Oriented Specification System

Pieter Hintjens,[a]

(a) Digistan

DOI: 10.5033/ifosslr.v2i1.32

Abstract
Pieter Hintjens tells the story of COSS (the Consensus-Oriented Specification System), a toolkit for emerging digital standards. COSS gives workgroups the tools to develop new digital standards with minimum bureaucracy. COSS is a product of the Digital Standard Organization (Digistan), which the FFII open standards workgroup founded with other participants in 2007. In this article he explains why Digistan built COSS and how it works.

Keywords
Law; information technology; Free and Open Source Software

Info
This item is part of the **Platform** section of IFOSS L. Rev. For more information, please consult the relevant section policies statement. This article has been independently peer-reviewed.

Standards are essential to any industry. By acting as contracts for interoperability, standards free customers to choose among suppliers. Standards thus create competition among suppliers that forces those suppliers to improve quality and/or reduce prices. By slowing and stopping disruptive innovation in layers where it is no longer needed, standards allow layering of new industry segments on top of old ones. For example, wiring and voltage standards for electricity underpin the business of selling electrical appliances, and standards for shipping containers massively increase the efficiency of the logistics and transport industries.

Standards-setting processes also reflect the nature of the specific industry concerned. Industrial standards are usually set by consortia, often backed by state regulation. For some industries such as telecommunications, standards are a prerequisite for development, whereas other industries develop standards later, in mature areas. George Stephenson may have opened the Liverpool and Manchester Railway in 1830, but the UK did not converge on a standard gauge until 1892.[1]

1 http://en.wikipedia.org/wiki/Standard_gauge

Software standards, and the processes which facilitate their development, reflect the nature of the software industry. In particular, the conflicts between old and new money, between large and small organizations, and between command-and-control and collaboration are evident.

Free and Open Source Software (FOSS) has been described by some - mainly proprietary software businesses -- as a "business model". That seems inaccurate. Most collaborative projects have no initial business plan beyond "1. Popularity, 2. ???, 3. Profit!" but many succeed nonetheless. It is more useful to view FOSS as a social/legal technology for software development. Every FOSS community depends on a contract that encourages or forces participants big and small to share their knowledge and work. Perhaps as importantly, end-users are brought into the social contract. Giving end-users the ability to inspect, improve, and if necessary, fork the source code means that they can help steer the direction of development. This happens even in small FOSS projects, if end-users get involved early on. Decisions about whether or not to add new features are more accurate when users are involved in the process.

It appears that the FOSS social practices and legal infrastructure – primarily represented by software licenses like the BSD (Berkeley Software Distribution license) and GPL (GNU General Public License) – result in faster technological development than the conventional proprietary software approach. We do not have studies to prove this, only experience and observation. One recent visible example is the Linux-based open source Android operating system for mobile phones, which seems to be overtaking its proprietary rivals both in rate of innovation (in June 2010 we see four versions on the market: 1.6, 2.0, 2.1, 2.2) and in market success[2]. Other open, collaborative projects such as Wikipedia provide more dramatic success stories.

Small FOSS teams often need to compete with larger established firms in the market[3]. This requires a divide-and-conquer approach (many smaller teams competing with a few large ones), which requires teams to agree on shared file formats, protocol, APIs, and languages so that components from different teams can work together in larger systems. These agreements will be more widely used and respected when they are properly written, formally endorsed by some body, accessible to all, and free of all constraints on use.

The supplier of a commercial product facing FOSS competition may arbitrarily change interfaces, file formats, protocols and such, to deny interoperability and thus keep customers captive. The classic example is the format for Microsoft Office documents, which started changing in each release of the software, when open source competitors began to accurately read and write files in that format. When these interfaces, file formats, and protocols are documented and freely usable, customers have a means by which to demand compatibility from the supplier. Without a written specification, the implied contract provides "compatibility with previous versions" at best, and suppliers can always introduce change under the excuse of "innovation." With a written specification, the implied contract becomes "compatibility with the specification."

