Cincom Smalltalk User Group, Frankfurt, 7-9 Dec 2004

My economical and carefully-scheduled travel arrangements worked perfectly up to the moment when I arrived in good time to get on the bus from the airport into Frankfurt. It was full but the driver assured me another bus would arrive in five minutes, which it did; alas, it did not leave for an hour and a half. Thus I missed the first talk, in which Ron Weeks (director of Cincom's advanced technology group) welcomed us to the conference, after which Dave Wood spoke on Cincom's commitment to Smalltalk.

Style

In the text below, 'I' or 'my' refers to Niall Ross (as does Niall; I refer to myself in the third person when it reads better that way); speakers are referred to by name or in the third person. A question asked in or after a talk is prefaced by 'Q.' (I identify the questioner when I could see who they were). A question not prefaced by 'Q.' is a rhetorical question asked by the speaker (or is just my way of summarising their meaning).

Author's Disclaimer and Acknowledgements

This report was written by Niall Ross of eXtremeMetaProgrammers Ltd. (nfr@bigwig.net). It presents my personal view. No view of any other person or organisation with which I am connected is expressed or implied.

I've reported the talks I attended, the discussion sessions and two of the impromptu demo's (Andrew McNeill on WithStyle and Steve Kelly on MetaEdit). There were other demos and some 'Meet the Experts' sessions I have not reported. There were also some parallel talks I did not capture. Apologies to all those missed.

The talk summaries were typed while I was trying to keep up with and understand what the speakers were saying, so may contain errors of fact or clarity. I apologise for any inaccuracies, and to any participants whose names or affiliations I failed to note down. If anyone spots errors or omissions, email me and corrections may be made. Likewise, reports of what was said in general discussion were noted down afterwards from memory, so may easily contain mis-heard names, errors of fact, etc.

My thanks to Cincom and specifically to:

- the people who organised the conference (especially Monika Laurent and Helge Novak who, amongst many other things, handled the not-always-trivial admin of my arranging to give a talk :-).
- Michael Bany and Helge Novak (again) for generously sacrificing ten minutes of their talk so mine could finish despite its machine problems

and also to the presenters.

Summary of Conference Events

The talks are in very rough chronological order, followed by the impromptu demos, the user discussions, and general discussion notes.

Presentations

Cincom Smalltalk Product Strategy and Roadmap, James Robertson

(I missed the first part of this talk.) VisualWorks 7.3 and ObjectStudio 7.0 are released this month. From now on, there will be one major release and one dot release per year. James reviewed upcoming development tool improvements (Pollock, web, etc.) and what has been achieved (some 'before Cincom' to 'current' comparisons of what VW looks like).

VW in 7.3 is supported on more platforms: WinCE, PPC Linux, Sparc Linux, MS Terminal Server and Citrix. (Some of these platforms are not too standardised themselves yet; if you see any problems, please tell us.)

Q. Beos recently popular in Germany? The UI is major part of any port and Beos has an efficient but very different UI so this will not happen unless a customer is paying for it.

James also spoke about the Webtoolkit, XSDL support, WDSL tools, etc. RMI over IIOP will make it easier to interact with Java widgets. COM has been improved so you can embed e.g. Crystal Reports into a VW window (any ActiveX widget). The Browser Plugin is now back again. (It was lost when MS removed IE interoperability with Netscape years ago 'for security' :-) For their .net strategy, see

> http://www.cincomsmalltalk.com/CincomSmalltalkWiki/Cincom+Smalltalk+.NET+Plans

They support web services are working on .net events and WinForms. James mentioned OpenTalk (see Len Lutomski and Andreas Hiltner's talk for some details) and the ability to connect to IBM MQ. He also discussed 7.3's security features (see Martin Kobetic's talk at ESUG2004).

Store will have file attachments in 7.3 and atomic source loading (i.e. can use source, not binary, when changing protocols that need to be synched) in 7.3.1 or 7.4. There are more supported back-ends. (Q. Sybase? Goody on CD, not supported as yet.) James mentioned GLORP (see Alan's talk).

Bottom Feeder has made Jim more aware of application deployment issues. Over the next two cycles they will move to a smaller run-time image to which you then add things as the default deployment process. Stripping will still be supported but will no longer be the usual process. The network installer has been upgraded and you will be able to define your own installation maps. They will also provide support for the run-time update capability.

Tool support will continue to improve: better change tools and Store tools, wizards for web service tools. The multi-process UI is now in place and Pollock tools will appear. They have done and will continue to improve internationalization support. They now have a fully unicode-aware image for Windows, and have the same for Mac in preview in 7.3 (OS9; X is planned, along with Unix and Linux).

VW-DEV gives people access to the weekly builds, being able to use the

latest code (good for you) and exercising the beta code (good for Cincom). Jim gave an example; a change to Store that was slightly backward-incompatible that they avoided because the VW-DEV community caught it. Later, they will offer on-line bug tracking; for now, use the VW-DEV mailing list.

ObjcctStudio has the XML parser and OpenTalk. A preview of full unicode support is available. ObjectStudio now runs on MS Terminal Server and Citrix; they are investigating .net.

Q. Make known bugs more obvious, so people can quickly see if an issue is them or a base release issue? Good point; some are on the Wiki (which ones is determined by a support process). Kim will be talking about this, and taking feedback. We hope that online bug tracking will help this issue.

(Georg Heeg) Be aware of some very recent goodies (in 7.3 release).

- From Anhalt Univ work the localisation work (c.f. Georg's talk in StS2002), has 480,000 translations from English German and similarly for French, etc., using new collection class MultiDictionary (structured keys).
- WikiWorks for WebToolkit (WikiWorksForSSP)
- BOSS

CE UI is being done for VW, can use bluetooth, etc. is coming. Eliot Miranda mentioned that the 64-bit VM beta on linux is also on the CD. Jim showed BottomFeeder displaying Dilbert.

Q. VM for Symbion? Eliot first heard of this 2 months ago. We will be having a look at it. The GUI will be the major cost. The commercial CD now includes the VM sources. We may work with outside parties. We understand that Symbion is important on this side of the Atlantic; the US market is not quite there yet.

Q. Lens support? When GLORP is put in production we will provide translation tools, just as for old GUI to Pollock. This is more than a year away.

Q(Roel). Goodies? The author must update and put in the Cincom Open Store. The latest version gets pulled and put on the CD. Roel has done a little work on something that would pull all the prerequisites and put them in a single parcel but he lacks time to finish it.

Building Web Applications in Smalltalk, Alan Knight, Cincom
Alan has been with Cincom for 5 years. Previously he was with the Object People and was chief engineer of TopLink.

Alan spoke about tips, headlessness and new features. His subject is conventional web applications, not web services. The visual works web application server has two technologies, visual wave and the webtoolkit, plus some technologies above.

VisualWave was one of the first web applications frameworks, automatically generating HTML from VW application windows. It still works but has not tracked many more recent changes to how things are done on the web. The webtoolkit tracks MS .asp, Java .jsp and servlets, to let you do the same things in Smalltalk. It shares infrastructure with VisualWave and increasingly with OpenTalk. It aims to be the same and better, i.e. to meet standards but to exploit Smalltalk's advantages to be simpler, more robust and more debuggable. Alan likes to demo putting `self halt` in a page and seeing the debugger pop up (impressive in itself to those who use rivals) on three lines of code versus looking at the 5 pages of code that e.g. WebSphere would generate and trying to find the relevant problem-causing line in that.

7.3 adds more on internationalisation, scalability, gateways and headlessness. It enhances international character support. It separates locale and charset, which VW previously tied together but which the web does not tie together. Web standards are not good in this area. International characters in domain names are not the same as in URLs, etc.; how to encode these? A URL with a euro character in it is an issue; there is no standard for that but the emerging standard is utf-8. However if you click on a URL it uses the encoding that the page uses, which seems sensible but there is no way the page can tell the server what encoding it was using. Hence they use a heuristic

- try UTF-8; if it does decode, can it be resolved?
- can we find a session; does it have an encoding?
- use the default

This is configurable so that e.g. in Japan you can provide sensible guesses for your probable clients.

This only works for URLs; for form data you just have to know the encoding. The internet explorer often does not honour the request_charset if it finds it can encode them in e.g. the MS default, so you may have to have a hidden field with characters it cannot encode to avoid this.

Gateways: pure Smalltalk works fine but you may want to use Apache or whatever so non-Smalltalk apps can run on the same port, so that you can use authentication (e.g. hold all your certificates on Apache, not duplicate them to the Smalltalk server). In 7.3 they offer a perl script to make it easier to route to Apache; it makes debugging easier and is fast. Alternatively, you can proxy to reroute the URLs and have Smalltalk serve them. They have updated ISAPI to stay in line with MS changes. NSAPI and FastCGI are deprecated; FastCGI is much more complex than the perl gateway and slower.

You must tune your memory; the defaults will surely be wrong in some of your web app's cases. 7.2.1 and 7.3 change the default web memory settings to have a very large (250 times old default) largespace with lower (160Mb) upper bound and tolerating high growth within that, i.e. less aggressive garbage collection. (7.3 adds some socket buffer reuse which may be able to reduce largespace considerably but has not yet been

optimised.) *Be aware* that when you load webtoolkit or VisualWave you get these settings installed by default; revert them if you are not building or running a web app. And always test.

Load Handling: it is better to serve fewer connections better than many badly; your browser will time out and retry so fail early. Within an image you can set maximum connections, process priorities and socket parameters such as the backlog size of socket connections you will listen for. Two load balancing applications are supplied, an existing one now adapted to use OpenTalk and a new OpenTalk-oriented one that is not yet adapted to web applications.

Headlessness and subsystems are relevant to building web apps. In 7.3 shutdown has been refactored. Previously big tightly-sequenced methods in ObjectMemory did everything. Now there are a number of subsystems each of which can be activated and deactivated. You no longer need to be a dependent of ObjectMemory; you can be, but you can also work more fine-grained, specify prereqs, use command-line options. (And you can subclass `UserApplication` and implement `main` and be sure it will be called on system startup if that's the kind of thing you're into. :-)

The code to make headless / headful is simpler

```
"to become headless"
  WindowingSystem canActivate: false.

becomeHeadful
  WindowingSystem canActivate: true; activate.
  ...
  "just a bit more complicated than the above, actually"
```

Command line options (headless, headful, pcl, filein, doit, evaluate - run this Smalltalk code with no GUI then quit, settings - XML settings file to be loaded) are useful (and also dangerous; you can turn them off for deployment).

Georg Heeg has done a framework for VisualWave. Seaside uses advanced language features of Smalltalk to build webapps; see Avi and Michel Beny's talk. WithStyle is an XML and stylesheet rendering engine made by a company in Australia. (Andrew McNeill from Cincom Australia will demo it and has demos of it.)

Q. (Georg) various associations (village football, local zoo, etc.) want simple web app - I can edit the app at home but it looks good. Jim: withStyle would let them use XML template and edit the content. Their product, easyXML, is being released shortly and is already in use. An Australian government site uses this and finds it better than XMetal, MacroMedia, etc.

