Self-Sustaining Open Source Software Development

— or —

Programmer or Prostitute?

Peter Gutmann

University of Auckland

Preamble

This is not a popularity contest

- Open source diatribe
  - "We will bury you"
- GPL rant
  - Blah blah copyleft blah blah a'om bomb blah blah
- Linux advocacy/Microsoft bashing

Tell it like it is, even if it's not what people want to hear

OSS: What's Hot and What's Not

Hot	Not
Networking	User interface
Filesystems	Office applications
Device drivers	Documentation

Microsoft rules the desktop because they can pay people to do the boring stuff

- OSS, by its very nature, can't do this
- Most users only experience the boring stuff

OSS: What's Hot and What's Not (ctd)

OSS is successful in servers because tweaking networking and filesystem code is cool

- OSS developers will cherry-pick the interesting parts of the work

Everyone wants to be a code god

- Less interest in being a documentation god or i18n god or online help god

Least likely to ever appear as a competitive OSS project: MYOB, Quicken, *generic accounts-receivable packages*

- Totally unsexy, must be modified yearly, customized for every different jurisdiction, requires legal expertise to write the rule base, negotiation with governments to support features like on-line return submission, …

Sustaining an OSS Project

Sponsored by your employer

- Usually only works for large companies (IBM, AT&T)
- Needs to match your company's needs
- Can't compete with their commercial offerings

Charge for support

- Only works in a few niche areas
- Most companies don't see this as a value-add except for firefighting
  - Databases: "We have our own DBAs to handle this"
  - Security: "It's easy, our programmers can add that before we ship"

Sustaining an OSS Project

Dot-coms

Spare-time project

- Sourceforge: Graveyard of abandonware/gedanken projects

Self-sustaining

- Usually starts out as a hobby/spare-time project that gets out of control
- Need to make a living or development work will stop

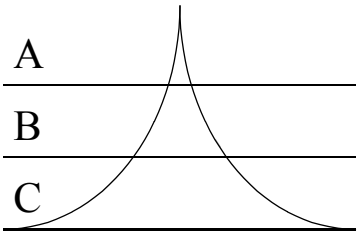
OSS Project Taxonomy

A = well-known, important enough to have outside support

- Linux, Apache, GPG

or popular enough to work on a hobby basis

- KDE, Gnome, audio/video codecs



B = self-sustaining

- MySQL, Berkeley DB, OpenBSD

C = abandonware, splintered into factions, gedanken projects

- See sourceforge, freshmeat

Self-Sustaining Open Source Software

What is self-sustaining OSS?

- Open-source and free (if possible)
- Commercially licensable (to support further development)

Many OSS projects don't translate easily into a sustainable business model

The Sleepycat License

Originated with Berkeley DB

Dual license, GPL or commercial

- If you're open-source, we're open-source
- If you're commercial, we're commercial

Users pay to avoid the GPL

- If you don't want to play by OSS rules, you can opt out… for a price

Enough users opt out that the "OSS sin tax" funds further development

The Sleepycat License (ctd)

Only works in some cases

- Product is a library-type product
  - GPL's viral nature would infect software that links with it
  - Doesn't work for standalone utilities, applications, etc
- One entity owns the IP rights
  - If many developers own the rights, there's no-one to approach for a single license

What if a Commercial User Violates the Lic.?

No commercial entity can really afford to do this

- Companies need to exercise due diligence; even obvious OSS users will check to make sure that it's OK
- Anyone who really wants to violate the license will do so, for example by buying a cheap demo copy and shipping product based on it

In some regions, piracy is the norm

- Handle tech support on a low priority
- Philosophical approach
  - Piracy is better than obscurity
  - As long as they're pirating ours and not the competition's
  - As long as they pirate Microsoft's stuff as well

Sleepycat Variations

Many minor variants exist, e.g Helix DNA (RealNetworks)

- Intent was to spread RealNetworks' media to cellphones, PDAs, …
  - Beat Windows Media Player to market domination outside the desktop
- Developers make project proposals
- RealNetworks assigns paid employees to act as project leads
  - Ensure interoperability, consistent look and feel, avoid duplication of effort
  - 250 of 900 employees are assigned to Helix projects

Avoids the freshmeat forking/abandonware and inconsistency/incompatibility problems

What Scares Companies off OSS

Companies need to be able to perform risk management

- Plan ahead for the entire product life

Software vendors open-source software of low commercial value that they want to abandon

- "Let's put it out there and see what happens"
- Is anyone really going to leap in to maintain 1M+ legacy LOC?

