Engaging Developers in Standards Development; the Cetis Code Bash Approach

A white paper

By Lorna M. Campbell and Adam Cooper

cetis

Centre for Educational Technology, Interoperability and Standards

Engaging Developers in Standards Development; the Cetis Code Bash Approach

By Lorna M. Campbell and Adam Cooper

Abstract

A linear process in which a written standard is created and then implemented in software is liable to fail for many reasons arising both from the difficulty in writing a specification that is sufficiently precise and accurate while also allowing for necessary flexibility in use, and from the intrinsic complexity of the human activities and IT systems in which it will be realised.Engaging software developers in the standards development process has been found to be an effective means to improve the written standards, to enlarge the scope of practical interoperability between software, and to identify and share effective practice. Over a period of years, Cetis developed an approach to this kind of engagement which we called a "Code Bash". This white paper outlines the motivation, typical outcomes and practicalities of running a Code Bash and is intended to motivate people working in either formal or informal standards-development settings to engage developers in the process and to provide them with some ideas to adapt to their own setting.

Keywords: code bash, plugfest, interoperability testing, standardisation process, developer

For the purposes of this white paper, "standards development" includes activity in formal standards bodies through to informal collaborations that are intended to lead to a documented common way for multiple parties to exchange data in an interoperable fashion. We are concerned with ICT standards and with the idea that stakeholders not involved in the documentation should be able to use the standard to effect interoperability without the need to develop and test against each unit of software they will exchange data with.

1. Problem: Written Standards do not Reliably Support Practical Interoperability

Our experience and our observation of the experience of people Cetis has worked with over many years leaves us in no doubt that practical interoperability is not guaranteed by a written standard. There are many reasons for this that arise from the environment in which the standard is implemented in software. While a badly written standard evidently decreases the changes of practical interoperability, even scrupulous care is limited in what it can achieve.

Even when standards are written by people with a collective understanding of both the application domain and current practice in writing interoperability specifications, standards may contain ambiguities, errors, or not cover areas that are found to be important for a given application in practice. In some cases, it may be that a perfectly logical structure fails to match common practice. These weaknesses are not always easy to pick up from a review of the specification document.

There is also the influence of context, which shapes the way a standard is implemented. Developers may interpret and implement elements of a standard in different ways. This might arise from different uses of the same term to refer to different concepts in different contexts or to a given entity having a different meaning arising from differences in its role (e.g. different significance in the business processes of UCAS and a university). The case of what "course" means illustrates the problem of interpreting a label[1] and is good evidence to justify the assertion that good standards should clearly and thoroughly define terms and not rely upon labels. Context also influences the way parts of a standard are chosen or how some of the necessary freedoms are exercised, in other words: how the standard is "profiled". This can happen explicitly or implicitly and, while each standard may each be conformant to the standard they may not be interoperable.

Furthermore, unforeseen issues may occur at the point of implementation that inhibit interoperability. Maybe the processes in different organisations operate sufficiently differently that, although the data can be represented in a common form, there is a mis-match between which parts are created or needed at different times. Maybe there are limitations arising from the technical approach to transferring the data.

These issues are worse when a linear model is adopted in which adoption follows specification. Most of the established standards bodies and consortia, and experienced standards developers, adopt an iterative process, with at least one public release of a draft specification with the potential for feedback. Over a period of years Cetis ran a number of developer-centric events focussed on improving the feedback process with the aim of achieving practical interoperability. The idea is simple: to get people together to try to share data between different software using draft standards, or to test a given standard in different applications. Developers fixed code, resolved differences, converged on shared understanding and identified ambiguities and errors in the standards on the day. This paper summarises some of the key points and experiences in running these kind of events, which we called "Code Bashes".

2. What is a Code Bash?

A Code Bash is an informal event, in which software analysts and developers from different organisations meet up with their software on laptops or with remote access and exchange data between systems, simulating typical and exception-case interactions. If difficulties are found; crashes in the code, unexpected results, or round-tripped data that isn't identical ,they try to understand the reason. Where possible, fixes or temporary work-arounds are coded-up and the process repeated.

Typical issues include: different interpretations or a mis-understanding of the standard; a coding error; inconsistency in the standard; irreconcilable differences in semantics of business systems and the standard.

Improvements in both the software and the standard are usually found and the following outcomes are naturally achieved:

- A bank of test-case files that are either agreed as being contingently correct or contain pitfalls.
- An assessment of readiness of the standard to advance from draft to final.
- Identification of gaps; entities not present in the standard.
- Bug fix lists for software and for the standard.
- Mutual understanding between developers from different businesses of the different meaning attributed to what are ostensibly the same entities.

[1] *The HESA report, "What is a Course?" by Andy Youell summarises the situation with reference to UK Higher Education. http://www.hesa.ac.uk/dox/publications/The_Course_Report.pdf (work funded by Jisc)*

- Clarity on the kind of implementation guidelines or other kinds of developer support/education that may be needed. Maybe even volunteers to contribute them.
- Good personal relationships and commitment to future bilateral working.