Q. Mixed languages: Russian newspaper in Stuttgart has problem with Russian and German in same page? Encoding within page should be easy (maybe hard in authoring tool) as the server should not care. (More discussion off-line.)

ObjectStudio, Len Lutomski and Andreas Hiltner, Cincom
This was a demo of ObjectStudio 7.0 and VisualWorks 7.3 talking to each other over OpenTalk. OpenTalk-STST is a production-quality Smalltalk-to-Smalltalk communication protocol, built on OpenTalk, that lets these versions (N.B. not earlier ones) interwork. More generally, ObjectStudio can now write and run multi-image applications, the multiple images including VW images running on Linux, Mac and other hardware that VW, but not OS, runs on.

Andreas then demoed a chat application talking in OS to Lutomski running VW on a Mac connected to Andreas' machine. They use this chat application internally. It exercises the framework but just shows objects and strings being exchanged. Len then demoed OpenTalk in general.

OpenTalk uses

- Request Brokers: `newStStTcpAtPort: 1902` (or specify address as well if your machine has multiple n/w cards). Look in the configuration code this message runs to see marshaller, etc., being set up and how you would configure by hand if you wanted to (e.g. you might want to have `scReuseAddr: true` since some hardware will not let you reuse a socket immediately if you explicitly close it but you may e.g. save and want to resume work immediately). On VW a process is a Smalltalk process but in OS a process is a host-level thread so for apps that will create many, you should always configure pool dispatch in OS.
- Initial References: you need to start two brokers, one on each side of the communication, and give them references to each other. At some level, an initial reference must be programmatic. One way is for the server to use `export:oid:` to export an object so clients can see it. The client calls `remoteObjectToHost:port:oid:` to acquire the object. The other, which is slightly the preferred method, is to use a broker's service registry. `ServerBroker registerService ...`.
- Pass Modes: the way in which an object is passed from one broker to another. These affect performance so several must be supported. Pass by reference and pass by value are the two obvious examples. Value costs up-front, reference is cheap initially but every message to the passed object must be sent back. There are various methods, class passMode, asPassedByReference/Value/Name/OID, passInstVars.

These concepts have obvious fundamental similarities to those of CORBA.

A protocol implementation has four parts

- the adaptor maps ST messaging semantics to the protocol type
- the transport layer (e.g. TCP)
- the marshaller translates between the objects and their encodings on the transport layer
- the dispatcher handles the incoming messages, sending them to the worker process (standard approach is to fork a new process for every incoming request)

Brokers respond to `start`, `stop`, `ping:` (replies 'Contact' or 'Nobody Home'), `echo:` (send object, remote broker remarshals and returns, echo checks object returned was same as what you sent). Some things, e.g. fractions, are passed by value but others are passed as references and will be resolved to the identical object when echoed back.

Len demoed overriding the pass modes, passing objects by value and by reference and showing the different state.

Q. (Georg) can you change pass modes for special values: True, 2, etc.? Yes, you can get the 2 of the other image if you want.

Cross-Dialect Issues: the documentation gives all the differences and you should be aware of them; some can surprise you. Len demoed connecting a VW client to both an OS and a VW server to demo some differences. On each server, he created a dictionary, added to it a value holder, a very high precision number, lots of bags, arrays, etc., and then compared them to the dictionary built by the same code on the client, thus proving that the base classes map perfectly but that Magnitudes have 3 issues.

- OS only has double (greater precision than float); they always preserve the greater precision (so end up comparing float and double in test)
- OS and VW differ slightly in how they represent a number that is higher precision than they can handle
- Fixed points are internally held as slightly different fractions in OS and VW

Lastly, the same error may raise a differently phrased error message on the two systems. Len repeated that you must read the documentation if you are building real systems.

They hope that other Smalltalks will adopt OpenTalk. The simpler remote debugger tools for VW should also work. The main porting issue would be the socket layer and process model. OS (which had a very different socket layer) took 4 weeks times three people.

VM Plugin Framework for VW, Dr. Sudhakar Krishnamachari, Cincom India

90% of the operations use 10% of the code. If that 90% included e.g. some heavy floating point operations then it would be best to code it in Smalltalk but then use a framework to execute compiled C. This framework is a port from Squeak slang, done by Eliot Miranda and Sudhakar Krishnamachari.

VM has compiler, engine and primitives; these last are the focus of this talk. The user can supply primitives which are modelled in Smalltalk. The framework is contained in the HPSVLM parcels: these have the namespace and the plugin classes, `InterpreterPlugin` and `InterpreterProxy`, and the API, which is just C Slang.

A subclass of `HPSVLM.InterpreterPlugin` is where you write the primitives. All testing and debugging is done in Smalltalk. When all is

done, you export this class to a C file and compile it. The default parameters and return types are int but can be coerced to what you need.

He showed example code on the slides and then demoed 'Hello World' (the demo is on the conference CD). There are similarities to DLLCC (subclass `ExternalInterface` ...). He walked through it in the debugger, showing how the development could be done in Smalltalk. Then call `VMMaker` to export as C. This done, you see two directories, one with the generated external plugins and one with whatever plugins you have created.

The result is that you handle Smalltalk objects in C code much as you would in Smalltalk. The 32-bit pointer is sent from Smalltalk to the C code, holding the class oop, size, hash and indirection pointer. At this point while debugging you can check (`isIntegerValue`, `isKindOf:`, ...). When the indirection pointer is mapped to the instVars you can call `fetchWord:`, `fetchPointer:` on named classes and the equivalent on arrays (most common) and other indexable objects.

`InterpreterProxy` has a range of methods to handle type coercions, object data (size, etc.), special objects (e.g. C NULL not equal to Smalltalk nil, get the safe object to use via a method call). You can also write

```
self cCode: aString inSmalltalk: aBlock
```

for code that you know is safe in C, has no slang equivalent in Smalltalk, but is adequately represented by the block, so that while testing in Smalltalk the block is run but in C the C code is run.

He then demoed a more complex example: the vector maths example in DLLCC. He showed the primitives used (results are returned as a parameter, as is normal in C, rather than as the return type) and showed the resulting C. The conference CD contains this example and a set of tests that exercise it. The plugin speed is four times faster than Smalltalk on simple vector addition and is also faster than in DLLCC as the latter has more ramp-up time to pass the objects. By contrast, Smalltalk reads simple bit maps efficiently and no performance benefit was seen reading one via this mechanism. A bit map with complex colour, etc. showed a 4 times speed up on a 1.2 Mb jpeg. He showed the performance profile with the colour conversion figuring prominently.

He then demoed the fast fourier transformation (raw port from Squeak); there was a 50 times speed up (100 ms vs 2 ms) for his example.

Besides the performance gains, we may hope for other benefits:

- reduce irreducible VM executable size by splitting out optional elements
- fix bugs in plugins without changing the VM
- encourage modularity, both in implementing primitives and in using them

This is currently a goodie on the CD; it will be released next year.

Q(Georg) this syntax is like ObjectStudio's OSPrim. Why not use that? (Eliot) It is Squeak's syntax. The work was originally done to get Warpblt working in VW. The aim is to keep code compatibility with Squeak (modulo a very few issues: Squeak does not have immediate characters and Squeak GC can run inside a primitive so must push things on a stack whereas VW, wisely, disallows this). Now we have this, all the Squeak plugins can be used.

Q(Georg) callbacks from primitives back to Smalltalk? You can use all DLLCC framework in this so can use standard DLLCC framework. (There was also discussion of .DotNetConnect which I missed.)

Next-Generation Database Mapping Strategies and Dialect Interoperability, Alan Knight

VisualWorks has Lens, now aging, ObjectStudio has POF, early and capable. TopLink is now only sold for Java but the original product was in Smalltalk and copies of that code are still in use, etc. Some of these frameworks are old, client-server oriented (i.e. not web-oriented), not maintained or even no longer sold, etc.

Cincom Smalltalk has the long-term goal of a next generation DB mapping library with easy migration from, or backward compatibility to, existing solutions. GLORP is the core layer for this.

Short-term, GLORP is usable but bleeding edge (no documentation, etc.). No framework is finished till it has been used to build 3 real applications. Alan is on his first; the backend to Store.

GLORP was an open-source CampSmalltalk project sponsored by TopLink just before they ceased to be. The licence is LGPL(S), where (S) meaning licence rewritten to be meaningful in a Smalltalk context. It is portable between Smalltalk dialects (Squeak, VA, OS, Gnu, Dolphin, ST/X and VW); VW tends to be where the work is done but they try to keep it dialect-neutral. They have a Dialect class to encapsulate this, and a fileout parcel to strip namespaces and produce fileouts for non-VW dialects.

GLORP aims not to restrict what you can map (e.g. map any 300 classes to any 300 tables without changing either). It is meta-data driven and has object-level transactions.

Alan then demoed the store replicator. John Brant wrote the original one. Alan refined it. The SQL being generated and run scrolled up on the transcript as he viewed top level packages, then all packages in his (Envy-reminiscent) UI. He inspected an object to show it holding some data as StoreBlobs, only retrieved if needed (e.g. package comment so held). He showed it retrieving various packages with their data, then showed browsing unloaded package versions in the Refactoring Browser (which he has extended to do this).

He then did the same in ObjectStudio, inspecting versions and browsing them. The UI in OS is cruder and slower as he has not yet optimised it but

works OK.

Alan returned to VW and looked at a load of Store package descriptor* methods. He showed how the basic code was very simple

```
methodMapping :=
  (aDescriptor newMapping: ToManyMapping)
    attributeName: #methods;
    referenceClass: StoreMethodsInPackage
```

with all else being optimisation to prime buffers, bring back additional data, e.g.

```
methodMapping alsoFetch: [:each | each definition].
```

brings back only the data you want, e.g. just the package name. FilteredReads let you get the extra data so you no longer see e.g. first column being populated for a thousand rows and then the next field being filled in clunckily row by row. Conversely you may want to just see row by row filling, not wait till all million rows suddenly appear.

When you write, the framework works out the order and referential integrity, the transactions, the grouping of tables (helps avoid latency problems). A cache of prepared statements reduces the overhead of dynamic SQL. You can specify a field (some version number, timestamp or whatever) that will throw an exception on write if it has been updated (done by a Where... row count check), so you can use optimistic locking.

Envy used to be a Smalltalk CM format that everyone could access. Alan hopes that Store will become a replacement. To use it in another dialect you need translation between that dialect's CM (e.g. Envy) and you must also have BOSS support (which Alan ported to VisualAge and found easier than he thought.

GLORP needs stored procedures, more thread safe classes, so fewer sessions, and connection pooling. The error messages, validation and documentation need improving.

Store could handle dialect differences by using the RB rewrite engine to convert them; John Brant has done Store in Dolphin work using this approach. Alan wrote a proof of concept tool that creates HTML pages for packages that have comments and have been updated within a year.

Q. Run on multiple database types? Yes. You can connect to multiple databases. You can switch the meta-data on the fly too if needed.

Q. Can we download the ObjectStudio port? Not now (he was working on it on the plane coming here) but it will be available soonish.

