks are designed around variants of MVC (Krasner, 1988; Tate, 2006) and there has been considerable work on both using web technology in desktop user interfaces (e.g. XUL (Feldt, 2007), UsiXML (Vanderdonckt, 2005)), and in developing model-based user-interface development (e.g. Paternò, 2009). However, the architectural design of web-based user interfaces is still a difficult issue. One reason is that the transactional and 'demand-driven' nature of server-side web computation is very different to the rapid feedback and data-driven nature of traditional GUI development. Perhaps more problematic the division of the user interface functionality between client-side and server-side computation often means that all levels of the Seeheim model (Pfaff , 1985) get split between the two (see Figure 9).

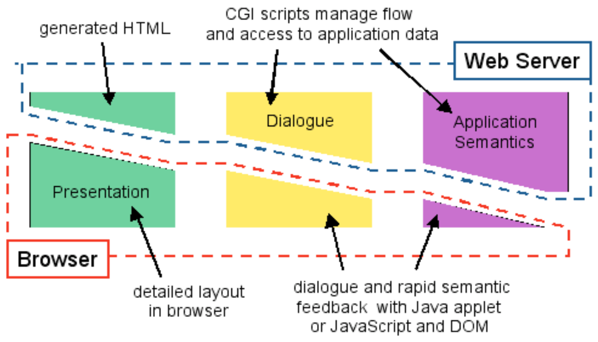

generated HTML
CGI scripts manage flow and access to application data
Web Server
Dialogue
Application Semantics
Presentation
Browser
detailed layout in browser
dialogue and rapid semantic feedback with Java applet or JavaScript and DOM

Figure 9: *Seeheim for web applications (from Dix, 2011)*

vfridge used a number of techniques to address these issues, most crucially the use of frames. In the screen shots in Figures 4 and 7 the three main areas: top navigation, sidebar and main fridge 'door' area are all separate frames. In addition, the top controls in the main fridge area (where the user chooses the view type and number of columns in List View) is yet another frame. Whilst general web design advice at the time was (and still is) against the use of frames (Nielsen, 1996), in this circumstance the use of frames was very important. First they reduced page reloads, important both for performance and to maintain display inertia. Furthermore, the frames meant that the overall URL was of the fridge itself, not some transaction with additional superfluous URL parameters, as was, and still is, common. This made bookmarks and emailing the URL clearer and less likely to cause accidental additional transactions. Occasionally it would have been useful to modify the URL whilst not reloading the whole frameset, for example, when the view changed so that the URL could specify the view as well as fridge.

The need for frames is now all but obviated by the use of AJAX. However, control and modification of the URL is, if anything, more of an issue on AJAX-based interfaces, which perform virtually all interactions without a page reload. The URI hash is being used increasingly as a way of allowing JavaScript to modify the bookmark-able URL without reloading the page; examples can be seen in many sites (e.g. Twitter) and support is now provided in libraries such as jQuery (2011). This is stretching the original intended use of the hash as a "fragment identifier" and causing a rethink of its use in an evolving web (Raman, 2011).

In simple interfaces each component of the interface is independent and updates to one do not affect any others. These are easy to implement whether through AJAX calls to RESTful services (Fielding, 2002) and client-side rendering or server-side re-rendering of the entire page. However, in many applications there is some level of interdependence between components; for example, in vfridge the sidebar would include extra information about the currently selected note in the main frame. The knowledge of these inter-dependencies is often embedded in the back-end business logic, for example, if the oil price changes this may affect profit-and-loss projections.

In a desktop interface this kind of change is often managed through multiple *views* on a shared MVC *model*, which pushes changes to the views using an *observer* or *publish-subscribe* pattern (Gamma, 1994). This is not so simple when logic is split between client and server in a web application and often applications are written so that the client 'knows' the potential impact of various API calls to the server and conservatively requests updates of all potentially affected components.

The vfridge implementation addressed this problem by using a variant of publish-subscribe distributed between client and server. When a component loaded into a frame it registered a list of change events on which it depended; for example 'selectedNote' when the currently selected note changed. These events could be triggered by client-side JavaScript or the server-side script could declare a number of events to be triggered as new frame contents were rendered. Any dependents of a triggered event were then refreshed. This meant that components could be designed with limited knowledge of each other. While some details of this scheme were based on the use of frames, the same broad architecture would be possible where components perform AJAX updates.