Uncertainty is your biggest enemy

- How do we know that you'll still be here next week?
- What do we do when we need help?
- Where is this going?
  - Forks, geek featureitis, abandonware

What Scares Companies off OSS (ctd)

Perception of OSS projects being unresponsive to user requests (as opposed to developer preferences)

- Example: strlcpy/strlcat in glibc
  - Maintainer doesn't like them and refuses to add them
  - Hundreds of packages all have to roll their own versions
- Example: TCP offload engine (TOE) support in Linux
  - Network stack maintainers don't like it
  - Linux networking has performance problems at GB speeds

This lack of response to customer demand would never work in the commercial world

What Scares Companies off OSS (ctd)

Some companies *expect* to pay for their software

- Need to prove money out/goods in to their accountants/auditors
  - Provides them with a fixed up-front cost breakdown
- Under what basis do we sue if something goes wrong?
  - No-one ever sues over software, they just need to know they could if they wanted to

Planning: "You have the source code, what's the problem?"

- How much do we budget for support per year?
- Most commercial vendors can provide a detailed breakdown of support options

What Scares Companies off OSS (ctd)

Support: "You have the source code, what's the problem?"

- Altair tech support, circa 1976: "You have the schematic, what's the problem?"
  - Tongue-in-cheek way to say "There is *no* tech support"
- Robert L.Glass' 60/60 rule
  - 60% of software dollars are spent on maintenance
  - 60% of that is enhancements
- Need to allocate one (or more) programmers to learning the code
  - Even at only ¼ full-time, this comes to $x0,000 per year
  - Luxury support contracts are only $x,000 per year
- It can take months before your programmer is up to speed

What Scares Companies off OSS (ctd)

Futureproofing: "You have the source code, what's the problem?"

- You're entirely dependant on the whim of the maintainer(s)
- What if they lose interest?
  - 99% of sourceforge/freshmeat is abandonware
- Commercial vendors are contractually bound to some form of support
  - You pay a premium for Compaq gear, but can get support for a 10-year-old product

What Scares Companies off OSS (ctd)

Masses of incompatible OSS EULAs frustrate lawyers

Example: SugarCRM (OSS CRM) has a ~700 screen page EULA

- Incorporates licenses from PHP, Apache, various GNU-ish licenses, MySQL, mod_dav, Python, libiconv, …
  - It's actually "only" 60+ print pages when reformatted and printed on A4 paper
- Further licenses are incorporated by reference
- All of these pieces have different terms
  - All are vaguely incompatible

At least Microsoft only give you a single EULA with a single set of terms to sort out

What Scares Companies off OSS (ctd)

Red Hat form 10-Q provides an insightful overview of what makes commercial operators nervous about OSS

- If open source programmers, most of whom we do not employ, do not continue to develop and enhance open source technologies, we may be unable to develop new products or adequately enhance our existing products
- We rely to a significant degree on several largely informal communities of open source software programmers and a relatively small group of software engineers, many of whom are not employed by us […] We cannot predict whether further developments and enhancements to these technologies would be available from reliable alternative sources

continues…

What Scares Companies off OSS (ctd)

…continued

- If third-party software application providers do not continue to make their applications compatible with our [product], our software will cease to be competitive
- We may be unable to predict the future of open source technology development, which could reduce the market appeal of our products and damage our reputation
- We do not exercise control over many aspects of the development of open source technology. Different groups of open source programmers compete with one another to develop new technology. If we adopt new technology and incorporate it into our products and competing technology becomes more widely used or accepted, the market appeal of our products may be reduced.

continues…

What Scares Companies off OSS (ctd)

…continued

Because of characteristics of open source software, there are few technology barriers to entry in the open source market by new competitors and it may be relatively easy for new competitors with greater resources then we have to enter our markets and compete with us

One of the characteristics of open source software is that anyone can modify the existing software or develop new software that competes with existing open source software. Such competition can develop without the degree of overhead and lead time required by traditional proprietary software companies.

continues…

What Scares Companies off OSS (ctd)

…continued

We could be prevented from selling or developing our software if the GNU General Public License and similar licenses under which our products are developed and licensed are not enforceable

We are vulnerable to claims that our products infringe third-party intellectual property rights because our products are comprised of software components, many of which are developed by numerous independent parties

The risk of infringement claims is exacerbated by the fact that much of the code in our products is developed by numerous independent parties over whom we exercise no supervision or control

continues…

What Scares Companies off OSS (ctd)

…continued

Despite testing by ourselves and our customers, errors have been and may continue to be found in our products after commencement of commercial shipments. This risk is exacerbated by the fact that much of the code in our products is developed by independent parties over whom we exercise no supervision or control.