Support to the rescue ... oops ... resolution, Kim Thomas, Cincom
Kim Thomas has been with Cincom systems for 11 years. She worked at first on Mantis, then moved to ObjectStudio 7 years ago. She has taught the intro classes for ObjectStudio. When she started support she wanted to be

the one place where people could get the answer that fully satisfied. In fact, support is about providing resolutions. Rescue is to save people from harm or evil and they prefer to think that's not the usual experience of their products :-); they aim to resolve the issue when you contact them.

Kim checked our knowledge of various terms.

- Case (lots knew it): the overall object that holds the overall customer's problem report, the customer data
- AR (lots knew it): for VW, the overall object holding a problem being analysed
- FR (lots knew it): as AR but for ObjectStudio.
- Resolution (only one knew it :-): resolutions contain information about patches, fixes, workarounds or patch builds.
  - Patch: (usually VW)
  - Fixes: customer-downloadable fix that is a full resolution, fully checked by developers.
  - Work Around: an un-evaluated solution or a customer-supplied fix made available to other customers.
  - Path Build: if a rebuild of VM or DLL is needed to provide the fix.

There is more than one way to receive support from Cincom.

- VW-DEV: ongoing beta testers, loads available to members
- vwnc mailing list: for non-commercial customers (but commercial also use it), run on a non-Cincom community servers, support resource is the smalltalk community, including Cincom employees who can generate ARs from what they see.
- Cincom corporate support: commercial customers

You enter a case via http://supportweb.cincom.com/support/centers.asp (the supportweb) or by calling one of the four regional support centres. Sometimes you may also call support people directly if you know them.

The customer assigns the severity to the case. Kim showed what support staff see when a case comes in. They can see what other cases exist, the status, the short description, etc. The notes page (standard web email layout) documents interactions with the customer. Notes are important to track what is happening with the case: its analysis, its resolution.

Support levels:

- A is a current fully supported release: solve to provide resolutions.
- B is an active release and has almost the same support level as A.
- C is an active release for which support can provide existing solutions. Other requests may be billed (time and materials).

- D and E are inactive release, F is discontinued: you will be billed time and materials for any support (and you will be encouraged to upgrade).

You can assign cases severities:

- 1: production is down; respond within 15 minutes
- 2: production down, urgent deadlines looming; respond within 1 hour
- 3 and 4: the problem is annoying but you could work around it; respond within 7.5 working hours (i.e. one day)

Support starts by understanding the problem, including its impact on you and your time frames. They then gather information to recreate the problem. Then they look for an available resolution and/or a root cause. 75% of the time support can resolve issues within itself. When they have to go outside, they lead the resolution effort.

When 3rd party vendors are involved, they turn things over to them, e.g. a they would pass a TestMentor problem to Silvermark, a goodie problem to its provider, but they will still monitor progress and contact you on the appropriate schedule. They also create the components of the resolution: the fix itself, the how-to information and, if the immediately fix is only a work-around, the AR or FR to continue resolution. You can view resolutions and support cases on the web.

They do also offer an escalation process: you can contact support for this if for some reason the normal support process just does not fit the immediate need. That has happened and you can ask support for this if you have reason for requiring a different approach.

She showed statistics to November 2004: 700 new cases received, 650 closed, 300 open in various states (case is never closed until fully resolved). Recently, the support group is doing more to publish resolutions. They have created a Cincom Smalltalk Customer news and encourage you to join that and to talk to each other. Customers' appropriate technical staff should be on this newsgroup (email Kim to be added).

The support group is doing training; support is not just like development. They are eager to reduce resolution time by adding to their resolutions database for you to scan and also to make faster use of customer-suggested fixes. They will also be sending out customer support surveys in all areas (only been done in America till now).

Q. Make known resolutions visible to all? Not all resolutions are visible to all customers. (Niall: follow-up) make *problems* visible, and whoever sees them would then contact you if a resolution was not shown. (Steve Kelly) I agree; I have sometimes spent a lot of time studying a problem to work out if it is an issue in my code or in the VW base, only to learn later that the symptoms were already known as an issue in that release.

Q. They can solve issues internally but then how to tell Cincom so it is resolved in the next release? You should report it and support should then use the standard process to do that part of resolution; Kim notes that it

should be made more visible when a fix provided to (or by) a customer causes a problem to be not (or no longer) discussed by them in the community although it is still potentially there for others to encounter.

Q. When you support old releases, can you encourage people to migrate, e.g. by telling them of people who have done so, telling them of resources, quote a few metrics? Kim noted the usefulness. Andreas mentioned that in Europe they often mention Georg Heeg's availability to help. (Alan) if papers on porting experience are published, we could point people at them.

Q.(Heeg) there are decade-old cases still around; the customers must have workarounds. Some cases, e.g. COM case, naturally take months of work.

Helge began to introduce the next talk and then recalled he had not thanked Kim; this is typical of the thankless work of support. :-)

Scrum, Joseph Pelrine, MetaProg
Joseph deals with real work issues of in-time, on budget on spec, and now has the problem of starting twenty minutes late. Joseph gets things done on time by having a strong meeting culture, so asked us re whether he could steal 15 minutes of our coffee break (and was invited to do so).

Corporations can be laboratories for new ideas or museums for tools and techniques. This affects how easy it is to get new techniques adopted.

Social complexity theory in 3 minutes: Ralph Stacy (researcher at Univ of Hertfordshire) graphs certainty and agreement (so at one axis point we agree we're not sure, on the other axis point we're sure we don't agree). Certain agreed things hardly need management. Beyond that, we get into politics. This axis is where agile is today: we all agree that waterfall does not work but lots of people are offering different solutions.

By contrast, eXtreme Programming is today on the other axis: we all agree but are not sure how to do it. If you do XP exactly according to a white book by Beck you are not doing XP. A comically extreme example of this occurred in a large consulting company who outsourced their XP to fresh-out-of-college people in Sri Lanka. The project was failing (it was in Java) but passing its tests. Kent's book says write test cases. Nowhere in the book does it say that you must assert anything in your test cases. So they didn't.

Joseph sees coding as a heartbeat oscillating up and down in this social complexity graph, trying things, finding patterns and so regaining certainty and agreement. Example: are you using the last stable release or working on the nightly build from sourceforge.

- On using techniques from an ordered realm in an unordered realm: you will *never* make them work.
- On using techniques from an unordered realm in an ordered realm: you are stealing scalability.

So you have to deal with real people: Harry is a poor coder but makes

predictable testable mistakes, Fred is a good coder when he shows up; if only he would test his code more. (Some in the audience thought they knew these people; others wondered if they were these people. :-)

If you know enough and know how much waste you are prepared to accept then a defined theoretical process will work. If you do not, an empirical process will work. Empirical processes are often far simpler than defined processes. Birds fly by an empirical process. Software development is an empirical process. Wat Humphrey (CMMI) says the software is an empirical process. Give competent people a vision and leave them alone.

Beck: "To be a consultant, you need to know that it is always a management problem, usually a communication problem and may be a technical problem." There is often incongruence between the type of company a business needs and the management it gets. Miller says companies, like software and like people, have a lifecycle; they grow up (some just grow old) and at every age they need a different kind of management. Startup managers are not right for companies of 200,000 people:

- prophet: creates the breakthrough
- barbarian: leader of crisis and conquest during rapid growth
- builder: shift from command to collaboration
- administrator: shift from expansion to security
- bureaucrat: crucifies and exiles prophets and barbarians
- aristocrat: inheritor of wealth, alienated from those who do work, cause of dissatisfaction and rebellion
- unifier: (sounded like a rare type)

Choosing two weeks as the length of your iterations is fine either to try it or because you've found it good. Don't choose that because Joseph says so. Sometimes you need to move people out of their comfort zone as the only way to get them to optimise their processes (picture a sumo wrestler evolving to a leaner wrestler instead of to an immobile structure). Defined processes do not scale. Empirical processes work well until you are well into chaos.

Scrum is easy to understand but hard to do. Jeff Sutherland did one of the first Scrum projects (for a firm who built case tools to help people build s/w on time and budget but could not get their own tools done on time). Joseph taught Scrum to a customer who took 1.5 months to inboard it and install a 24 x 5 development process. They do configuration software for mobile phones: Nokia released some new work 2 weeks early and the customer got a $1.2 million bonus. Microsoft use Scrum on .net. The BBC uses Scrum.

Scrum sprint time is 30 days (maximum time management can be induced to leave developers alone). Scrum has a vision and a product backlog: things needed to reach vision. A selection from the backlog is assigned to a thirty-day sprint. The goal of the sprint is to deliver *functioning* software (essentially, value to the customer). The team meets every 24 hours for a

very structured planning meeting. Scrum is readier than XP to think ahead about things not needed yet; put them on the backlog and you will be aware they will appear sometime.

Every project needs a scrum master who is the coach. Every day there is the scrum meeting: no making a cup of coffee just before the meeting and you pay a fine (starts at £1, one euro or whatever) if you arrive late. The meeting is in same time and place every day, everyone answers 3 questions only: what did you do yesterday, what will you do today, what problems or blockers are you encountering.

Scrum meetings divide people into chickens and pigs re the issues being discussed, meaning who is committed and who is simply involved (from the old joke about the chicken and pig opening a restaurant. called 'Ham and Eggs'; the pig is committed, the chicken is only involved). Scrum says, make this explicit. Scrum teams are self-organising and role-free; 7 +/- 2 is the ideal size. People are responsible for committing to work and the team must have authority to let work be done. One person is the product owner and decides what from the backlog is done (somehow the CMMI conformance work always seems to drop off this list. :-)

There is a sprint planning meeting at the start of each increment. Sprint projects track work still needing doing, not work done.

XP and Scrum combine naturally; they are complementary technologies. (It can also be defined to meet CMMI level 3; Joseph knows of 6 Scrum projects that have been so certified.)

Q. Sometimes a company mandates a single process across the entire organisation, or otherwise makes wrong decisions? Scrum's impact will be visible; some are uncomfortable with that. How to handle this is beyond the scope of a software development method but it is wise to be prepared for it.

XPerience eXtreme Programming, Vassili Bykov and Joseph Pelrine
Vassili has been working on tools since VW5i.3. Joseph and he aim to show that programming can be fun. Without VisualWorks and Envy and GemStone (and Don and John doing the Refactoring Browser), eXtreme Programming would never have happened.

If your customer could only pay you for one day of software development, what would you do differently ... especially if giving them something they could earn money with that would let them pay you for another day.

[Eliot: 'The drug dealers approach to XP' :-) Joseph pair-programmed (paw-programmed - Eliot's cat was on the monitor) with Eliot once with the DNU, program in debugger trick. Eliot delayed to have some chips and that was the only time Joseph ever missed a flight :-).]

When is development finished? (Eliot: when the customer has given you all their money. Joseph: that's the psychiatrist's approach to XP; you are healed when you have given the shrink all your money. :-)) When it is no

longer financially beneficial to develop further. Joseph showed a slide (taken from original napkin written by Kent at airport, with beer mug stain carefully removed :-).

The waterfall method starts with requirements and it is all downhill from there. The phase when you will truly discover errors is at the end, the worst possible time. XP says, define your requirements by writing tests. You are done with that requirement when the test is done. If you add too much complexity to a system without cleaning up, it reaches chaos, so you must refactor.