Thus, properly written specifications of interfaces and formats (even without the backing of a formal standards body and even without cooperation of all vendors) are a key part of allowing

2 "Android overtakes Apple in US smartphone market", http://arstechnica.com/apple/news/2010/05/android-overtakes-apple-in-us-smartphone-market.ars
3 http://en.wikibooks.org/wiki/FOSS_Government_Policy/Strategic_Importance_of_FOSS

FOSS teams to compete in scale with entrenched vendors. Even within a single FOSS project, documented specifications for interfaces, protocols, and formats create contracts between developers who may work far apart in space or time.

Microstandards

The first "Requests for Comments" (RFCs)blished 30 years ago, were small software specifications that solved specific pieces of a vast decade-long puzzle: how to build a global computer network available to all at a cost approaching zero. These RFCs were built by small teams, often just one or two people, using a modest process based on peer review and rapid maturation. They escaped the normal commercial pressure to turn the standard to favour any particular vendor In contrast, the industry consortia that were solving basically the same problem developed heavily patented telecommunications standards that created a captive market, dominated by incumbent telecommunications firms.

There can be little argument that this approach was wholly successful, because every competing networking technology, developed at great cost by major firms like IBM and international standards development organisations, lost against the Internet RFCs. Standards such as token-ring, LU6.2, SNA and VTAM were quickly eclipsed by the open RFC approach. The few areas where proprietary networks still dominate (such as mobile telephony) are remarkable for their high cost and lack of true competition. In 2010 Engadget.com reported[4] on the high cost of international mobile telephony, and especially roaming mobile data, concluding "*Do not ever, under any circumstance, roam with your American mobile broadband card. You'll never pay off the roaming bill.*"

The term "microstandard" means, in this context, a free and open standard that is short (perhaps a dozen pages and certainly less than 100), developed by a small team, and part of a stack of free and open standards. In general microstandards can evolve faster thanks to rapid maturation-layering cycles; they can be more accurate (meaning: good solutions to real problems) thanks to review by a wider, more diverse community; and they can be cheaper to implement and use, thanks to simple, focused designs. However, these qualities do not guarantee success, and only a minority of potential microstandards are ever properly written down, published, shared, and built upon. If we were to count, we would find tens of thousands of potential microstandards worldwide, most of which never emerge from a niche: little scripting languages, data schemas, file formats, ad-hoc protocols, encodings, patterns, templates, and APIs. This informal economy is massive, but it is also massively inefficient. Most of these potential microstandards exist only as source code.

The vast bulk of these microstandards remain potential and local because their authors lack experience and economical tools for properly writing and maintaining standards. Doing so is just too hard and too costly. The inherent friction between innovation and standardisation has turned into a barrier against innovation in the standardisation process itself.

This has been my experience in three decades of software architecture and programming: where

4 http://www.engadget.com/2010/06/09/how-to-stay-connected-while-traveling-internationally/

there could be tens of thousands of properly built microstandards, each clearly documented and published under standard licenses, we see barely dozens or hundreds. The standards process is in my view failing at the grass roots level, exactly where the FOSS economy needs it most of all.

Software standards have been the focus of conflict between smaller FOSS teams and larger established players, and these conflicts are starting to interfere with standards setting at the international level. The key case is the ISO (International Organization for Standardization) adoption, over several years, of two conflicting formats for office documents.

Standards War

In 2007, a world-wide standards war broke out over document formats. On the one side, Microsoft, the largest global software business, was advocating its OOXML (the confusingly named "Office Open XML"). On the other, a coalition of activists, engineers, and FOSS businesses advocated the adoption of the Open Document Format (ODF). In the end, after a year of massive, global conflict in hundreds of national standards committees, and despite a well-organized global campaign by the pro-ODF coalition, OOXML was given the ISO stamp of approval. It was a dirty fight in which committees were stuffed, coerced, and bypassed. ZDNet wrote[5]:

> *"What now transpires is that [Microsoft] have hijacked the committee and are not only stepping outside the established procedures but are also working towards amending the standard in order to make it compatible with Office 7, rather than building or amending their Office Suite to be compliant with the ISO standard. Apparently, it's only through leaks the we can find out what's happening."*