This is a must-read for OSS → commercial developers, since it outlines exactly the concerns that commercial users have with OSS software

Perception Management

Commercial companies expect to see themselves when they look in the mirror

Commercial	OSS
Large staff	1-2 people
Businesslike attitude	Laid-back
Suit and tie	Jeans and t-shirt
9 to 5	Noon to 4am
Manuals	FAQ + mailing list
Web page with Flash animation	Sourceforge

In order to be taken seriously, the OSS effort must be able to look like a commercial operation

The truth well told…

Tell us about your company

Can we rely on you?

- "We've been in business for nearly 30 years, have 1,200 employees, offices in 5 countries…"
- We've been around for a while, we're not a fly-by-night, and we have an office in your time zone

" " " " ", specifically how many full-time programmers you have working on the product right now

- "Uhh, 2 I think"
- Just because there are a lot of people here doesn't mean that they're all working on the code

Staff Size

Most companies have very inefficient software development

- PGP 2.x: 3 people, part-time
- GPG: 1 person, full-time
- NAI PGP: ~200 people, full-time

The small dedicated team has the edge over the large team

- It'll take a 200-person team a month just to agree on the font to use in the requirements document
- See "Software Craftsmanship: The New Imperative" by Pete McBreen

Staff Size (ctd)

Well-known OSS project

- "We've tried to do an equivalent commercial project, but the size of the team required made it economically infeasible. How big is your development team? How do you manage to fund it?"
  "He's sitting over there. We feed him pizza"

Staff Size (ctd)

In a Byte magazine interview, Steve Jobs multiplied Bill Atkinson's work time by six to make it credible to readers

How many man-years did it take to write QuickDraw?
— Byte magazine (to Bill Atkinson)

I worked on it on and off for four years
— Bill Atkinson

We invested twenty-four man-years in QuickDraw
— Steve Jobs (to Byte interviewer)

Staff Size (ctd)

Companies are suspicious of work by one or two people, since it would take them 30-50 people to do the same thing

- OSS software is often perceived as a quick hack either because
  - It *is* a quick hack
  - Companies could never do it with the same effort
- Affiliate yourself with a larger company
  - Incubators provided this service during the dot-com era
- Use code metrics to show that your code would take a 20-person team 10 years to write (even if it was written by one guy in two years)

Documentation

OSS docs: Howto, FAQ, and mailing list

- "Why should we pay for your software when there's an OSS equivalent?"
  "We have a manual and tech support. They have a manpage and a mailing list"
- Almost all OSS projects are characterised by *appalling* documentation

Documentation (ctd)

A proper manual

- *Not* a manpage. Manpages are cheat sheets for quick reference, not documentation
- (Professionally-done) online docs (PDF) are OK
  - Real companies do this too
  - Printed docs cost too much and are quickly out of date

Hypertext help

- May be just the manual in another form
  - Can generate PDF and HTML from the same source
- See MSDN for an example of good hypertext help

Documentation (ctd)

Manual quality is inversely proportional to tech support volume

- Fielding queries increases overhead → tune documentation to minimise user queries
- Alternative is to hire dedicated support staff or divert developers to doing tech support

If your product is aimed at programmers, "code samples" are actually "templates"

- Write "samples" that can be cut & pasted into applications

Bad Tech Support

Mailing list, even if it has a 1-hour turnaround time

- "We're not interested in our customers"

Totally nondeterministic

- May get a reply in 10 minutes… or never
- No-one is responsible for ensuring that the question gets answered

Bad Tech Support (ctd)

I want to know where I can send money to have this fixed, because it's ridiculous. If it's coders you need, I can solicit readers to help out. Just please, someone tell me what to do. Let me know where I can put a bounty on this bug, who I can email or call to hassle about this, or anything else I can do. I don't know much about the Mozilla bug-fixing process, but this seems ridiculous. Has this seriously been an unfixed issue for over six years now?