While this refreshing of frames was suitable for 'big' interactions such as changing the view or selecting a note, they were not suitable for highly interactive note dragging on the fridge 'door' as they operate in the 150–200ms timescale of hand-eye coordination rather than the 1-5 second timescale of consciously considered interaction. These updates were performed locally and propagated to the server asynchronously (using unrendered 'images' with URLs encoding update information). While AJAX would now be used, the same interaction issues apply; web user interface architectures/frameworks need to deal effectively with rapid interaction rather than leave this to ad hoc solutions.

Another issue, still a problem in current web-based user interfaces, is the efficient generation of dynamically created pages. vfridge used a custom template engine; as with other template solutions, such as Smarty, this enabled a better separation of dialogue logic and detailed presentation. However, templates are criticised on efficiency grounds, with many systems instead using 'code' as template (e.g. JSP or PHP). The vfridge template system required data not as a fully elaborated data structure (such as XML). Instead the data structure included 'promises' for certain parts, objects that would yield concrete data on demand; a form of lazy computation (Hudak, 1989). This meant that dialogue/semantic level processes could create dictionaries of data for the template engine, but only those parts that were actually used in the template were evaluated. For example, the 'data' for a fridge included the list of notes as a promise; if only the fridge title was used in the template, the notes field would never be accessed and the database not queried for the notes themselves.

4. BARRIERS AND FAILURES

vfridge was built based on a vision of the web that has since been vindicated, and used a metaphor that users found compelling, but it did not succeed.

When searching for explanations one should never underestimate the impact of sheer chance. When a system has strong positive feedback effects it is subject to potential dramatic differences in outcome due to small and apparently insignificant changes in conditions: this is precisely the domain of catastrophe theory and chaos theory (Jordan, 2007). Under such circumstances it is not necessarily the best systems that succeed, but just those that happen to be first to reach a critical size, or more precisely a critical density, in some group of people. There is a natural human tendency to look for 'rational' explanations or reasons, one source of superstition. In the face of random phenomena, explanations can often be arbitrary, whether analysing the success of Flickr, Facebook or YouTube, or, as in this case, failure. So, as we seek explanations, we keep in mind that we do not expect a clear cause–effect, but more factors that made success or failure more likely.

Some of these factors are external, the socio-technical environment in which vfridge developed – the right idea at the wrong time (just as boyd (2004) concluded regarding the demise of SixDegrees). Indeed new concepts are notoriously difficult to establish even when the situation is ripe. Other factors are internal, things that could have been done differently and might have made a difference. The former are interesting from an historical perspective, the latter more informative for future design of innovative products.

4.1 Technical barriers

There were technical barriers 10 years ago, that have all but vanished now. Indeed, while the original (Java servlet) vfridge took 2 years to develop, the facsimile (PHP) re-implementation in 2010 took just 10 days of effort. In fact, the move to PHP started long before; the server-side image generation proved far too slow as the Java toolkit depended on the presence of a video card, so part of vfridge was implemented in PHP using the native GD library. As found in many large-scale systems that use 'scripting' languages, the efficiency and scalability of a system depends on far more than the raw speed of the underlying language.

To be fair the first implementation of vfridge was equally fast to develop, taking less than a week's effort to create a desktop Java version that synchronised peer-to-peer or via a server to enable asynchronous sharing, even when the client was disconnected for periods. Surprisingly, commercial synchronisation technology is still crude. Mac OS Address Book and iCal synchronisation with the iPhone will create two copies of each contact and event if the user elects to synchronise both directly through docking, and indirectly though MobileMe. Synchronisation research was mature even 10 years ago, and vfridge synchronised correctly even when using multiple paths.

However, the difficulty of deploying Java applications at the time and the need for lightweight access for ease of sharing led to the web-based alternative, which over time became the main focus. The first approach was to take the core of the desktop application and put it into an applet.

However, those who have used Java extensively know that Java's "write once run anywhere" promise, while better today, is far from universally borne out. In particular, Java browser support was very patchy, with, at best, very early JVM versions, and substantial memory management problems. While this has improved and Flash replaced most Applet-based applications, recent proliferation of mobile devices again means that full browser support cannot be guaranteed. Of particular importance and difficulty is getting good debugging output from client-side applications once deployed. This is an ongoing issue, especially when early development is performed on a high-end platform.