For most of the conflict, the coalition was trying to decode the rules that defined who could vote, and when. At the same time Microsoft was rewriting them to ensure it would win any final vote, in any case. A typical story came from Denmark, which, as ZDNet reported[6] "decided to back Microsoft's Office Open XML document format, reversing its previous disapproval and bringing the format closer to fast-track approval by the International Organization for Standardization." Wikipedia documents[7] complaints about national processes in Norway, Sweden, Portugal, Finland, Switzerland, India, Australia, Germany, Poland, Spain, and the UK. In Poland, says the Wikipedia article, "*Borys Musielak, a member of Poland's Linux community, wrote on the PolishLinux website that Poland's technical committee KT 171 rejected Office Open XML. The vote was invalidated and assigned to KT 182. A member of Poland's Linux community believes this was due to "reorganisation in the Polish standardisation body." KT 182 voted to approve Office Open XML.*" The NoOOXML.org campaign followed and documented these and other similar stories via hundreds of contacts across the world. That documentation is still online and accessible via the NoOOXML.org website.[8]

5 http://www.zdnet.co.uk/blogs/moleys-musings-10008506/ms-stuffs-ooxml-jtc1sc34-maintenance-committee-10014322/
6 http://www.zdnet.com/news/ooxml-standard-vote-down-to-the-wire/194601
7 http://en.wikipedia.org/wiki/Standardization_of_Office_Open_XML#Reactions_to_standardization
8 http://www.noooxml.org

Many people were left wondering if the traditional standards business could still represent the needs of the modern software industry or whether they were to be dominated by narrow but powerful individual corporate interests.

Standards processes appear to reflect their industry as a whole. The great shift over the last twenty years has been from traditional ways of making software to the collaborative approach of FOSS. It has become evident to many in the industry that FOSS is a better way to make software, producing more accurate solutions faster and cheaper than older closed source approaches. The difficult question is rather: "how do we make money when the software is free?"

For monopolists like Microsoft, the answer is to prevent the software from being free at all. This requires a number of mechanisms, including software patents, EM (original equipment manufacturer) licensing agreements, proprietary languages and frameworks, and arbitrary changes to key file formats, interfaces, and protocols. If governments and businesses insist on those being standardised, that requires control of the standardisation process. If governments and businesses insist on ISO standards, that eventually means controlling the relevant parts of ISO itself.

And so in early 2008 the unthinkable happened and Microsoft effectively took over parts of ISO, both directly and through proxies. Most national committees and the ISO secretariat were persuaded, bullied or otherwise encouraged to accept Microsoft's proposal with minimum changes as a formal international standard. The anti-Microsoft coalition was just as ready to use political influence and committee stuffing, but failed to appreciate the depths to which Microsoft was willing to go in order to save its Office monopoly.

Lessons from the Standards WarThe relevance of this case is as a record of how easy it was for a determined large firm with deep pockets to turn what had been an advantage for proponents of ISO certification of ODF into a direct liability. Since Microsoft now in effect controls ta sufficient number of national standards-setting organizations, it effectively controls ISO standards setting, and thus it controls ODF and is positioned to slowly strangle it.

Participants in the process (both in committees and outside) took home various conclusions from this dramatic series of events. Many decided that it was time to make peace with Microsoft. Others decided to create new institutions like the Open Web Foundation. Others, and especially the activist core of the NoOOXML.org campaign, came to the conclusion that ISO and other institutional processes had become irrelevant, if not actually distracting time wasters for standards engineers working on open standards, often *pro-bono*. For example, after years of work by participants around the world to get ISO approval of ODF as the standard for document formats, this very success may have spelt the death of ODF.

The activists, among them myself, realised that any standards-setting institution, including the W3C(World Wide Web Consortium) and IETF (Internet Engineering Task Force), represents a target for determined corporate attacks of the kind we had witnessed during the document format standard-setting process within ISO. Ultimately, any process that depends on goodwill is vulnerable to hostile takeover.