Maybe just because I had to waste time reading your useless comment and the useless replies that followed, I should delay checking in whatever fix I eventually get reviewed. Did you really think the TWO HUNDRED NINETY comments before yours hadn't made it obvious to us that people want this?

— Support request and response on bugzilla
(this issue still hasn't been fixed)

Good Tech Support

Tech support address/phone number, even if it has a 3-day turnaround time

- "Your enquiry is important to us and is being processed on a priority basis" in as close to real-time as possible

Having a business relationship means that the user can disclose confidential information (source code, client data) to help track down problems

Feature Requests

OSS: You've got the source code, make the changes and send us the diffs

- Translation: "Go away and find another product that does what you want"
- "Come back with some code" → "FOAD"

Commercial: Sure, we can do that…

- For a price
- If you have a support contract

This may still be a "Go away", but the user gets to make the decision

Feature Requests (ctd)

Process must be deterministic

- OSS: Someone may eventually look at it when they get time
- Commercial: We'll have it delivered 6 weeks after we receive the signed contract

Contract Development

Everyone loves you for writing OSS, no-one wants to pay for it

> Software gets written for you by some troupe of crack-whore programmers simply because, well, they love this communist utopia s**t
>
> — Matt Robinson, lazycat.org

Companies will promise you all manner of contract development work… in 6-12 months time

- Few ever deliver

Contract Development (ctd)

Customers are left in somewhat of a bind

- How do they maintain the code after the contracted work is complete?
- Commercial organisations will provide ongoing support, not just one-shot coding jobs

Don't rely on contract work for anything other than windfall income

- If you line up developers, no-one will want work done
- If you line up work, you can't find developers to do it

Contract Development (ctd)

Some customers will want to keep their newly-acquired code proprietary

- They'll only ever do this once
- They've now forked the code and will need to maintain it for the rest of eternity

Don't sweat this one too much

Contract Development (ctd)

Best way to handle contact development is via paid feature development

- "We'll rearrange our roadmap for you if you pay for it"
  - Similar to the way commercial operations prioritise work allocated to their developer pools
- Minimises damage if things don't work out
  - The work would have been done anyway, just at another time

Example: Borland, Corel, MusicMatch paid CodeWeavers to ensure that their software works under Wine

Example: SAP paid MySQL AB for SAP/R3 support

Usability Testing

OSS developers only write software that they want to use

Software by geeks for geeks

- Q: Why are there dozens of (expensive) commercial Windows network/packet monitors available when you have Ethereal?
- A: People willingly pay good money just to avoid having to use that GUI
  - (Although it's gotten a tiny bit better in its Wireshark incarnation)

Five GUI developers → five UI designs, we'll make it skinnable and let the user decide

Usability Testing (ctd)

Companies like Microsoft have usability labs for this

- OSS has the world's largest usability lab, you just need to know how to take advantage of it
- Test results tell you what to focus on… and what not to focus on

However, paid developers will provide feedback

- Toss-it-over-the-wall users usually don't provide feedback
- They just complain in blogs, or go somewhere else

Using the OSS Usability Lab

Frequently-encountered problems/frequently-asked questions

- Usually a sign of usability problems, either in the code or the documentation
- (This doesn't work for all cases: Security failures are silent, so your software could be completely broken without users noticing)

Interpret (valid) user queries as a sign that the UI or manual need fixing

- This is a strength of OSS. Companies like Microsoft pay $*x*00,000s for this

Concentrate development on these hot-spots

Using the OSS Usability Lab (ctd)

Code instrumentation

- Track how/how often features are being used
- Can be problematic due to privacy concerns

Disable little-used features in betas

- It's a beta, that's what it's there for
- If no-one complains, #ifdef/remove them

Software adapts over time to do what users *really* want, not what developers think that users want

Licensing

Plagiarise^H^H^HTake inspiration from standard boilerplate

Get a lawyer to check it/make it look good

- More perception management: A professionally-prepared legal document implies a robust company

Licensing (ctd)

A properly-done license may save your bacon later

- "Share and enjoy" sounds good, but isn't very sound legally
- Standard clauses cover use in life-support systems, real-time control environments, allocation of liability, etc

"Your product is crashing and costing us $50K/day in lost revenue"

- You don't want to hear that without a formal, signed license agreement in place

Exclusive Licenses

"We will port it to *obscure-platform* in exchange for an exclusive license"

"We will support it in *obscure-country* in exchange for an exclusive distributorship

- Good: You get support in an environment you'd never get to otherwise
- Bad: You can *never* do anything in that environment again

Alternative wording: We would like you to sign a contract restricting you from ever working in this environment again

Exclusive Licenses (ctd)

Solution: Gentleman's agreement to forward all enquiries for the environment to them

- "Virtual Exclusive License"
- Made binding by the fact that both sides benefit from it (no incentive to cheat)

Legal Issues: Continuity

Licensing is handled by a separate legal entity

- They are held legally liable, not you

What if you get run over by a truck?