The continuing, and in the end insurmountable, difficulties with the applet solution led to a completely web-based approach using DOM, which had became usable (just) at the time. The back-end was implemented in Java servlets initially using a bespoke data management layer (the original synchronisation server), and later a MySQL back-end. The Java-SQL interface using JDBC was always a cause of problems, due in part to the tendency for Java frameworks to be published without reference implementations. This continues to be an issue with many standards (e.g. the early days of OpenID and OAuth), so this is certainly a lesson for today. No matter how clear the specification, with any infrastructure technology the path to rapid and standards-observing uptake is to have simple and readily available code examples.

Now these may not seem like HCI issues, however, then as now, there is nothing so destructive of user experience than buggy or slow systems.

In fact, it was the Java–SQL interactions that eventually killed the original implementation. Java has been good at maintaining backwards compatibility so that the original complied code continued to run on newer versions of the Java VM. In 10 years the raw machine speeds increased 50–100 fold and the JVM increasingly optimised for speed. However, despite this, the Java code actually performed less and less well with slower page reloads and often total failure. The reason for this appears to be slight differences in the network behaviour of MySQL.

Similar network-related problems are still a major problem in many user interfaces. For example, Dreamweaver and Word often seem to crash or experience long 'freezes' when network connections are intermittent or slow. The solution to this is partly rigorous stress testing away from the perfect conditions of the usability lab. However, there is also a need for tools that are able to trace those parts of code that are likely to be subject to delays due to network communications. This is important both to identify parts of code likely to be subject to race conditions and potential corruption or crashes, but also to determine when a user action has the potential for a long delay and hence the possibility to modify the interaction to ameliorate the problem.

The biggest problem faced was cross-browser compatibility. Now-a-days greater adherence to standards and JavaScript toolkits that hide differences make this far easier. However, this does not mean the problems have disappeared, numerous apparently minor differences remain, and yet, as the Java–SQL experience reminds us, small differences can have unexpected consequences. Furthermore, development environments, especially .NET can make it easy to use vendor specific features without realising.

4.2 Market barriers

These technical barriers to development interacted with the market situation. At the time 'fast' connections were only 56 Kbits (c.f broadband speeds of 2-10 Mbits today). This meant that a 1 Mbyte download was about as large as practical. Actually delivering code onto a user's machine was a major barrier. This was one of the reasons for originally moving towards a web-based solution.

At first this seems like a problem that is long past; certainly the sort of download sizes that we considered problematic 10 years ago would be negligible today. However, when developing on a site with high-capacity internal and external network, it is easy to neglect download times. In the UK whilst the government promotes Internet services, and BT are announcing "BT Infinity" with 40Mb download speed, still a significant proportion of rural areas do not have access to broadband at all and, looking worldwide, this situation is more severe. Most software today has regular internet delivered updates. Rather than delivering small deltas, the tendency is to download a large proportion of the code and assets (on MacOS it is not uncommon to have several 200Mb updates for different system software packages). Presumably those at the end of long phone lines with no or reduced broadband do not update their software with consequent risks to stability and security.

In 1999 the use of the web was patchy at best. Furthermore, even if people had access to a PC and knew how to use it, for many new users there was a tendency to only turn on the PC intermittently to pick up mail or surf the web. aQtive, the forerunner company to vfridge, was founded on a vision that looked forward to an always on, always connected society, of course this is still far from universal even in the UK (try getting a 3G mobile signal outside a major population centre), and certainly not globally. As middle-class professionals, whether academics or developers, it is easy to assume this is a past problem. However, only 70% of the UK population have access to the Internet at home and 60% access it daily, nearly 20% have never accessed the internet at all (ONS, 2010). The majority of the 20% who have never accessed the internet are elderly or poor. In the UK it is illegal to discriminate on grounds of gender, race or sexuality and yet eGovernment, preferential rates for internet purchases, and the movement of whole areas of commerce online are continuing to isolate and alienate significant segments of society.

For collaboration, the lack of common access was, and is, a major barrier. If your family or friends do not have a suitable connectivity, then sharing becomes impossible or at least a very secondary activity; boyd and Ellison (2008) note that this was a major issue for SixDegrees.com, another early social networking site. This is easier now than 10 years ago, but it is interesting that Facebook began its path to success only 2 years after vfridge was mothballed. Crucially Facebook first focused on students at elite US universities, a clique with above average connectivity, who could act as a bridgehead to broader adoption as identified in previous CSCW theory (Grudin, 1988, Dix, 1997). While there are opportunities for products that mine the potential of the 'bottom of the pyramid' (Prahalad, 2004), it is often easier to target the rich.