What had been missing, we concluded, was a standards-setting process that matched the fully distributed, attack-resistant processes of the FOSS ecology. It was not enough, we felt, to be free today, we needed the unambiguous guarantee of freedom tomorrow. As Andrew S. Groove said, "Only the paranoid survive."[9]

Thus, in 2007 we created the Digital Standards Organization (Digistan) with a mission to "promote customer choice, vendor competition, and overall growth in the global digital economy through the understanding, development, and adoption of free and open digital standards". Our first publication was a new definition of "free and open standard" based on analysis of the standards development process.[10] The core definition is that "*a free and open standard is a published specification that is immune to vendor capture, at all stages in its life-cycle*".[11] “Vendor capture” is a term chosen to focus attention on the economic interests of those who make standards. Those making money from products – vendors – have an economic interest in selling more products by reducing competition. They can achieve this by taking control of the standard – capturing it – in various ways.The language of Digistan's definitions is heavily influenced by the standards war of 2007, because of the enormity of this event in shaping our understanding of the economics of standards development.

The Effects of Vendor Capture

Frequently, a group of firms will create a standards-setting organization and then exclude upstart competitors from participating. With the addition of some patents, even though the patents may be made available under a so-called “reasonable and non-discriminatory” licensing scheme, this creates a neat and legal cartel that is immune from serious competition authority oversight. Even when there is flagrant and long-term extortion of the market, patent pools make legal behaviour that would otherwise result in prison terms[12]. The mobile phone sector is a case worth studying.

Ewan Sutherland of the LINK Centre, University of Witwatersrand and CRID, University of Namur, writes[13] that in 1996 the European Commission (EC) opened its first dossier on international mobile roaming (IMR), at the behest of the mobile operators, who sought and received an exemption from anti-trust laws of the EC Treaty (now Article 105 (1) EU). In 1999 the International Telecommunications Users Group (INTUG) filed a complaint with the EC about persistently high prices. Cases against four operators were closed without any penalties, and an inquiry into the handling of these cases was abandoned, with no explanation to the high prices. There were a set of directives in 2002, and two more Roaming Regulations in 2007, but prices for IMR remain high, especially for roaming data. Sutherland concludes "*The mobile network operators maintain there was no problem. The EU institutions,... and many users maintain that*

9 http://www.intel.com/pressroom/kits/bios/grove/writings.htm
10 http://www.digistan.org/text:rationale
11 http://www.digistan.org/open-standard:definition
12 http://www.crowell.com/documents/Antitrust-Risk-in-Patent-Pool-and-SSOs-Avoiding-Price-Fixing-and-Exclusionary-Conduct.pdf
13 The European Union Roaming Regulations, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1574981

there was and is -- even if they can only describe the symptoms, not explain its causes."

To uncover the causes of the problem, and why national and EU regulators have been unable to correct what looks very much like collusion by dominant market players, we need to understand that the mobile telephony market depends on standards, and these are very heavily patented. As Tobias Kaufmann explains[14], the GSM (Global System for Mobile Communications) market depends on 554 patent families (not individual patents) declared "essential" by their holders. 74% of these patents are held by four firms.

Kaufmann explains how patent licensing is used to create the cartels that dominate the European and American mobile phone markets:

> *In 1998, the ITSUG (International Telecommunications Standards, User Group) authored a complaint to the European Commission in which it summarized the GSM licensing problems in stating that the inability to acquire timely licenses coupled with the uncertainty of essentials lead to "costly and complex licensing negotiations" and "excessive cumulative royalty rates", thereby creating substantial transaction costs and high barriers to entry to the GSM market. In addition, the ITSUG alleged the existence of a "low/zero club for established European telecommunications players" while others have to pay royalties of up to 40% of the ex-works selling price".*