2.1 Running a Code Bash

Here are some check-points for running an effective Code Bash, drawn from Cetis' experience.

PRE-REQUISITES	PROMOTING PRODUCTIVITY
Identify independently developed software, preferably more than three separate sources or sinks for exchanging data. Establish a secure place where instances of exchanged data can be stored and documentation created on the systems that exchanged it and the results obtained. It should be quick and easy to post test data, retrieve it for testing, to document the outcome, to post revised/fixed data, etc with tracking. Secure access to data that can be exchanged and has been cleared of sensitive personally identifying information etc. It may be helpful to have some sample data available for testing prior to the face-to-face meeting. Make sure participants have software that can be modified and that they are sufficiently expert to do this on the day. Ensure that at least one attendee knows the standard thoroughly and understands its approach to architecture. If access to off-site servers will be required, make sure the TCP ports being used are not blocked by a firewall.	Experience shows people from sales, project managers and others who may be defensive, inhibit open and honest behaviour from developers, or make potential attendees fear the consequences of perceived failure on the day are expressly not invited to the working session. This applies to software suppliers, integrators and buyers. Event organisation should focus on necessary infrastructure and light-weight intervention from developer-savvy facilitators. Set the stage for developers to self-organise on the day and to feel a sense of ownership. A pre-meeting evening social can establish a good basis for collaborative activity. A neutral venue can be helpful to set the tone and to promote balanced attendance numbers. Similarly, a facilitator without a vested interest is advisable. Be positive and appeal to professional pride. The event is about solving problems and getting stuff that really works. Make sure the participants feel important. Emphasise that the standards-maker is desperate for feedback and have evidence that feedback will reach them. Try to make sure that there is a balance of data producers and data consumers both in the interest of effective testing and good morale. Make sure evidence is captured Develop some user scenarios to guide walk-throughs of data transfer, modification, further exchange etc. Although a public report on the findings is sensible in the interest of open process, the meeting should adopt the Chatham House Rule. A concluding plenary to summarise and agree the main findings may be useful to complement what may have been a free-flowing process.

We also made efforts to engage both private and public sector participation in recognition that both have a part to play in realising benefits in an environment where standards are a means and not an

end.

Private sector participants will generally be more concerned about intellectual property rights than public sector participants but the topic should always be explicitly covered; we took the view that copyright of all resources created for or during the Code Bash remains with the authors and that there was no expectation that source code would be shared. A public dataset of both “known good” and “known bad” items can be a very useful resource but it is important to gain an explicit consent to make contributed data openly available using a specified licence.

2.2 Limitations

Complex or highly variable data exchange processes are not really tractable using the approach outlined above. They are best avoided but a code bash dealing with more simple exchanges may open up channels of communication that are valuable in dealing bilaterally with complex cases.

The timing of a code bash matters. Code should not be frozen or require change control process. Conversely, when timed well, Code Bashes have the potential to take a lot of cost out of acceptance, arguing whose fault it is, change control, rework, regression testing etc.

2.3 Similar Terms

We called these "code bashes" but they go under many similar terms such as “plugfest”, which appears in the Cabinet Office Open Standards Principles as a glossary item.

The term “hackday” has also become quite common in recent years but this usually refers to an event to create something rather than to validate existing code/standards. We have used this format to prototype the XML exchange format given an existing draft information model. This kind of hackday may be useful to both subject a draft information model to some practical testing before much time is spent on fully documenting it, and to develop a first draft binding to XML, JSON, RDF/XML etc. This early intervention approach has the potential to bring forward some of the positive outcomes of a code bash and reduce the scope for inconsistencies which may be undetected in purely paper-based approaches. The whole process becomes much more one of collaborative design, while recognising that it will usually be most productive for a small group to undertake the specification writing process.

3. Further Information

More detail on the history and background on how Cetis ran Code Bashes may be found on Lorna Campbell’s Cetis Blog[2], which includes links to the original accounts from past Code Bashes. A paper submitted to the Cabinet Office on hackdays as part of the procurement process[3] includes a contribution outlining Code Bashes in this context, contributed by Cetis staff member Scott Wilson.

[2] *http://blogs.cetis.ac.uk/lmc/2012/12/05/codebashes/*
[3] *http://www.scribd.com/doc/114734608/Hackdays-in-Procurement*

About this White Paper
Title: Engaging Developers in Standards Development

Authors: Lorna M. Campbell and Adam Cooper

Date: October 2013

URI: http://publications.cetis.ac.uk/2013/871

About Cetis
Cetis is the **Centre for Educational Technology, Interoperability and Standards**. Our staff are globally recognised as leading experts on education technology innovation, interoperability and technology standards. Cetis provides impartial strategic, technical and pedagogical advice on educational technology and standards to funding bodies, standards agencies, government, institutions and commercial partners. Cetis is active in the development and implementation of open standards and have been instrumental in developing and promoting the adoption of technology and standards for course advertising, open education resources, assessment, and student data management, opening new markets and creating opportunities for innovation.

http://www.cetis.ac.uk