People have different meanings for the word done; when a developer says, 'that task is done', what does that mean. Climbers (Joseph's sport) clean up all the pitons they've hammered in on the way up as they descend. Likewise you should clean up code. A code review is best for this. The best code review is the one that happens while you code: pair-programming.

That is the first circle of XP: testing, refactoring, pair-programming.

The next circle is the team one, when there are several pairs working on something. This circle is responsible for coding standards, for collective code ownership and for continuous integration: always have a working functioning baseline.

IntelligentViews in Darmstadt (too busy working to be here today) wrote the coast environment,. K-Infinity, etc. In Envy, Joseph can go from repository to CD in single step; Store is a little more involved. However Joseph showed loading completely from scratch and creating a build of the UML demo in Coast, after which res-hacker tweaked the icon. Then he just double-clicked the icon and the demo started; that is what Joseph means by continuous integration; be able to do that *entire* process.

The next circle is the process circle, about the team rate of work. Joseph once did weight training. It increased the weight he could lift but the recovery time between lifts did not change more than 10%. The 40-hour week is what XP recommends; Joseph suspects 30 hours (6 productive hours per day) may be more appropriate. Hours are not a good estimating unit. Divide the day into its four seasons: before coffee, after coffee, after lunch, evening. 'This can be done by lunch', 'this will take all day'; these units of time mean more to people (and will prompt informal coordination). For Joseph, Winter is before morning coffee: you are cold and just starting. Spring is after coffee: your are raring to go and making process. Summer is after lunch: warm, confident, perhaps a little lazy. You can work in Autumn, but you are aware things are running down. (Other teams might have different assignments.) If a task is hard, consider doing it in your springtime, not in winter.

The last is the planning circle: do the planning game, do small releases, seek an on-site customer.

Then Vassili and he worked on a task: a manager of table of data for

courses, etc. Vassili wants to map objects from a relational database to XML Schema. He plans to call his utility XOMBIE (XML-to-Objects Mapping Bridge, Impeccably Engineered).

(`NamedFontSelector open` lets you change style properties to do more than just large-fonts, small-fonts. Alas there is no way to change the keyboard so that it would have Swiss layout for Joseph but U.S. layout for Vassili; I saw this as another example of the value of screen-sharing tools.)

Joseph began by creating a class Person (some, e.g. Sames and maybe Eliot 'what are undeclared variable for' Miranda would have written the test before this.) Then he created a test class and wrote his first test, which passed. This is bad; the test did not give you any additional information. When writing tests for new code, not legacy, never write a test that (you expect) will pass. This is one answer to the question, how do you write a test that has value. He quickly extended the test to something that would fail, at first on DNU, and fixed it 'accenture-style' by providing a method that did nothing. Now the assertions fail (with useful message as they wrote an assert:description:). Once the test passes, write another test (and make it fail). Vassili took over here (commenting on Swiss keyboard :-) and they began to read the XML and extract Person instances from it.

The point is simply that Smalltalkers write tests all the time in workspaces. Like M. Jourdan, who discovered he'd been speaking prose all his life, all Smalltalkers who use SUnit discover they've been writing tests all their life. SUnit organises the code and its comments into tests and description strings, helping both you and later users.

Joseph then showed logging from SUnit. His utility produces an XML file. In SUnit 3.2, TestFailures and TestErrors are full state objects with more context, so can report more when logging. The format is the same as that used by ANT (in the JUnit report task; use Java's tools against them :-). Joseph showed running ANT test report on a Smalltalk test suite.

VW Tools, Vassili Bykov, Cincom
(Vassili slides are just for illustration; the talk was mainly demos.)

Tools development must handle the three areas of legacy tools, new development and third party tools. Vassili sometimes feels like he has a single hose and is standing between three burning houses, running from one to another. :-)

Vassili showed a 'paste what I mean' feature during demo; code text just appeared when he wanted. :-) Cut and paste on a multiple selection in an inspector on a collection does the obvious thing (as does undo), and you can drag and drop between them (even arrays !).

Pragmas are anything in angle-brackets or, more usefully, are things that change the execution semantics of the method. <primitive: ...> or <C: ...> prevent the subsequent code running but do something else. By contrast, a <menu: ...> does use the following Smalltalk code (and is Smalltalk code -

like literal arrays, its params cannot by dynamically changed).

The pragma populates an instVar 'attributes' of the compiled method. You declare pragmas via `pragmaDeclaration` on the class side (if you forget this you will see a warning). Pragmas let two applications add menu items to an existing menu, an example of a more general problem of a resource that is a collection of items (e.g. menu items) where you must separate the code that provides the items from the code that assembles them into a single 'collection' object. Methods `instanceMethodsChanged` or `classMethodsChanged` are sent when a class' pragmas are changed. Class Pragma has methods for finding all pragmas.

Q. (Joseph) could use this to assign tests to methods (c.f. NUnit annotations)? Vassili agreed.

Vassili demoed pragmas by creating a tool to browse classes and then using a pragma

```
<component: 0 class: #{BrowserOpener} spec: windowSpec>
```

to add it to the window spec of the launcher. He then added a `CompletionDriver` (watches input field, tries to complete what user is typing) to it and showed typing auto-completed class names.

Next, Vassili showed creating an `IncrementalSearchDialog` to match not Smalltalk classes but the (unusual :-) favourite dishes of the Santa Clara development team. Use

```
(self forRequestWithSuggestions:...filterBlock:...)
  firstLabel: ...;
  secondLabel: ...
```

(`forClassSelection` does this for standard case). He next showed a two-pane dialog, left as `IncrementalSearchDialog`, right showing detail of selected item. (Ctl-E-enter to get a browser on class whose name is selected. PDP and other tools have 'go to class', etc., new items.)

Vassili dislikes the API of the current standard list block, so has created a class `ListModule` that offers the 'right' API. See his slide for architecture of dialog: `ListModule`, `EntryModule`, `IncrementalSearchModule`, `IncrementalSearchDialog`. The user sends messages to the `IncrementalSearchDialog` but in fact many need to set values on the internal objects, which should not be exposed. Rather than delegating method by method, he has a delegation pragma. Whenever the `IncrementalSearchDialog` DNUs, it sees if a specified set of accessors return objects that can understand the message.

The settings framework can be used to define settings in your own framework, or new settings for VW. (Info on Cincom Smalltalk Wiki about how to do this.) `tools25SomeNewSetting` with suitable pragma will add a new setting on Tools setting page in settings tool (25 controls placing; not the ideal way of doing this). Vassili wrote a deliberate error in the pragma to show debugger coming up, saving and then you did not even

have to proceed; it was already fixed in settings tool (`instanceMethodsChanged` response).

(Suppose you want to inspect a window: CTL-Y, third frame, window manager, find window you want. He showed seeing the hierarchic decomposition of all the UI wrappers of a window in Trippy, thence do inspector-based programming by editing methods in the Trippy pane.)

Vassili has built some tools using Pollock, to test Pollock and to understand how it would work; these tools make up his demo. Pollock is in feature set 1 (not to be confused with production set 1), and Vassili is giving feedback to Sam.

First he showed fly-by-help. Try fly-by-help in adobe (go to button, see help, go next, wait a bit, see help). In VW, he showed fly-by-help (i.e. tool-tips) showing immediately as he moved from button to button. This is done by a state model. Start in state Cool, can go to Armed (ready to pop help on a widget). If the user clicks widget or leaves it, back to cool, else after 0.5 sec go to state Reaper (show the help). After 5 secs, close tooltip and go to Cool. However if user leaves widget while in state Reaper, go to Warm state, when help is popped immediately, thence to Cool on another timeout.

Vassili walked through code in `FlyByHelpTracker` that calls the `cautiously:` method. This cleans up if an exception occurs so it resets the semaphore and state; no dangerous moments for meeting an exception. He showed tooltips on a class hierarchy browser and browsed `ToolBarAgent` that implements the state machine described above, in `buildTooltipMachine`, `setUpTooltipMachine`. He uses a new `componentAdded` event as he needs to know when buttons are entered so must know when new buttons are added to the toolbar. Although we could keep a reference to the tooltip machine, the code all works with it just in a temp. When `buildTooltipMachine` exits, the machine is not GCed because it has interest in events. He mentioned the overlay machine; Sam and he plan for Pollock to support `zoneEnter`, `zoneExit` events.

`EntryAssistant` is a Pollock tool that watches as the user types and helps them. He built a Pollock tool with a text window and gave it an entry assistant whose suggestion block tries to complete with the name of a store pundle. In `CompletionDriver>>uglyShowCompletion: aString`, the method is contorted by Wrapper's needs, where the `changed` method is a lie; the view still has to update the selection point and so on. He has to wait on a timeout until this is happened. In Pollock, there is a sane event to use: `processedKeyboardEvent`. He showed the (very fast) completion.

He then showed putting the suggestions in a popUp window. This uses the `aboutToProcessKeyboard` event in Pollock (and would be even uglier to implement in Wrapper). He raises a `VetoAction` (so the widget does not process the event) and instead pop's up the suggestions list. (John Sarkela mentioned that in VSE, raising an exception in this way was the standard behaviour of code that was interested in #aboutTo... events.) Vassili's conclusion was that in Pollock when you try the reasonable way

to do something, it works.

Q. Pragmas in Namespaces? Could be in variables, use for loading.

Discussion: tool support interacts with having state machine accessible in global or not.

Never mind the quality - feel the width; 64-bit VM, Eliot Miranda, Cincom

Customers need to be able to exceed the 4Gb limit, which means addressing the full 64-bit system. It must be performant. For symbolic computations it would naturally run slower (twice as much data to walk) and this must be addressed. On the other hand, floating point should naturally be faster. Ideally, it should be just another VisualWorks platform.

The implementation runs but is incomplete and has no auto-conversion between images. Eliot loaded a patch for an issue he found last night and then converted an image from 32 to 64 bit (this is moving to a menu near you soon :-). The first cut runs quickly after which it does an exhaustive leak check. When this is released, this check will not be needed so it will be fast. He did maxVal on 32 and 64 - feel the width !!!

Eliot was using Martin Kobetic's presenter framework - lets you edit code in the middle of powerpoint. He converted his image and restarted - there's confidence for you. It just works. The infinite-precision arithmetic and VM primitives provided all the abstraction he needed to hide the difference. The primitive set is 97% the same (3% SmallDouble differences). If a method has 6 or few bytecodes, it is held in compressed form; this could have been upped this to 7 bytecodes in 64 bit but that would have broken many compatibilities.

There are two formats: immediate (2 bits of class-identifying tags and 30 of data) and everything else. (header of address 32, class oop 32, flags...). In 64 bits, there are immediates (3 bits of tags, 61 bits of data) and everything else (header with 64 bit address, flagword 64, ...).

The class tag in 64 bits is a hash of 20 bits and the class tag fits into the 32 bit hash (limits you to one million classes; not expected to be an immediate problem and will be revisited :-). The class tag fits into the 32-bit inline cache which is very useful (e.g. Sparc 64 needs 6 instructions to synthesis 64 bit lookup). A table (1024 WeakArray of 1024 elements) maps the class tag to the class. Size growth is ~52% (was 50% when 16 bit became 32 bit); much of the system (~1/3) is byte data which does not change.