So while GSM depends on standards, those standards and the market they depend on are entirely captured through patents, and escape normal competition pressure to lower prices. By adding new patents to the pool of patents that a business must license, the normal 20 year lifespan of a technology like GSM – which originates from 1991 – can be extended almost indefinitely, and indeed Kaufmann reports that only 149 of the 554 patent families in the GSM patent pool are judged essential by technical experts. The rest exist only to perpetuate the licensing barriers. As Wikipedia states[15], "*Patents remain a problem for any open source GSM implementation, because it is not possible for GNU or any other free software distributor to guarantee immunity from all lawsuits by the patent holders against the users. Furthermore new features are being added to the standard all the time which means they have patent protection for a number of years.*"

In a properly competitive market based on free and open technological standards – such as the traditional wired phone network – we would expect costs to fall by 50% every 18-24 months, or value to double every 18-24 months (hence "Moore's Law", stating that chip density will double every two years[16]. If it cost 1.00 UKP to make a 1-minute international phone call in 1990, today that cost should have fallen by half, ten times by 1000 times.

Free and Open Standards

The "immunity from capture" analysis is key to Digistan's work. Many definitions of "open standard" work by defining properties of an open standard. This mixes symptom with cause. A

14 Intellectual Property in Broadband Mobile Telecommunications: Predictions on 4G WiMAX , http://www.frlicense.com/IntellectualPropertyinBroadbandMT.pdf
15 http://en.wikipedia.org/wiki/GSM
16 http://en.wikipedia.org/wiki/Moore%27s_law

standard may be open because it is not yet captured. Or it may be open because it cannot be captured. We need to express and measure its immunity from capture, not the consequences of "so far, so good". For example common term “open standards” is often stretched to include standards with “reasonable and non-discriminatory” (RAND) patent licensing conditions.

Standards have a lifecycle that often lasts many years. At various stages of the lifecycle, different tactics can be used to try to capture the standard. For example, a standard developed by a community, free to read and implement, can be trivially captured by a single patent, by a decision of a copyright holder to start charging for updates, or by an injection of complexity that makes it impossibly hard for independent vendors to implement.

We would argue that de-facto, any standard that affects a significant market will be subject to such capture attempts. In Digistan we therefore defined this new term, "Free and Open (Digital) Standard" to express this additional criteria. A free standard will be open, and continue to be open. A standard that is only “open” (now) is not necessarily free, nor open in the future.

The resonance with "free and open source" was not deliberate but it is accurate, and one could argue the same cause-and-effect relationship between "free" and "open" in software.

What makes a free standard? Here we diverge from conventional standardisation wisdom and take an idea that has been wildly successful in FOSS projects, where it plays an essential role in preventing certain forms of capture: the right to fork.

The Right to Fork

We realised that forking was a necessary right when studying the ways that a standard could be captured at different phases in its lifespan. It seemed a counter-intuitive proposal at first, and one that provoked extensive debate among the Digistan founders. Even a small, young standard exists because it is definitive. How can the right to fork make standards better? Surely it would result only in arbitrary, destructive variations?

Ultimately any specification must aim to be definitive and to get approval from a body like the IETF, but such approval may not be achieved for five or more years after the specification is documented and implemented. Our goal is to ensure that a specification survives those early years and gets ready for international certification unharmed and 'un-captured'.

There is a natural tension between experimentation and standardisation. These are often described as opposing processes. In fact, they are just different stages in a lifecycle. Early on, engineers must experiment in order to learn what the best solutions are. As they learn, they capture their growing body of knowledge as standards. Each standard, or stable version of a family of standards, encapsulates some layer of knowledge, freezes it, and creates a solid basis for constructing further layers.

Capturing one of these building blocks effectively captures everything that depends on it, much as buying a software company effectively buys the market for that software.In both cases, the right to fork is leverage that encourages good governance. If a firm that buys a FOSS project makes bad decisions, key developers and contributors fork the project, and walk away.

Here is a practical example of how the right to fork helps keep a standard open: firm A develops a standard in a lucrative new market. Firm B buys firm A and acquires the copyrights of the standard. It then creates a new version that is compatible with firm B's products, and not those of competitors. It uses patents to prevent competitors implementing that modified version of the standard except for a heavy fee. It uses copyright to prevent further modification of the standard to avoid the patented aspects. In effect, firm B has captured the market and everyone who built on the original standard faces the choice of stagnation, or the higher prices charged by firm B.