Even where households were adequately connected, the underlying ecology of home Internet use had a profound social impact. Accessing the Internet was not the pick-up-and-go approach of today that (when they work) wireless, quiet and often discrete devices support; instead, connecting was a slow process punctuated by protracted musical negotiation as modems sought a speed to share data, and, having done so, hog the only telephone line into the house, curtailing any other form of telephone communication. Furthermore, most dial-in connections charged by the second, so that home internet connectivity was an additional non-trivial expense. This focussed people's minds and actions on what they primarily went online to do, and then to disconnect as quickly as possible

Together this meant that going online was not the social process that it is today; it was the antithesis. Thus, people tended to have a specific need to connect in order to initiate the process, and this lack of easy access counted against vfridge. The nature of Internet connectivity goes further however; permanent, non-blocking and effectively free connectivity allows systems such as Facebook to exist on the peripheries of our awareness, allowing us to dip in and out of them in snatched minutes between other tasks, and this easy awareness of social activity and our simple, gap-filling engagement with it demands little of us: this is very different, mentally, from having to actively check for something in a costly manner.

4.3 Commercial pressures

The post-crash environment also created commercial pressures. Rather than "Get Big Fast" the emphasis was, quite sensibly, on traditional ROI (return on investment). While the web market was a desirable long-term strategy, there was also a need for short-term revenue, if not profit. This lead to a divided focus, as vfridge had potential applications in domestic, educational, and business settings, all were pursued – contrast this with Facebook's very narrow focus. The search for a good-enough-for-all design can easily yield to one that is best for none and furthermore when cash or time is limited a narrow focus is likely to be more successful even if the longer-term vision is broad.

This spreading of market potentially clashed with the graphic design of vfridge. Modelled on notepads, post-it's and fridge magnets, it gave a reasonable facsimile of the real fridge door, and so was immediately recognisable, understandable and appealing for informal 'family and friends' use. But for some it looked somewhat childlike. This is reminiscent of the difference between mySpace and the cleaner more business-like styling of not only LinkedIn, but also Facebook.

4.4 Losing sight of vision

The above factors in various ways were contributory to a weaker product and hence potential causes of failure. However, there was one entirely internal factor, which could potentially have made the difference. The vfridge team lost site of the big vision.

This paper started with the web sharer vision and showed how it led to a specific product. This vision itself was posited as a *potential* future for the web and has turned out to be *the* future (at least for the current time). Furthermore, potential users were excited by the vfridge concept and loved the ability to drag notes around the 2D fridge. However, they also loved the 'list view', which simply arranged notes in columns. This was a little surprising as to some extent it was not so different from the many bulletin boards and forums available at the time, or even email itself, just a chronological list of things. However, there were crucial differences.

The styling of the notes, the 'pretty' note paper, which to a hardened pre-experience-focused HCI eye looked like pure surface candy, was engaging and allowed users to see the metaphor of the fridge even when the surface appearance was less fridge-like. Furthermore the notes were like, and yet not like, the postings in forums, or the messages in email. Like chat, the note metaphor and layout encouraged short pithy utterances, not long missives. However, unlike chat, the asynchronous nature suggested semi-permanent yet lightweight sharing of the apparently insignificant rather than conversation. That is the list view was equally consonant with the larger web sharer vision and shared crucial features with subsequent success stories (Facebook wall and twitter stream).

So, given the technical problems with the 2D fridge and the fact that users still loved this second-choice interface, why wasn't the more complex interface dropped, at least temporarily, in favour of the simpler list view? The time saved on the often unsuccessful development of the 2D interface could have been used to improve other crucial sharing features (e.g. better group support and notifications) or in business development. Whether this would have made sufficient difference is impossible to tell, but it would have increased the likelihood of breaking through the crucial self-sustaining growth point ahead of the later social networking applications. While there were extensive technical barriers, looking forward a few years, Facebook could have been produced using 1999 technology and a very simply list-like look and feel. The simplified vfridge was not so different.

The answer to the "why?" is simply a loss of vision, or at least a loss of the big vision (the web sharer) because of the focus on a particular way to approach the vision (the 2D fridge). This is certainly not simply a problem of the millennium, but one that can overtake any project, especially one creating innovative products.