- Set up a trust/foundation to own the software

Terms of the trust: Open-source, no exclusive licenses, etc

Poison-pill clauses to discourage hostile takeovers

- The software becomes public-domain if development on it falls below a certain level
- Sunset clause

Legal Issues: Protecting your Work

What if you can't pay your credit card bill? Get divorced? Get sued?

Create two legal entities

- One owns the IP and licenses exclusively to the other
  - Structured to be judgment-proof
  - Choose a geographic location of benefit to you
- One handles the business side
  - Structured as a business
- Agreement dissolves on the insolvency of one party
- Potential target is the business entity, not the IP holder

See a lawyer for more on this

- THIS IS NOT LEGAL ADVICE

Legal Issues: IP Protection

Patents are to impress investors, not to protect IP

- Investors want IP and barriers to entry
- In practice no-one cares about your idea until you succeed with it

Since patents are just investor theatre, save money by filing them yourself

- Provisional patent
- "Patent it Yourself", Nolo Books

If really necessary, you can do it properly once you have the money

Legal Issues: IP Protection (ctd)

Always at least trademark your product

- Fairly easy, just fill in a form and pay a filing fee
- If someone wants to fork your code, they can't do it under the original name

PhDs are also investor theatre

- In academia, what your peers think matters
- In business, what your customers think matters
  - Your customers are rarely your peers
- PhDs tend to be really bad at running businesses

Keep a PhD around for investors and the press

Dealing with Lawyers

Your goal is to explain what you want done, not convert the unbelievers

- You are a slightly nutty character who has something they could be getting rich from, but chooses to give it away
- You want a safety net, not a means of extracting every last dollar
- No, no need to patent it, but thanks for asking

Provide clear instructions on what you want done

- Take pre-prepared written notes with you
- Cuts down on time ($$$), removes ambiguities

Resist the urge to preach

Management

OSS startups are developer-centric

- Most stay that way until they get VC funding
- VCs impose (for better or worse) management and sales staff

However, post dot-com VCs are nervous about funding companies getting less than a few $M/year

- Difficult to get to without a good sales staff

Going for VC funding takes a lot of the fun away

Management (ctd)

Find someone else to run the business

- *Never* start a software business just because you like writing code
- You can get away with bad code provided that you know your market, treat your customers right, and make good business decisions
- Even the best code won't save a badly-run business

If you're under 30-ish, try and find an experienced mentor

- Friend or acquaintance who can look over your shoulder and provide advice
- Local government bodies/business associations often provide mentoring services for small businesses

Marketing

Hand it off to the marketroids

- You are not even remotely qualified to do this
- (Micromanagement/interference is a problem with small businesses in general)

Never look at what they're doing to your software, unless you're

- Under heavy sedation
- Drunk

Sales

Use the right tool for the job

- The right tool for repairing a TV is a TV repairman
- The right tool for selling/licensing software is a software salesperson

They know about

- Pricing schemes
- Volume discounts
- Resellers
- Managing web storefronts
- Clients' spending authorisation limits
- Handling credit card payments/wire transfers
- Dealing with unpaid purchase orders

Sales (ctd)

Geeks are used to dealing with other geeks

- Both sides have a high level of technical knowledge
- Can extrapolate from technical details to see that your product is cool

Geeks like to sit back and let their work speak for itself

- How many people here have parents who understand what you do in your work?