Now, in a second scenario, the standard was developed under a license guaranteeing the right to fork. When firm B buys firm A with the intention of controlling the market, other implementers fork the standard. They create a new version that bypasses the patents, and it is that modified version that is promoted to the market for adoption. Firm B has no mechanism with which to capture the market, so instead it competes with better products at lower costs or it is forced to conform to the forked standard.

GPL for Free and Open Standards

The idea that forking was a solution rather than a problem in standardisation processes informed our search for an appropriate license for standards texts. We consulted the Free Software Foundation (FSF), who told us that the FSF did not have a license appropriate to standards specifically. In the end we chose the FSF's General Public License (GPL) v3.[17], to avoid license proliferation.

Using the GPL for a specifications text is unusual but we can make more sense of it if we imagine that a specification might often be a kind of software. For example, a specification may include an XML schema or a set of formally described methods and arguments. Many modern standards start to look like a mix of metadata and comments, and thus start to overlap with software source code.

To apply the GPLv3 to a specification, we state that the specification is "source code" and we ask that contributors license their work under the GPLv3. An alternative to the GPLv3 would be the Creative Commons Share-Alike 3.0 license but this lacks the GPLv3's provisions against software patent abuse. At the least, using the GPLv3 gives the right to anyone who feels they can "do better" to take the standard in question, improve it, and release their new specification. Although an untested speculation, use of GPLv3 may force vendors who have made silent extensions to publish their revised specifications.

Creating the Digital Standards Organization

Institution building is costly. In July 2008, David Recordon announced the Open Web Foundation (OWF) at the O'Reilly Open Source Convention. Since then, the OWF participants have been occupied with building the organisation: collecting members, electing a board, and defining the

17 http://www.gnu.org/licenses/gpl.html

goals of the organization.[18] OWF is modelled after the Apache Foundation and has strong support from industry. However, after 18 months it is not yet ready with tools for standards developers.

When we started to build Digistan we decided to make a lightweight structure that would cost as little as possible to construct and to maintain. In practice this meant no legal entity, no formal membership, no formal elections or board. To create Digistan we used a number of email lists, based on a model we inherited from the FFII. There is an "eboard" (extended board), which takes the decisions. There is one additional list per project, plus a public discussion list. We also use a number of microsites (built using a wiki technology), which let us rapidly create homes for the various projects we wanted to start.

We started discussions in October 2007 and published our "standards for standards" in April 2009. During that two year period we also established chapters in Poland, France, Germany, and Spain, and we launched the Hague Declaration on open standards.[19]

A Reusable Legal Framework

When JPMorganChase, Red Hat, my firm (iMatix) and others created the AMQP (Advanced Message Queuing Protocol) Working Group in 2006, the legal paperwork involved in the specification process was extensive. The legal and organisational effort involved seemed disproportionate to the objectives, and in practice, the need to send contracts around for signing became the major barrier to attracting new participants. For Digistan we decided to create the lightest possible legal framework, based on the notion that each contributor licenses their contributions to others for reuse under the GPLv3. The explanatory note says[20]:

> *"This policy is specifically designed for a community with no centralizing legal entity. There are no transfers of copyright. Every contributor owns his/her work, in perpetuity, but grants a unilateral license for others to use and extend it under the conditions of the GPLv3."*

The lack of a central legal entity means no formalities for contributors except a click-through acceptance of the IP policy. The text of the policy is also short at 1,400 words, since the bulk of the legal framework is provided by the GPLv3.