Eliot ran some performance tests (always run block once, then N times profiled, to ensure all caches primed to avoid results being confounded with that cost).

Eliot looked at why he has restricted to only 3 immediates. Most problems are easily soluble but the advantage of compatibility between 32 and 64 bit engine source was just too great to make changing worth it. The space

saving is only 0.4%.

SmallDouble fits into a single 64 bits and reproduces 32-bit double. If the number grows larger, shifting it to full IEEE 64bit float is very easy and takes only 3 times as long as shifting a SmallInteger to an Integer - hey, aren't floats supposed to be much slower in Smalltalk. Eliot demoed multiplying (not inlined) and plus (inlined for SmallInteger); the latter showed an additional factor of two due to the inlining. (And if you check code like 1 + 1 then the translator can tell to push 2 and so that looks much faster; be careful to compare integer arithmetic to floats, not some special case optimisation to floats.)

The ImageWriter reads the image, constructs the object graph and processes it to write the new image.

Q (Roel) ... discussion ...? The WeakArrays used for class hashes are hidden from allInstances, as for Context and other key things used in execution.

The image startup does rehashing if hash maxima change, e.g. moving from 64 to 32 bit. Because this works on image startup, an exotic hash function could cause problems in early startup - very undesirable. No such problem has been seen as yet.

You cannot control where things are mapped in Old Shared perm space; you certainly cannot say 'store things low'. Mark van Gulik spotted that you can store two 8 bit words in a 16 bit length as 0\\16 or 8\\16; these two alignments give you an extra tag bit which can be used to handle this. Shared perm should be done by 7.3.1. Today it runs on Linux x86-64. A French company is keen to see it run on 64-bit HP-UX. They will also pursue 64-bit Solaris (especially if there is an x86).

The work caused 5,000 new lines of code in 200,000. The area most changed is the translators instantiation generator where 50% of the code lines have changed.

In the future, the tag idea could be used in the 32 bit system (16 bit headers in 32 bit system). Adding class tags to a 32-bit system would grow the id-hash from 14 to 18 bits. As we grow beyond a few Gb, the garbage collector may need review.

Q. Any users of the floating point 64-bit type? Quallaby have very large images (2Gb footprints). Quallaby have just done a 9-digit deal with BT to provide BT management software throughout Europe.

On 32 bit, float is faster than double. On 64-bit, double is faster than SmallDouble. OK, we could have done an immediate float but there was no value. You could optimise to use all doubles but at the moment we don't want to change the semantics on everyone. In a few releases, we may.

64-bit system benchmarks are 15% slower than 32-bit on identical

hardware, which is doing quite well.

Q. You can tell what you are in? Yes.

```
(SmallInteger maxVal + SmallInteger maxVal) class
```

will return different values in 32 and 64.

The Smalltalk RunTime Environment
Stripping is a nightmare; it happens in the dark and takes a lot of effort for real savings. Having a GUI is great for GUI applications but Smalltalk should also be a great scripting language. The aim is to build a small core image, let it communicate with its context, not just through the GUI but via web browsers, unix shell or whatever for i/o and debugging.

As of 7.3, many more of the tools are in parcels. The command line has evaluate, etc. The subsystem hierarchy lets you modularise startup issues. You can use stdio on all platforms (except MacOS9).

The final core image will be 2+ Mb with maths, collections, all the kernel services of the memory management, etc. With this you can script, just as if you had a workspace, run in e.g. a web plugin, and while the immediate debugging will be much more useful than in the stripper, the key value is when remote debugging tools let you use a full GUI to interact with the image. Preview remote debugger exists and works now. (I saw it at ESUG.)

Q. Use Traits for rebuilding system? We like it, have concerns on whether we can use it and hide it for backwards-compatibility. They cannot make a release dependent on a successful Traits implementation as it might not work. So they must prototype it.

The Value of Smalltalk: Insuring in Smalltalk, Niall Ross, eXtremeMetaProgrammers
This talk began with every presenter's nightmare. As first in the session, I had time to prepare, checking that my slides displayed OK, etc. Just as Helge was introducing me, telling the audience what a clever chap I was, I interrupted the 'knights of the square bracket' screen-saver to redisplay my slides - and there they weren't!!! OK, I thought, here's a chance to show I can deal calmly with a minor difficulty; I'll reboot the machine to clear the corrupt display state, giving the talk for my not-too-important introductory slides while it does so, and so get back on track. The laptop reboots - still no slides :-/. Sudhakar Krishnamachari kindly lent me his computer and, after 5 minutes 'talk amongst yourselves, please', we had slides once more and the talk resumed. My notes below blandly ignore this little hiccough (and read a little more smoothly than the talk did in actuality :-).

I find it hard to explain things I have learned by experience instead of by being taught them. Noone ever told me to use Smalltalk because ... . One day, in the early nineties, my quondam manager told me I'd be working on a project in some new language "... Smarttalk - er, no I mean Smalltalk, I think ..." and I said "O.K.; I don't mind learning another language." (This way of discovering Smalltalk is not uncommon. Avi went to OOPSLA in

2001 with a paper on Java, but rashly walked into the Camp Smalltalk room to see what was going on there; in a sense, he never left. :-)

A few years later the scene repeated itself, with my then manager telling me that now everything would be written in Java. The manager was more keen this time, but I was less so; less keen for myself, but also I didn't think it would do the company much good. But how to explain why?

Over the years since, I've had several attempts at explaining to various audiences why it is not in their interests (or mine :-) to ignore Smalltalk. Very often, in the years of the Java wave, I had but few and hard-won successes battling against fashion. My most recent attempt, at ESUG 2004 in Kothen this September, was more successful. (Of course, I was presenting to an audience of Smalltalk enthusiasts. :-)

Rather than just repeating that talk, which is written up on the web and which those of you in the audience who were there doubtless remember well from so recently, I would like today to complement that talk by presenting yet more reasons for, and examples of, Smalltalk giving value. So this talk could be subtitled:

(yet more illustrations, and explanations of) The Value of Smalltalk

This write-up should be read alongside that of the previous talk (notes at http://www.whysmalltalk.com/events/nfrESUG2004report.pdf page 33, slides at http://www.esug.org/data/ESUG2004/ValueOfSmalltalk.pdf).

The aim of these talks is two-fold. Firstly, I'd like to be convincing; talking to people helps me to build a clear body of material and to improve the clarity with which I handle it. And secondly, I want other Smalltalkers to be convincing. Some of the ideas and examples I present in this series of talks may help others defend Smalltalk to their managers and clients.

First, a little theory. Why does Smalltalk have value? If I tried to explain all the reasons, this talk could be a lot longer and still have no time for any examples. I'll describe one.

Why Smalltalk is better than its main commercial rivals, C#, Java and C++, is captured by the term Smalltalkers use to express the most obvious technical difference between them. Smalltalk is a dynamically-typed language. These rivals are commonly called 'statically-typed languages'. But a Smalltalker will often call them 'stiffly-typed languages'.

This isn't simply a piece of abuse (though it's certainly that :-)). It expresses succinctly the feeling you get if you experience building and maintaining real systems first in Smalltalk and then in C#, Java or C++.

- Firstly, the rival language seems 'stiff'. Like a bad employee, it resists your attempts to make it do something, and it resists still more your attempts to make it do the right thing.
- Secondly, much of this 'stiffness' seems to be related somehow to its static typing.

Many Smalltalkers will talk for hours to anyone who will listen about what the relations are between static typing and poor productivity. I will talk for a few minutes, not about *what* these relations are, but about *why* they are.

I must begin with a confession. When I was young and starting my career as a software engineer, I was a fool. You will all recognise instantly that this seemingly humble confession has the not-so-humble implication that as I am now old-*er*, I must be wise-*er*. However that may be, I know that when I was starting out, I accepted uncritically a lot of very silly ideas about software engineering:

- that errors cost more the later in development you find them, so you should spend a lot of time up-front finding them all in the design phase
- that, 'Why aren't you coding yet?', was a wicked question asked by bad managers, which should always be ignored;
- that 'Coding from a design is like walking on water; it's easier when it's frozen.' And since (if you believed this) you spent time up-front to get your design right, now that it's 'right', you should not change it.

These were truisms of software engineering gurus when I started out; by many, they're still believed today. I believed them because I was stupid enough to think that I was clever. That is, clever *enough*; clever enough to know early in a project what I was trying to do and how I could do it. The process of learning that I never did was long, embarrassing and painful. The process of learning that my team leaders and managers never did either was long, infuriating and painful.

One day, when I had learned this lesson but not yet thought to learn from it, I was taught the discipline of eXtreme Programming, the first agile method. It says the way to deliver good software is *first* make it run, *then* make it right, *last* make it fast. To the obvious objection, "Why not first make it right and then make it run?", XP replies, "That shot is not on the table." If you know any programmers who can get it right first time, hire them; pay them well; don't let them leave. Because you will not replace them in a hurry. Mostly, the only programmers you can hire are people like me; people who get it wrong the first time round.

In short, no one is clever enough to make it right without a running system to study. The only real option is to make it work and thence to make it right. Here, 'wrong' does not just mean explicit error. It also means building features the customer doesn't want that use up time and get in the way of features you then discover *are* wanted. 'You won't need it', is an XP motto.

The third part of the mantra, 'last make it fast', is saying the same thing. The optimisations you haven't already got by making your program 'right' will be ones that make your program more computer-friendly and less human-friendly. At best, they will trade flexibility for speed. At worst, they will trade it for nothing: 'first make it wrong' applies to optimisation code as well. In short, they will make your program 'stiff'. Since your program will not be right till well on in the project, this trade must be made as late as possible. And since you will not have a system whose performance problems can be meaningfully studied till then, this trade cannot be made

successfully till then at the earliest. All performance gurus agree that you cannot guess about performance; you see a system's true bottlenecks only in realistic use. XP guru Kent Beck expresses it thus, 'There are two rules about optimisation. Rule 1: do it later. Rule 2: see rule 1.'

Now we can get back to our subject. Statically-typed languages are 'stiff' because their most basic design violates these rules. The type system is an optimisation built into the basics of the language by designers who thought they could work out in advance what was needed. Smalltalk's designers told themselves, 'We won't need it.' (It's no accident that the theory of eXtreme Programming was invented by Smalltalkers.) Because Smalltalk is flexible, researchers have since written acceptably-performant utilities that add static typing to Smalltalk. They never caught on because, as everyone who writes Smalltalk finds, you *don't* need it. Smalltalk, more than any other language I know, consists of things its users found they *did* need, not of things its designers thought others *might* need.

Just as agile methods don't say what features of a particular stiffly-run project will inhibit its success, only that there are likely to be some, so I will not try to illustrate all the particular features of C#, of Java and of C++ that cause each to be less productive than Smalltalk. I've presented a theory of why there must be many such. In the time remaining I'll mention a few examples I've encountered of how this, and other aspects of Smalltalk, translate into business values.

