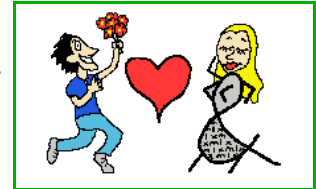


Xcruciate, an all-XML server

I'm Mark Howe, the project leader of Xcruciate. That means I got to pick the names. The rest of the team wanted you to know that. To find out what's behind the names, you could study our copious documentation and eyeball the code. Alternatively, back in the real world, here's a pithy summary.



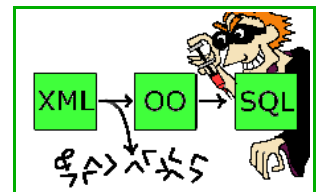
Maybe, like me, you have a love-hate relationship with XML server technology. You fell in love with the notion of technology-independent data representation. XML was well-formed, and made you feel validated. You told all your friends, and XML moved in.



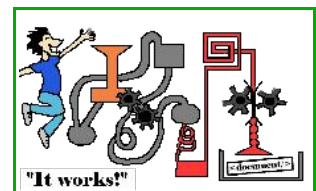
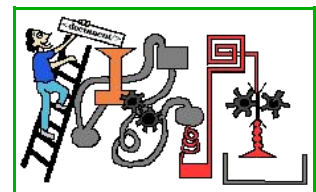
Then you discovered that XML had copied your apartment key and given it to all its friends, none of whom could talk to each other except via XML. You discovered that most of XML's friends spoke to you using their own non-strict superset of the DOM API.



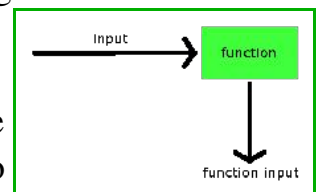
You used assorted DOM dialects to turn your XML into a homemade object-oriented data structure. There were plenty of methods, but it still seemed like madness. You stored data in a database, found yourself SQL-escaping XML-escaped text, and wondered if your data integrity would survive the experience. You realised that each translation introduced more scope for misunderstanding and downright criminality.



Then you repeated the whole process in reverse, often to end up with XML remarkably similar to what you started with a dozen translations earlier. You found that there were lots of proposed solutions, and that most of them involved giving a key to yet another XML technology.

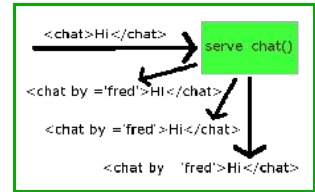


It wasn't supposed to be this way. Maybe functional programming provides a route back to that rose-tinted all-XML world. Imagine a stateless server as a function that takes socket inputs as parameters and returns its socket outputs as the result. In the case of a simple web page, the input is on one socket and the output to the same socket is a static value indexed by the input.

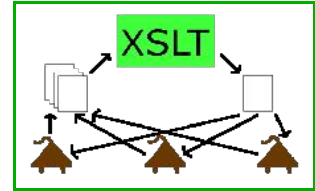


In the case of speech in a simple chatroom, the input is on one

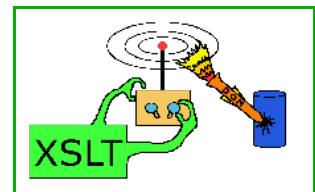
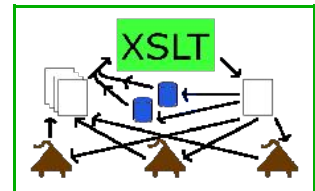
socket and the output is a slightly modified version of that input mapped across a number of sockets. The input and output can be described using XML documents, and an XML-aware functional language should be able to deliver the output on the basis of the input document.



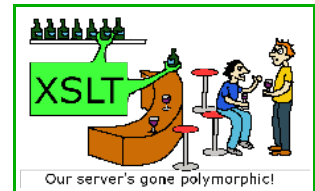
XSL transformations are at the heart of our all-XML server solution. We gather up socket input into one XML document, feed it into our XSLT engine along with a suitable transformation, chop up the output document and dispatch it to sockets as necessary, repeating ad infinitum. That, basically, is what Xcruciate is all about.



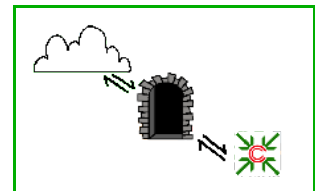
Most server applications are at least minimally stateful, so we needed a way to store persistent data. The most obvious XML way to do that is... XML documents. Conceptually, modifiable XML documents are included in the input to the transformation and modified according to the transformation output. Rewriting potentially huge XML files in order to change one attribute isn't very efficient, so we came up with a way for XSLT to pilot surgical DOM modifications via the output document. Then we cached the XSLT and DOM representations of those documents to speed thing up a bit more.



We then had XML-based I/O and data storage, with transformations specified in XSLT, which is itself an XML vocabulary. So what is to stop our transformation from transforming XSLT in real time? The answer turned out to be 'nothing', which opens up some quite exciting possibilities (as well as some frankly frightening ones).



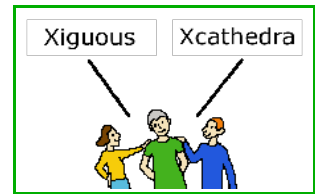
Obviously there are times when you need more than pure XML. That's why we wrote an HTTP gateway. It translates HTTP into XML, passes it to our all-XML server, and then turns the resulting output into an HTTP response. It can send email too.



All those ideas are embodied in the the Xcruciate project's current codebase, with Daniel Veillard's LibXSLT at the heart of our Xacerbate virtual machine. An early version of Xacerbate has been running a chat room for a year now, and the Xcruciate website is served by Xacerbate & Xteriorize, our HTTP gateway.



We have developed an XSL library called Xiguous for chat-type interaction, an XSL library called Xcathedra for website generation, and we're working on merging them into a cross-media content management system for social networking projects. Future work will include document validation and linking together multiple instances of Xacerbate.



That's the big picture. The detail is on the xcruciate.co.uk website. If, like me, you dream of a world in which reinvented open-source wheels have pointy parentheses, you might wish to consider using or contributing to the Xcruciate project. If not, thank-you for your time, and do give my kind regards to all those DOM dialect-speaking lodgers.