The core text of the policy reads:

> *"by submission of a work to this Website in the form of a Contribution, the Contributor merges their work with the Website, with or without other Contributions, to create a Derived Work, owned by the Contributor, the Operators, and any other Contributors. The Contributor agrees irrevocably to license this Derived Work under the License and the terms of this Policy. The Contributor retains all rights over their original work."*

18 http://groups.google.com/group/open-web-discuss?hl=en%3Fhl%3Den
19 http://www.digistan.org/hague-declaration:en
20 http://spec.digistan.org/main:intellectual-property-policy

With respect to forking, the policy states:

> *"third parties may create Derived Works under the terms of the License. Derived Works may not contain misleading author, version, name of work, or endorsements. Software applications that implement specifications are not Derived Works and do not fall under the terms of the License. The use of a fragments of specification for purely illustrative purposes does not create a Derived Work."*

With respect to patents, the policy states:

> *"to the extent that a Contributor or the organization he or she represents or is sponsored by (if any) has or acquires patent claims that would be necessarily infringed by a compliant implementation of any part of any Specification, they grant a perpetual, irrevocable, non-exclusive, royalty-free, world-wide right and license to the Community with respect to their patent claims for such purpose."*

The policy assumes that a specification is developed within a microsite (website) and it classifies the entire website, along with associated email lists, irc channels, and other communication resources, as a "work". This means that when a contributor has agreed to the policy for a single website, that website can safely house many specification projects, and contributors can remix those as needed.

How COSS Works

The 1/COSS specification[21] is meant to make it easy for small teams to write, prove, and improve technical specifications. In other words, to produce small standards. It aims to be lightweight, cheap, transparent, and to fit the natural pattern by which experiments become accepted solutions, over time.I have previously said that experimentation and standardisation are two phases of the specification lifecycle. Based on experience from the AMQP specification process,[22] I defined a formal lifecycle that covered the phases of a specification as it moved from experiment to retirement:

- Raw Specifications - all new specifications are raw specifications. Changes to raw specifications can be unilateral and arbitrary. Those seeking to implement a raw specification should ask for it to be made a draft specification. Raw specifications have no contractual weight.

- Draft Specifications - when raw specifications can be demonstrated, they become draft specifications. Changes to draft specifications should be done in consultation with users. Draft specifications are contracts between the editors and implementers.

- Stable Specifications - when draft specifications are used by third parties, they become stable specifications. Changes to stable specifications should be restricted to errata and clarifications. Stable specifications are contracts between editors, implementers, and end-

21 http://www.digistan.org/spec:1
22 http://www.amqp.org

users.

- Legacy Specifications - when stable specifications are replaced by newer draft specifications, they become legacy specifications. Legacy specifications should not be changed except to indicate their replacements, if any. Legacy specifications are contracts between editors, implementers and end-users.

- Retired Specifications - when legacy specifications are no longer used in products, they become retired specifications. Retired specifications are part of the historical record. They should not be changed except to indicate their replacements, if any. Retired specifications have no contractual weight.

- Deleted Specifications - when raw or draft specifications are abandoned, they become deleted specifications. To change a deleted specification, the editor should first make it a raw specification again. Deleted specifications have no contractual weight.

What this lifecycle does is to formally define the contractual weight of any specification depending on its phase. Thus it is clear to all parties how much change they can expect, or conversely, make.

When it comes to editing a specification, we decided to restrict editing to a single person. COSS says, "*a specification has a single responsible editor, who is the only person that can edit the text and change its status. A specification may also have additional contributors who work through the editor. The editor is responsible for accurately maintaining the state of specifications and for handling all comments on the specification.*" The theory is that restricting a specification to one editor encourages a divide-and-conquer approach,i.e., a large body of work will naturally break into pieces, one per expert, creating a more layered result. Anyone has the right to fork: a fork is considered a separate specification, with its own editor.

Finally, COSS resolves natural conflicts between teams and vendors by allowing anyone to define a new specification, remixing part or all of any existing specifications as desired. There is no editorial control process except that practised by the editor of a new specification. The administrators of a domain (moderators) may choose to interfere in editorial conflicts, and may suspend or ban individuals for behaviour they consider inappropriate.