Note here that the issue is not about whether a company has a vision for the future. The vision on its own doesn't tell you the best place to start, and what to do next amongst the myriad paths, nor whether there is a workable path at all towards the realising the vision. The problem here is that there was a defined route towards the vision (the 2D fridge), just that when there were technical barriers and an achievable alternative (the list), vfridge did not step back, and take a wider view. To be fair few companies would have fared better. First because it is a natural human trait to problem solve at the finest level first. Second because to diagnose the problem and solution requires a clear view of both technology and business strategy and the way they interact. Getting it right would have been difficult, but still not impossible.

5. DISCUSSION

We have seen how vfridge in many ways did 'the right things' based on a vision, which, whether through foresight or luck, turned out to be a very accurate view of coming social networking. As we considered reasons for its ultimate failure, we found examples of technical problems that will be familiar to many involved in web development, issues of browser incompatibility, now re-appearing with device diversity, the nature of the URL, the surprising speed of applications built on 'slow' languages, and inadequate interface debugging/ development tools, for dealing with intermittent networks and timing issues. Some technical solutions in vfridge may have value today: an event-based UI architecture cutting across client and server, mechanisms for rapid local interaction, lazy template data and effective synchronisation.

Examining boyd and Ellison's (2008) history of social networking, there appears to be a move from public dating or contact-based sites, through more intimate sites supporting offline contacts, with sharing late in the cycle. Maybe vfridge was simply too early in this evolution, although this feels too easy an excuse.

Basalla's (1988) analysis of technological evolution suggests that technological novelty should be seen in the light of *continuity* with the past and the *selection* in the present. This is certainly the case with vfridge, with roots in Whiteboards and Conferencer, and where even the web sharer vision grew organically from previous CSCW theory. Crucially, the selection due to the state of the Internet market at the time was a major barrier to uptake. However, this and indeed all analysis of success factors suggests a level of external inevitability. If there is a single lesson to be drawn it is not this, but the importance in innovate technology of holding to the big vision.

6. ACKNOWLEDGEMNTS

Thanks to the anonymous reviewer who pointed us to interesting relevant literature, and to past friends at aQtive who shared with us those exciting years.

7. REFERENCES

Basalla, G. (1988). *The evolution of technology*. Cambridge University Press.

boyd, d. and Ellison, N. (2008). Social network sites: Definition, history, and scholarship. *Jnl of Computer-Mediated Comm*, 13(1), 210–230.

boyd, d. (2004). Friendster and Publicly Articulated Social Networks. In *CHI 2004*, pp. 1279–1282.

Brynjolfsson, E. and Kemerer, C. (1996) Network Externalities in Microcomputer Software: An Econometric Analysis of the Spreadsheet Market. *Management Science*, 42(12), 1627–1647.

Cockburn, A. (1993a) *Groupware design: principles, prototypes, and systems. PhD thesis*, University of Stirling, Scotland.

Cockburn, A. and Thimbleby, H. (1993b) Reducing user effort in collaboration support. In *Proc. of IUI '93*, pp. 215-218.

Crabtree, A. and Rodden, T. (2004) Domestic routines and design for the home, *Computer*