Firstly, forget rapid delivery, cost-effective delivery and all those nice-to-have's and think simply about actually delivering anything at all. Some years ago, I worked on an insurance system written in Smalltalk. The domain was subject to fairly rapid change. The insurance product set changed. The business logic also changed: to enable new products or to give existing ones competitive advantage, to improve the business process, and last but not least, to comply with, and keep working in the face of, regulation.

The systems designers soon found they needed dynamic domain models: meta-data. They implemented and evolved a framework for personal insurance products. The result was a stable team of some 10 or more supporting, maintaining and developing a system that grew over several years to handle hundreds of insurance products, easily adding new ones.

At the time I worked with them, the firm had recently merged with another, giving us all a close-up view of some alternative approaches; for example, using a larger team to maintain 700 hand-coded forms that supported just 9 products. It was not that the rival teams *had* to do it that way, just because they used statically-typed languages. Coding a meta-data approach would indeed have been much harder for them. But what would have (and proved to be) been very much harder still for them was *thinking* of it. Over a two-year period following the merger, a colleague who worked with both the Smalltalk and the stiffly-typed systems gained personal experience of this.

Firstly, he experienced the technical issue. Ralph Johnson's meta-data rules

are not to build a framework until

- you need to
- you've done three examples (his 'rule of three')

Smalltalk suits this rule well. Everything is an Object, so nothing resists being treated as an object when you discover the framework requires it. More importantly, every type is object, so you can mix and match meta-data and ordinary data in your algorithms, can easily override your first-attempt at a generic algorithm in the special cases it cannot yet handle, etc.

Stiffly-typed languages don't suit these rules well. Being told that what you regarded as a class last week, or in another mode of a program's use, is here and now to be regarded as an object, or as a string of source code, is just the kind of thing its designers didn't expect you to need, either at all or in a range of cases where you find you do. Therefore it's just the kind of decision they force you to make in an un-agile way. Either you make it up front, building a framework before you know you need to and before you've learned from examples, or you find you need a framework after building a system that resists evolving to use one. He felt developing such a framework from scratch in such languages would have been all but impossible. Attempting to reuse their known design hit this problem quite often enough as it compelled frustratingly-slow redesign and rework.

Secondly, he experienced the psychological issue: this proved the greater problem. He explained it to me thus: "It is not that it's *impossible* to implement really accurate models in other languages. Rather, one must constantly overcome the inertia of an inelegant language implementation. If all developers think in the elegant patterns that Smalltalk encourages, the inertia can be overcome. However, most developers have not worked with a fundamentally elegant language to shape the concepts they project onto their daily work. When I'm denied the use of Smalltalk as an implementation tool, I still think with the benefit of the elegance of those concepts embedded in Smalltalk. You can take the developer out of Smalltalk but not the Smalltalk out of the developer." And conversely, if a developer has never had significant contact with Smalltalk in real large systems, it is very difficult to put the Smalltalk in.

In short, what he found was not that it was strictly impossible to implement an effective meta-data design that could have let the team of several tens shrink its numbers of staff and raise its numbers of products. Although it was much harder, he did enough work to show it could be done, especially when you had an existing proven design as a basis. What proved too hard was getting non-Smalltalkers to resist the pull of their stiffly-typed languages. Meta-modelling is an intellectual challenge anyway. Asking people who had no experience of anything else to meta-model in a language that was profoundly hostile to its key patterns was too much. The theoretical ability to deliver meta-programming in a stiffly-typed language proved practically unavailable even in these favourable circumstances.

(One may note by way of illustration that as a Smalltalk program ends up as bytecodes, it is always theoretically possible to write any Smalltalk

program directly in bytecodes. However it is psychologically impossible for all but the simplest programs. What this group found was that, when meta-data enters the picture, something similar begins to occur for the jump from Smalltalk to a statically-typed language.)

The group I mentioned above therefore continued to deliver a slow drip of hand-coded forms for a small product set (and similarly in the other non-Smalltalk groups). The half-an-order-of-magnitude smaller Smalltalk group continued to manage a two-orders-of-magnitude more powerful system. It would be unfair to offer all this difference as an example of a second value of Smalltalk, more rapid delivery of what *can* practically be delivered in any language, since if they had managed to get a meta-data framework working, they would have caught up part-way and if the Smalltalk group had chosen not to use meta-data, they would have achieved less. However my colleague found that even when coding ordinary deliverables in ordinary ways, there was a noticeable difference.

Both the rapid evolution of insurance products and the (even more :-) rapid evolution of regulations for them meant that all the firm's systems were under continuous pressure to adapt to new requirements. You can imagine how frustrating it could be that the Smalltalk could be adapted quickly, often via mere data-fill, leaving plenty of time for work on C#, VB, etc.

A third key Smalltalk value is scalability. I think the scalability of this system, is already illustrated by what I've said above so I'll move on to my fourth Smalltalk value, reengineering. I saw lots of examples during my work with the project. I'll describe the one that impressed me most.

A key function of the application was to enforce business logic. A key part of its usability was helping users understand whenever a function was blocked, or supplied data not accepted, because of business rules. Enforcing and reporting business rules had been designed in from the beginning, but a moment came when the chief architect realised that it was not being done as well as it could be. The meta-data framework had evolved to the point where you could deduce the widgets in which the user had supplied data that was causing the business rule to fail. Hence, the chief architect could write a generic function that, for any current or future rule and any current or future UI, would scroll to the relevant UI page(s) and flash the relevant widget(s) whenever the user selected the corresponding item in a business logic analysis?

Unfortunately, noone had foreseen this. Hence the relevant data and method calls were scattered in random order through the code, united by the flow of control and by nothing else. Enter the rewrite framework. The custom refactoring that the chief architect wrote to tie together all the data he needed was the most complex I have yet seen used for a real business purpose. Its invocation logic needed some knowledge of the framework and it had seven match-and-replace pattern-pairs, some of which were themselves complex. Nevertheless, it was vastly easier than recoding both the business logic in the model layer and its handling in the UI layer by hand in a system of some 5000 classes and 100,000 methods; it took one

week to write and test this refactoring. Much faster than they could have expected, users received the new, highly visual and interactive business logic explanation feature, and were very pleased with it.

(This refactoring's code is not in the public domain but a simpler follow-up refactoring that I wrote with a colleague is in the Custom Refactoring project's code examples, trivially rewritten for generality and anonymity; see `ExSearchSymbolsTestNfr`. Since we wrote it test-driven while remotely-paired over NetMeeting, the paper I submitted to Dan Antion's Agile Methods panel at Smalltalk Solutions 2003 was a detailed write-up of how it was written and what it does. The paper is on Dan's website.)

That Smalltalk has a powerful, customisable refactoring framework whose features can easily be mixed and matched with ordinary Smalltalk code is an example of the point I made at the beginning of the talk; there is no type system to get in the way (some of the issues are hinted at on page http://portal.acm.org/citation.cfm?id=367992&jmp=cit&dl=portal&dl=ACM). Adapting this framework to project-specific needs is just one of the more powerful and unusual of the many reengineering techniques that are easier in Smalltalk.

From Fat Client to Service-Oriented Architecture, Helge Novak, Michel Bany, Cincom

Often today, people are maintaining technology and then a manager says, 'Let's go to the web'. Or, 'We must have a service architecture.' CEOs want to leverage existing systems, not do costly and risky investments. Re-writing a system costs a new development project, *plus* securing old data, transferring it, etc.

Making a system server-ready needs several things. A fat client is single-user whereas a web-based system will be multi-user from higher up. Performance needs will change. You need to build the server and add administration and monitoring tools.

Web Applications are a sum of all evils: page-oriented computation with no state, i.e. no knowledge about what happened between pages. The protocol (http) between client and server is very poor. You have little control over the GUI as regards the flow of the user interaction, and the GUI is very dumb. The connection is largely out of your control and can be very poor. Your client is embedded in an external application (web browser) and you have virtually no control over the user's use of its features (back button, bookmarks, ...).

You must split the user interaction into chunks that fit this situation. You need sessions for concurrency. Sessions need state to handle control flow, and you still have to worry about transactions and loss of connections. You have to build a good UI, not just layout but also Javascript or whatever.

Smalltalk offers server pages, servlets, custom tags. It also offers frameworks: Seaside, Tsunami, generation from Smalltalk UI.

Michel then demoed an example application to count the registrations for a conference. The app was built in ObjectStudio. The demo was to expose it as a service, (OpenTalk), build web user interface (VW + WebToolkit + Seaside), then build a SOAP gateway to expose it as a service (VW+WebServices) and finally he built a simple SOAP client (VW + WebServices).

Helge and Michael had generously sacrificed ten minutes from their talk so that my preceding talk could finish despite my machine problems. As they tried to start their demo, they found that they too had to reboot (it seemed to be a feature of that session :-/). Hence they finished the demo later in a parallel session.

How Smalltalk saved us from an error: Koramis SpecSheet: an intuitive web application, Hans-Peter Fichtner Koramis GmbH and Co KG

Peter is an electrical engineer who since the 1980's has built up several companies.

Suppose you have 12,000 parts with 6,000 characteristics. What do a cup, a toilet tank and a canal lock have in common? They all care about the level of a liquid. If you want a device to manage them, it should know about that.

Peter's system uses SpecSheets to capture these requirements. The user logs in (it is a webapp) and specifies their most important characteristics. Peter demoed by specifying an industrial-scale thermometer. The UI captured the ranges and distributions expected, the input mode (analogue) and style, the mounting, etc.

A customer specifies the general characteristics of what device they want, then they must say how the device must operate, and so on, thence in the end they construct a meaningful request to the supplier: "Can you supply this device?" This meaningful request is the value proposition; the customer usually does not know what questions the supplier needs answered before they can answer the customer; the system maps the customer's knowledge to this. The system can produce a simple listing of these, a formatted email or a standard engineering specification sheet. Peter then presented a hypothetical spec sheet for the system.

In 2000, Peter's vision was this system that could understand a device' requirements. He had to transform this into a system. He started looking for a web system in 2001; ABB recommended VisualWorks to him in late 2002. In March 2003, he contacted Heeg and things started to move.

The market is huge: in Germany 600,000 people in 75,000 engineering companies and 5000 affiliated groups write such specs as part of their daily work. (SpecSheet is multi-lingual, so perhaps will be used more widely even than this.) So far, feedback on the system has been very positive.

Q. Very large market? Indeed (statistics above from German government).

Q. Tool is domain-neutral; could be power plant or chemical plant, not electrical engineering? Yes; it could be PC specification if wanted. (And use in Cincom's support web? No that is a different domain.)

Web Application development in the MVC Style, Andreas Tonne, Georg Heeg

Andreas talked about how the above system was built. Peter started with vast amounts of data in Excel spreadsheets containing lots of specific specification examples. When he was told to look at VW, Andreas had to satisfy him that solution was feasible. They used VisualWaf (Georg Heeg product) to do this.