Who decides what is an authoritative specification? In traditional consortia this is done by vote. But votes are easily manipulated. COSS rejects the notion of a single authoritative specification, favouring choice and competition instead. What this means is that we ask the market to choose on the assumption that, as competing specifications (if there are multiple choices) move through from experimentation to stability, implementers will agree on the best specification. In effect we trust implementers, driven by users, to decide what is authoritative. As COSS says, "specifications have no special status except that accorded by the community."

Use cases for COSS in Real Life

In 2007-8 we tested COSS on real specifications, documenting and tracking eight small specifications (including COSS) related to AMQP.[23] Over a year or so, we were able to accurately map specifications as they moved from raw through to retired. From this experience we developed a reusable template website[24] which new workgroups could copy and use for their own specification work. This gives teams a completely functional specification tool (working wiki, legal framework, terms of use, etc.) in a few minutes.

We've since used this template website in three further specification projects:

- A set of specifications grouped under RestMS.org.[25]
- A set of specifications for the ZeroMQ[26] project, at http://rfc.zeromq.org.
- An experimental web protocol (BWTP), at http://bwtp.wikidot.com.

Today we're starting to promote COSS more widely.

Conclusions and further work

In this article we've examined the key role that free and open standards play in competition, particularly between smaller free and open source teams, and larger proprietary software businesses. We’ve looked at the history of standards development in the software world, and seen that traditional standards setting seems to be sub-optimal at best, and failing at worst. When powerful monopolies are threatened by new standards, they may go to great lengths to subvert those standards. When large firms work together to make new standards, they may protect these with patent pools that can keep prices inflated – legally – by many orders of magnitude.

Digistan has examined the causes of these failures and concluded that a successful free and open standard must be robust against “vendor capture” at all stages in its life-cycle. In other words, that standards can represent the economic interests of users rather than those of product suppliers. One of the key defence mechanisms against capture is “forkability,” i.e. the right for anyone to create a derived specification.

Further, Digistan has developed a set of tools - a legal framework and template website - that now allow any workgroup to create a home for specifications, in just a few minutes. The Digistan legal framework uses the GPLv3 as its default license but allows other licenses to be plugged in.

On top of this legal framework, we have built a standardised process - COSS - that gives implementers peace of mind when it comes to how much change they can expect in specifications. This lifecycle formally defines specifications as raw, draft, experimental, stable, legacy, or retired.

23 http://wiki.amqp.org
24 http://spec.digistan.org/
25 http://www.restms.org/
26 http://www.zeromq.org

Each state has different contractual weight.

Finally, COSS and the underlying legal framework need no centralising organization. They allow a fully distributed peer-reviewed specification process. Authors own their own work and license it for reuse by others. Final authority is delegated to the community, i.e. implementers and users who invest in specifications.

The discussion of whether or not to allow forking has now become mainstream[27] with the HTML5 specification. This is essentially a fork of the W3C's HTML specification, yet forking is still seen as a bad thing by most respondents. We have tested these tools over the last two years in four different specification domains, and they do seem to work successfully. We expect that over time people will understand that forking of specifications is an essential freedom, and specifications will move more and more to share-alike licenses. Just as with software, the license defines a community, and the rules on remixing the work of others play a significant role in growing the community.

Acknowledgements

The Digistan eboard (founders) email list was instrumental in developing Digistan's legal framework, IP policy, and template web sites. Mark Webbink and Graeme West provided very helpful review comments and improvements to the article.

About the author

Pieter Hintjens *is an expert in the creation of collaborative communities. He has 30 years of experience in the software business, starting his first company at the age of 18. He wrote his first GPL application, Libero, in 1991. He is past president of the FFII, which works for freedom of the software economy. As FFII president he organized the European Patent Conferences (EUPACO), the Digital Standards Organization, the European Software Market Association (Esoma), the Campaign for Ethical Patents, the NoOOXML.org campaign, and many smaller projects. He is also founder and CEO of iMatix Corporation, which builds the worlds' fastest free messaging software, ZeroMQ. He was the lead editor for AMQP (the Advanced Message Queuing Protocol), and for the RestMS web messaging protocol.*

27 http://www.w3.org/2010/Talks/doclicense-20100323/#%281%29