Sales (ctd)

Non-geek customers can’t make this leap

- Too much detail is lost in translation for the customer to understand the product

Let the salespeople handle this

- (Or sell exclusively to geeks)

Sales (ctd)

Easiest approach is to align your product with someone who has an existing sales channel

- Incubator/University business unit
- Current/former employer
- Existing user with a large investment in the product

Advantage: Everything necessary is already in place

Disadvantage: Your product takes second place to existing ones

Sales (ctd)

Sales through public vs. privately-held companies

- Privately-held: Convince the owner that it's a good idea
- Publicly-held: Convince the board of directors and company lawyers

Legal concerns can kill you

- "Can the company be sued by acting as sales agents for this?"
  - In general, yes, even with an intermediate company in the chain
  - The target will be the company with the largest assets

Sales (ctd)

Typical risk analysis from corporate lawyers

- "Who's your largest customer?"

  "Shell Oil"

  "So if your software fails, we're going to be sued by Shell?"

May require that you buy liability insurance

- This is in general infeasible

Selling Out

A brief word on VCs…

- Make lots of money or retain control of your baby, choose any one
- When you IPO, you lose almost all of your negotiating power
- You won't be able to sell your stock for years
- Company will be driven by the need to grow profits, not to continue your dreams

(See any standard work on experience with VCs)

Operating Environment

Commercial: Windows, Visual Basic, ActiveX, Java

OSS: Unix, bare-bones Windows (e.g. Cygwin)

- Many OSS ports to Windows seem designed to be as Windows-hostile as possible

A difficult problem to resolve

- This is the boring stuff that you have to pay people to do

An ActiveX wrapper will satisfy a lot of people, but…

- Most software doesn't have an interface amenable to an ActiveX interpretation
  - Object-oriented
  - Methods, properties, attributes

Operating Environment (ctd)

OSS can target environments that commercial software can't

- "Tropically tolerant code" (Hermann Chinery-Hesse, Ghana)
- Writes data to disk frequently
  - Avoids problems due to power outages
- Works offline as much as possible
  - Don't require an Internet connection
- Runs on old/resource-limited hardware
  - 386 PCs with 4MB of RAM

However, OSS won't work in some countries

- If company employees have the source to the apps they use, they can modify the code to syphon off funds

I don't do toilets and I don't do Windows…

Microsoft is not the competition, Microsoft is the environment

- Microsoft have 70% of the revenue of the other 100 top software companies *combined* (Soft*letter)

Ignoring Windows loses you 95^{+}% of potential paying customers

This includes passively ignoring Windows…

- Non-native Windows GUI
- `tar.gz`
- Having to download and install additional software

Versioning

Microsoft has conditioned users to expect

- Only versions ≥ 3 are usable
  - Windows NT started at 3.1 to give the illusion of stability
- Service packs are an essential part of any software

Good	Bad
Version 4.2a SP3	Version 0.3.71 Beta 3

- Use versioning that reflects this user conditioning

Versioning (ctd)

All OSS is in permanent beta

- Commercial user complained about the presence of an OSS component labelled “beta 3”
- Used a hex editor to make it “SP3”
- User was happy

Releases

OSS: It’ll be ready when it’s ready (see “Permanent beta”)

- Most developers who are in it for the long term take pride in their work
- Releases take awhile to appear

Commercial: You can release crap as long as you include a URL for service packs

- Release early and release often
  - Versions like 4.2.021.1034 aren’t uncommon in the Windows world

This is difficult to do for perfectionist developers

Releases (ctd)

OSS release

- Format = `source.tar.gz`
- Install = `make; make install`

Commercial release

- Format = `binary.msi` / `binary.rpm`
- Install = double-click

Releases (ctd)

Users detest software that they have to tweak/configure to try out

- "What's this Gimp thing, and why should I have to replace half my OS binaries to test-drive it when I can buy Photoshop?"
- `make` or even `unzip` loses you > 50% of potential new users

Use automatic install scripts or dynamic binding/linking

- Put screen shots on your web page

Once they're hooked you can force them to replace half their OS to run your app

Online Presence

Commercial sites have Flash, JavaScript, style over substance

OSS sites are simple, contain technical detail, feature direct links to CVS/source code

Maintain two sites

- `foo.com` features graphics, animation, link to "Free test drive version"
  - Without this, people assume that they have to pay to try it
- `foo.sourceforge.org` features source code links, bug tracking, previous betas

Both sides provide links to the other if users go to the wrong one

Product Roadmap

Keep one handy for people who ask for it

- Shows where you're heading
- The company is robust in terms of its vision

No-one ever takes you to task over it if it doesn't eventuate

- "We looked at it but there was insufficient industry support"
- "We went with an alternative system that works just as well"
- "Microsoft have said that the future lies elsewhere"