As Helge mentioned earlier, web applications are a bit alien: transaction-oriented stateless text-only system, no interaction between transactions and very primitive presentation options. Smalltalkers are used to MVC, value models and aspect channels, and events. Heeg wanted to make web-app development as easy as Smalltalk GUI development. They also wanted to preserve the application concepts of the web because that is how web designers think and you must work with them, not become 'the aliens' to them. Web browsers emit requests into the Smalltalk domain and get responses back from it. VisualWaf has a PageView to provide responses and a PageController to receive requests: MVC for the web.

Andreas demoed. He initialised the server and showed it in the WebToolkit admin page. He defined a demo task via a <task: DemoTask> pragma. He specified the task in a new code tool in the RB. The DemoTask was just a class with a defined interface. `entryFromPage` returns the page object with suitable values

```
entryFromPage
page := Heeg.Page new.
page value: ..type:... mapTo: self @ Customer.
...
```

This is the last point at which the programmer needs to be aware of the web. He brought up the page and showed that the debugger would show you what was not yet connected. He needed to connect up the 'Accept' button, which he did.

```
accept
  ...
  self hasErrors ifFalse:
    ["do the accept thing"
    ...
    "and you add a page"
    page value: #Accept type:... mapTo: self @ Accept.
```

At this point, Andreas noted that the raw generated form would need work by a web designer. He showed what the pages looked like after they were done, attractive in the web browser, ugly HTML text in a text browser but that's OK. The designed page has brief scripts that connect back to the Smalltalk. Moving them from the initial generated page to the stylish designed page is easy.

The Koramis model-layer application is more complex than the demo of

course; it has eight tasks running on its page. It has 12 packages, 111 classes, 612 methods. Its web application is 10 packages, 53 classes, 403 methods, 9 tasks, 6 views, 1 controller and 57 ssp pages.

Web apps can be done quickly: 15 minutes work shows progress. It is as cheap as GUI applications (unless you hire costly web designers to have superbly styled pages of course). Heeg offer a CPU-based licence (there is an evaluation version).

Q. Like Java Struts? Yes, they copied from us. :-) We had this working first (it grew out of work they did in VA years ago). Behind the scenes, Struts is very different from VisualWaf.

Q. Could you manage the html in Smalltalk (and in CM, e.g. Store)? We have not yet done this.

Reusable Web Development with Seaside, Avi Bryant
(See also Avi's talks at Smalltalk Solutions and ESUG, written up in their conference reports on the whysmalltalk Events and Trip Reports page.)

How do you build web frameworks to let reuse happen?

Typical UI frameworks have widget-level granularity; build a nice widget and you can reuse it. On the web, it is page-level granularity. You need to break out the parts you will reuse. More challenging still is the coupling in Web apps. Web pages use go-to's to jump between each other. So you cannot reuse because you must always jump back to the same page.

Paul Graham had a colour-swatch in his eCommerce store builder. It let you choose colour for the background, the foreground and much else. His colour picker was a page to itself, called from everywhere and wanting to go back where it was called from (i.e to go on from there) once the colour was set. Doing it with go-to's would have been a problem (as rivals found).

To break out widgets you must decompose your pages into a tree of objects, each contributing a snippet of html. In Seaside, these are objects and `renderContentOn:` is in a visitor-like pattern to walk the graph and add html to the tree until the whole is done. However that is only the response side. What if one of these widgets has a name for a form input and another widget uses the same name. If you use a widget (e.g. an address form) twice on a page, you will definitely have this happening.

Seaside solves this by having block-based callbacks. Don't give things a name, give them a block.

```
renderContentOn: html
  self batch do:
    [:each | html anchorWithAction:
      [self chooseClass: each text: each name]]
```

Each Seaside widget subclasses `Component`. Avi showed the `BatchList` component that simply encapsulates a collection and renders it. He used it to show the classes in his demo. Selecting an item opens a system browser

on page (a web component showing refactoring browser ships with Seaside, giving another meaning to the phrase 'web development'). Avi wrote some more code and got a web page showing a DNU. He fixed it and then he could select a class and see a sublist of its subclasses.

That was reuse of a piece of a page. He then talked about reusing pages. The Cincom non-commercial download web app makes you visit the page 'What do I want to download: VW or OS or ...?' twice. Avi knows why they did that. If they had a specific second page for each product, the prior page could not hardlink to it. If we could handle pages as subroutines, not go-to's, we could reuse them more easily. He showed the pseudo-code (which Avi misspelt - after all, when you have Smalltalk, who needs pseudo-code)

```
registeredDownload
  | product user|
  product := self chooseProduct.
  self showProductInfo: product.
  user := self getUserInfo.
  self offerDownloadOf: product to: user.
```

In Seaside, this is exactly what the code looks like. And because this control flow is now all in one method, we can change it very easily. Do we want the user to register before listing our products instead of afterwards; Avi showed changing code and rerunning the pages easily; now the user can go back and change their mind.

Q. This is unique to Smalltalk? Seaside uses two basic things: web blocks (Java anonymous classes for the callbacks would be atrocious) and continuations. Most languages do not give you access to the context. Some languages (Scheme and Ruby) give you continuations as first class objects so you could do that. Continuations are not possible in C#, in Java, etc. because they do not allow access or provide continuations as a supported context. You would have to hack bytecode !!!

Michael Bany now presented. He had to provide a solution for a customer so looked at Seaside. Michel has ported the latest Squeak build to VW and will be keeping the two in sync; just load the Seaside parcels. You can choose two flavours, one using the WebToolkit and one using Swazoo; it is not strongly coupled to other frameworks.

Michel loaded into a clean image, showing how the load prompted you to create sample sites (gets you started quicker if you say yes), sets the timezone (US or Europe - I must add a UK timezone setter). He showed the shopping demo, selecting some items and providing his billing and shipping addresses. The application assured him his raw fish was on its way to him. :-) He then showed the page in development mode with icons ('halos') letting you see an item's html, open the class browser on its Smalltalk (in VW, it is a reimplementation of the Squeak class browser; I will look at getting the RB working). (Usual demo hiccough - needed to find CSS file for demo; he just needed to clear the browser's cache.). He typed a bug into the code, saved and elicited an error (shopper DNUed deleting items). He saw the debugger (again on web) and would have

corrected his code in the debugger but this block's home content was not on the stack so he corrected it in the class browser.

He showed inspecting objects over the web, browsing the current state of his shopping cart. Another halo lets you see the CSS style of an object; he changed a widget's colour from grey to blue. He then changed the deletion to let the user delete all items of a given kind, not one by one. He created WAOneOrAllDialog (like WAYesNoDialog in the distribution) and changed the callback code for deletion to use it. He clicked to remove: the list was replaced by the dialog. He chose and the list repeated, updated. He showed walking to the showing-dialog state and then changing its appearance in development mode, spacing out the buttons more.

Q. Halos: what more is needed? The VW ones are inspired by Seaside (not the same code). A useful addition would be to popup an inspector or class browser in the image, not on the web, when that was the best way to progress coding.

Impromptu Demos

MetaEdit demo, Steve Kelly, MetaCase

Domain-specific modelling languages:

- most people are well below doing this approach
- those who use it find a 5 - 10 times speed up compared with how they did it before

This is a general result, not specific to a domain. (Or to their tool. Dome was like their tool, went very quiet, is little used outside Honeywell, who now say they'll do it in Java!!! GME is free but not open-source, and Eclipse is aiming to be a little like this, but those are frameworks, not tools and Steve, having done the equivalent in VW, knows how hard it is.) Steve mentioned two examples:

- German insurance company: 00's of products for insuring against fire, theft, etc.
- Telecoms company: many products for configuring telecoms services.

He showed their standard training stop-watch example: the key issue is abstracting out threads and synchronisation. This example generates Java (as everyone has that VM on their desktop, so demo output can be given to anyone).

Nokia has a more complicated domain-specific modelling language but builds things essentially this way.

Smalltalk has less need for this kind of tool as it is already three times as productive or more. Thus the tool is in Smalltalk but in a typical use scenario it may generate other languages.

It takes 2 weeks to 2 months effort to build the model and the generators. It takes 50-60% on thinking, 10% on implementing the modelling language and 30% drawing the symbols (the drawing is easy but agreeing what they should look like is hard).

Q. Automotive companies want more assembly line automation; could use

Steve knows (just count the lines to verify it) that it is a 13 month project to write the Java in Eclipse that is the equivalent of a one-hour-to-write model. (And an IBM consultant at a conference claimed that this 2000 hours estimate was excessive and it just took them 600 hours. :-)

Microsoft are trying to build such a tool; downloading it requires two spare server machines and a client.

Steve then built Vassili's tooltip finite state machine in MetaEdit. He created a graph model, added a State concept to it and put down Vassili's four states. In MetaEdit, a relationship has three roles connecting it to its two end-points and to any object that provides data for it. Steve drew a directed relationship shape. He then assigned it to abstract relationship class, specialising to types Timeout, that has a duration (drew shape to show it), and Event, that has a string whose values must be in the list of known event names (he typed a few into the property definition to get things started). He then created bindings saying what the relationship's from, to and data roles were. He then had to reopen the specific-modelling window to make the new tools appear on the toolbar (MetaEdit was written before there were pragmas). With the tools available, he drew Vassili's model.

Vassili generated temporary variables for each state. Steve created a new report. (With his code speed-up templates) he quickly produced simple generation of the states iteratively, tweaking the generator and checking what the output looked like.

The whole took one hour including 30 minutes of slide presentations. OK the code generator was very rough but you could see clearly that it was going to be well done in a few hours work.

WithStyle (demoed by Andrew McNeill, Cincom Australia)
WithStyle attempts to blend the advantages of thick clients with the web. It works by serving XML styled with CSS (2 at present, 3 with Pollock next year, when they will also be able to style Pollock widgets). They expect to be used as classic thick clients, as smart thin-clients on the web, as an embedded widget and as working with the VW plugin. Andrew showed a range of ZenGarden pages that WithStyle can handle. The Web Browser is just an application written in WithStyle; it is not just a web browser. Another application provides content for end-users of the road transport authority; it gives them a wysiwyg editing environment for creating pages for their site.

Andrew demoed a language-swapping application, showing the XHTML code. He has just discovered that you can show the output as a coding tool within withStyle for faster development. He showed popping up a dialog, adding text to a page, executing Smalltalk code (he changed it and reran). He browsed the methods showing the raw XML being served.

He then showed a page controlled by XSL so that in each context he could select a hierarchic-generalising list showing what he could change at each point in the screen (e.g. local text, paragraph, page).

For more on withStyle, see Rowan and Michael Lucas-Smith's talks at Smalltalk Solutions 2004 (my report is on the whysmalltalk Events page).

Customer Advisory Board Discussions

There was a general discussion group and specific ones for VW and for OS (which I missed).

General: Helge Novak, Claude Poole, Jim Robertson, Kim Thomas, Ron Weeks

Q. Unify VW and OS? Customer wants OS to become 100 times faster. Tomorrow there is a talk on what is being done here.

Q. Generally, Smalltalk is the second largest group in Cincom and 10% of company revenue. Smalltalk is 20% of Ron Weeks organisation and maybe 15% overall (30 to 40 people). 80% of support is answering usage-based Qs, only 20% is actually solving issues. Issue is how to get the message out.