Supported Cooperative Work: The Journal of Collaborative Computing, 13(2), 191-220.
Dix, A. (1997) Challenges for Cooperative Work on the Web: An analytical approach. *Computer-Supported Cooperative Work*, 6, 135-156.
Dix, A. (1999) Annotations a UI Perspective (panel presentation), *COOPIS'99*, Edinburgh Sept. 1999. http://www.hcibook.com/alan/papers/coopIS99/
Dix, A., 1999b. The Web Sharer Vision. eBulletin, aQtive Ltd., November 1999. http://www.hiraeth.com/alan/ebulletin/websharer/
Dix, A. (2000) Designing a virtual fridge (poster). *Computers and Fun 3*, York, 13th December 2000. appears in *Interfaces*, 46, 10-11
Dix, A. Beale, R. and Wood, A. (2000b). Architectures to make Simple Visualisations using Simple Systems. *Proc. of AVI2000*, ACM, pp51–60.
Dix, A. and Shabir, N. (2011). Chapter 3: Human-Computer Interaction and Web Design. In Proctor, R. and Vu, K. (eds), *Handbook of Human Factors in Web Design, second edition*. Lawrence Erlbaum.
Donahue, J. and Widom, J. (1986) Whiteboards: a graphical database tool. *ACM Trans. Inf. Syst.* 4(1), 24-41.
Economides, N. (1996) The Economics of Networks. *Intnl. Jnl of Industrial Org.*. Oct. 1996
Facebook (2011). Company Timeline. http://www.facebook.com/press/info.php?timeline
Feldt, K. (2007). Programming Firefox: Building Rich Internet Applications with XUL. O'Reilly Media. pp. 76–77. ISBN 0596102437.
Fielding, R. and Taylor. R. (2002). Principled design of the modern Web architecture. ACM Trans. Internet Technol. 2(2), 115-150.
Ferebee, S. and Davis, J. (2009). Factors that persuade continued use of Facebook among new members. In *Proc.* of *Persuasive '09*. ACM, Art. 35.
Goldfarb, B., Kirsch, D. and Miller, D. (2007). Was There Too Little Entry During the Dot Com Era? *Journal of Financial Economics*, 86(1),100-144.
Grudin, J. (1988) Why CSCW applications fail: problems in the design and evaluation of organizational interfaces. in *Proc. of CSCW '88,* ACM, pp. 85-93.
Hart, J., Ridley, J., Taher, F., Sas, C. and Dix, A. (2008). Exploring the facebook experience: a new approach to usability. In *Proc. of NordiCHI '08*. ACM, pp.471-474.
Hudak, P. (1989). Conception, evolution, and application of functional programming languages. *ACM Comput. Surv.* 21(3), 359-411.
Jordan, D. and Smith, P. (2007). *Nonlinear Ordinary Differential Equations: An Introduction for Scientists and Engineers*. Oxford University Press
jQuery (2011) *jQuery: The Write Less, Do More, JavaScript Library*. http://jquery.com/ (22 Jan 2011)
Katifori, A., Lepouras, G., Dix, A. and Kamaruddin, A. (2008). Evaluating the Significance of the Desktop Area in Everyday Computer Use. in *First Intnl. Conf. on Adv. in Computer-Human Interaction, ACHI 2008*. IEEE, pp. 31-38.
Kirman, B., Lawson, S., Linehan, C., Martino, F., Gamberini, L. and Gaggioli, A. (2010). Improving social game engagement on Facebook through enhanced socio-contextual information. In *Proc. of CHI '10*. ACM, pp.1753-1756.
Krasner, G. and Pope, S. (1988). A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *J. Object Oriented Program.* 1(3), 26-49.
Liebowitz, S. and Margolis, S. (1998) Network Externalities (Effects). *The New Palgraves Dictionary of Economics and the Law*, MacMillan, 1998.
McCarthy, J. and Miles, V. (1990) Elaborating communication channels in conferencer. In *Proc. of the IFIP WG 8.4 conference on Multi-user interfaces and applications*, Gibbs, S. and Verrijn-Stuart, A. (Eds.). Elsevier, pp. 181-193.
Nielsen, J. (1996). *Why Frames Suck (most of the time)*. http://www.useit.com/alertbox/9612.html
Norman, D. (1992) *Turn Signals Are the Facial Expressions of Automobiles*, Cambridge, MA: Perseus Publishing.
ONS (2010). *Internet Access*. Office for National Statistics. http://www.statistics.gov.uk/cci/nugget.asp?id=8
O'Reilly, T. (2005) *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*. dated 09/30/2005 http://oreilly.com/pub/a/web2/archive/what-is-web-20.html
Paternò F. Santoro C. and Spano L. (2009). A model-based approach to address the design of Web 2.0 applications based on Web services. In: *IxD&A - Interaction Design & Architecture(s), "Design for the Future Experience"*, 5/6, 17–22
Pfaff, P., & ten Hagen, P. (Eds.) (1985). Seeheim Workshop on User Interface Management Systems. Berlin: Springer-Verlag.
Prahalad, C. (2004), *The Fortune at the Bottom of the Pyramid: Eradicating Poverty Through Profits*, Wharton School Publishing.
Raman, T. and Malhotra, A. (eds) (2011). *Repurposing the Hash Sign for the New Web*. W3C Working Draft: Putative TAG Finding. http://www.w3.org/2001/tag/2011/01/HashInURI-20110115
Sterman, J. (2000) *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Irwin/McGraw-Hill.
Tate, B. and Hibbs, C. (2006). *Ruby on Rails: Up and Running*. O'Reilly Media
Vanderdonckt, J. (2005). A MDA-compliant environment for developing user interfaces of information systems. In: *Proc. of CAiSE'05*. pp. 13–17. Springer-Verlag
Wray, R. (2010). Ten years after the crash, the dotcom boom can finally come of age. *The Observer*, Sunday 14 March 2010. http://www.guardian.co.uk/business/2010/mar/14/technology-dotcom-crash-2000