Q. Web apps, e.g. Seaside the way to get the message out? Yes, they see areas.

Q. 5 years out? Gurus say IT shops get out of the way and end-users will specify applications. As UML is hopeless for this, it is an application for Smalltalk. There are also customer verticals. We are not looking to make Smalltalk the next big answer because we don't have a $300 million budget. Instead we grow with customers.

Q. Having Pollock would be really valuable. OO databases are becoming more and more valuable; improving integration with GemStone would be very valuable. Seaside; we have the latest stuff ported. Ask Steve Kelly about what meta-case is doing these days.

Q. (Steve Kelly) There is no huge marketing budget for Smalltalk or indeed for meta-case. One way to get the message out is to use the computer press by getting articles published; I've seen Alan and Eliot publish articles. Jim mentioned that a paper just on Smalltalk would not be accepted, nor would one just on Java, but you can get talks on XML, RSS or aggregators accepted (a book on Apache mentions BottomFeeder).

Q(Georg and Walt) I feel we are in a similar situation as in 1989. Walt was marketing manager in ParcPlace then; what could we do now to grow as we did then. Walt now works for Objectivity. There were RDB companies in 1990 that thought they would be the next revolution but the technology was too far ahead of its time and the applications had requirements that were never met as people dummied down. Now, this a time for challenging applications. Today (especially since 9/11), Objectivity finds people willing to invest in apps not with vastly complex front-ends but with vastly complex data processing. Also be aware that researchers and editors are

significantly influenced by who buys their research and their papers, so it is an uphill battle.

Q(Andreas Tonne) we had a very bad time in late 90's and just after the .com bubble burst, only seeing people they new. Then we started finding young people were interested in the technology and were interested. We need to educate them. (Jim) dynamic languages are on an upswing with both the young and the existing programmers. What we need to do is make people aware that Python and Ruby are not the only dynamic languages. For example, the Sydney Smalltalk User Group got involved in Python and Ruby groups; this is easy as they are usually very open.

Q. Also project Linux and apps, BottomFeeder, books, capture VA? We are looking at working with people who want to move from VA to VW; strategy is still being worked out.

BottomFeeder is a good example of where Smalltalk scores; it can function in a world of shifting standards, inconsistency and new needs surfacing randomly, and it is usable by a wide general audience. Publicise similar cases. Someone suggested games; provide templates. The existing toys often do not work (usually just dialect version issues); needs to be sorted. He has the corrections and can supply them (Jim: to me and/or work with the author). The toys could be used in BottomFeeder and they work fine

Roel is pleasantly surprised that there are lots of people here he does not know despite being in ESUG for years. Also he is the only one in academia; he teaches Smalltalk to his students. ESUG is ready to help people publish papers. (Jim is also ready to blog stuff, and to provide blogs.)

Local user groups are important and easy to organise. Cincom works closely with ESUG, with Squeak community. We need more internships; Stephane, Serge and Noury are professors who teach Smalltalk. They need places to place their students. Tell Cincom if there are opportunities for students to work with you.

A popular magazine in Germany asked about Smalltalk; some people replied they loved ST years ago, didn't know it was still alive, were glad it was. Some people got into it and liked it but met hurdles: pricing, availability of German material, how do I build GUI, package, etc. There is positive feedback on the language but these hurdles. Publishers are not negative about publishing in Smalltalk. (Alan) Editors want content all the time.

There is a new book, 'Introduction to Smalltalk', in German. Helge Novak will help people who want to write books in touch with a publisher.

The buzz site wants to publish content on-line. They have asked Jim (N.B. not other way around). Email Jim and he will guide you to offering content.

Can downloads from the Open Repository be counted, to motivate goodie providers?

Pollock sooner? MS has the same issue with windows forms versus Avalon. Its product manager has published a timeline and a 'why you should still work with windows forms' that is almost the same, even as to timeline, as Pollock.

Jim held a vote on which of various things were preferred. People want GLORP finished rather than the Lens improved. They want Pollock finished rather than Wrapper worked on. Other votes were less clear cut.

Q. Pricing model? Jim had a customer who thought they would move to Java and use open source tools but when they did the numbers they found their ROI was 10 - 15 years away, even in their own analysis. The only problem is that sometimes no-one is prepared to sit down and do that analysis. The open-source for Java stuff mostly comes from IBM to sell WebSphere which is not cheap and from Sun (whose business model makes little sense to Jim). ObjectShare went bust selling licences; VAR is essential for VW's continued existence.

Q. Who is your greatest competitor? New Smalltalk developments usually cite Java or C++ or C (sometimes Ruby or Python) as rivals from which they moved for various reasons.

VisualWorks
What is wanted?

Native widgets: the Cincom smalltalk Wiki has a roadmap for what is wanted and when it is expected.

Threads on the processor: map Smalltalk threads to native platform threads? When you run multiple platforms, native threads are an issue (c.f. the OpenTalk port to ObjectStudio). To do this right, which Java massively does not, you need a two-layer approach in which native threads cooperate to run a much larger number of high-level Smalltalk threads. Java only has one layer. But suppose you could do it. Then VM bugs become unrepeatable and failure modes explode. Eliot rather prefers clustering, with multiple VMs sharing memory (permSpace) since in this case the failure modes do not explode. Native threads would scale better on multi-processor machines *unless* you have shared perm between images, in which case this difference largely disappears. In a multi-thread bug, a serious bug in one thread brings the whole VM down and you cannot reproduce. Today, it is very easy to reproduce bugs and therefore fix them; reliability comes from this. (Solaris has kernel threads (not exposed) and Posix threads are a layer above these.)

XML Schema: the people in charge of this have been working on internationalisation issues, (which are important and took longer than expected). It is expected to move from preview to supported in the next major release. It may well be release quality now (the internationalisation work meant the team could not take time to bless it appropriately).

Scalable vector graphics: not planned, unsure what timeline would be.

GF/ST is really important to e.g. MetaCase. They are seeing more and more use of graphical UIs. Should we coalesce GF/ST and HotDraw. Kent Beck's slide framework in HotDraw is good. Steve suggests that GF/ST was a commercial product and might be the best place to start, but he notes that HotDraw is extremely interesting in terms of what is in there and how it is done. Sames will be looking at business graphics reimplementation and we should involve him. GF/ST is missing constraint stuff. Neither can fill all of an ellipse (due to a VM issue in windows; John Sarkela is aware of this and I *think* I heard he now has a fix).

Q.(Roel) Code configuration? Parcels will disappear except as a file format. 7.3 has packages in the base and drops categories as a primary view in the browser. Conditional dependencies are not being worked on at the moment. Jim sees cleaning up the requirements story as a key aim over the next one or two release cycles. Runtime packager will not be the way people deploy. Instead they will have runtime and development images giving more predictable deployment. At the moment, Runtime packager needs you to know what you are doing.

Q.(Niall) More support for porting from Envy and browsing old Envy? Alan's work on Store for GLORP aims to let you run Store in an old Envy VW image, thence remotely browse code from a connected image (i.e. running the new ported system) to assist diagnosing problems and understanding the old code.

Q. Automatic building and updating from Store. Doable (Steve Kelly does initial builds that way). It could be done (and with digital certificates so organisations with strict control of such process could use it). However a Store-connected image serving parcels over http, ftp or whatever is trivial and seems to meet current needs.

Q. Report framework? WithStyle could be the way to write a reporting tool that would produce stuff direct from VW. Jim has found WithStyle very easy to work with, very cleanly coded, when using it for presentation.

Helge mentioned that a German company <something>ReportStudio have a report framework and would be willing to provide it if they were paid for the work of extracting it from their system.

Q.(Georg) what needs are there for mobile applications?

Q.(Marten) Simple form generation? Georgio Ferraris has just built this.

Jim asked who used what. More people used web services than he expected. COMconnect: Joops sells a component (Cincom can resell it to you at a discount) called Smallcom/X that lets you embed an ActiveX component in a Smalltalk window and generally use COM more elegantly.

Q. Documentation? SmalltalkDoc is a project to make the system more accessible.

Q. Security? SSL is in 7.3. You can use SOAP over an https connection.

Q. Graphics e.g. transparency? Warpblt is in there but is not yet hooked up to the VW graphics. Squeak handles less than VW which must handle whatever the X server throws at it. Eliot hopes to get transparency released by 7.4; for now Warpblt is a goodie.

Eliot asked what Smalltalk needs. I suggested good meta-patterns (deepCopy, deepCompare and all the rest; I have some and will make them available). Georg thinks we should move away from methods as strings to reify them more. You could run Ruby or Python in VW and/or on the VW VM, and Eliot would like Squeak to target the VW VM. Then any Squeak program that needed high performance could get a cheap licence for just the VM. Roel would like continuations in the base, not as a goodie. Avi feels that a few VM tweaks could make continuations significantly faster; Eliot thinks the VM is already tweakable in that way and would enjoy enabling it.

Q. Could continuations enable Prolog in Smalltalk? Mark van Gulik wrote a paper in 88, 'Backtracking in Smalltalk without Kernel Support'. The problem is that the collection class hierarchy does not maintain its state.

Q. Working with GemStone? They are interested. When Cincom 64-bit VM is visible to them there will be possibilities; it needs agreement but they will be trying to make it happen.

VW7.0 has polling (deprecated). Polling cannot work in VW7.1 and after due to multi-processor scheduling.

General Discussions

Avi told me how the University of British Columbia did a Seaside project because they briefly hired an 'arts' professor who was very good at causing software to get written. He simply ignored their (non-Smalltalk) process.

Gemi<something> (GemiNet was what I remembered but that may not be quite right) is a German company (linked to Danish Navision?) doing EAI and ERP products in Smalltalk. They recently became an MS partner so Smalltalk is now a secondary part of the company but still important. New work is being done in VW and they also using Squeak on mobile devices (in warehouses: what items should packer collect). When they started this work, the VW VM did not run on such small devices, which is why they use Squeak.

Cornelius Ventris (IIRC) told me about the Calypso system of the Rand Merchant Bank (N.B. not the Royal Merchant Bank in London, who are also VW users). There are 900 - 1000 users of the system.

Database-DB do work for DeutcheBahn in Smalltalk. They have encountered some "Can you rewrite it in Java ..." issues.

Raat is a spin off from Roots to handle products, e.g. a contact management

product.

Alfred Wullschleger's talks usually describe his solutions / patterns for specific issues. We talked about one he has not yet presented which I encouraged him to write up (e.g. for this year's ESUG). (He mentioned that the Swiss National Bank no longer requires Swiss nationality for all employees; good news for non-Swiss Smalltalkers if they're ever hiring.)

Troudo Manz works with Daimler-Chrysler in Stuttgart on Smalltalk applications. She expects these applications to continue in use for another year to two years - the usual "We'll rewrite that in Java/C#/... in two years"; as the applications are not so complex, it could actually happen in this case.

Conclusions

Roel remarked that it was good to see so many people one did not know and so few from academia; this was very much a *commercial* conference. As a regular ESUG attender, I had also been expecting to see mostly familiar faces. Instead, the conference seemed to attract a whole other stratum of Smalltalk users. This is good, and good to know.

I look forward to working with VW7.3.

---

* End of Document *
