Smalltalk Conferences from December 2006 to May 2007

This document contains my reports of

- the Cincom User Group conference in Frankfurt, December 5 - 7, 2006
- the Smalltalk Solutions conference in Toronto April 30 - May 2, 2007
- the VASmalltalk User Group conference in Frankfurt, May 24, 2007

As a number of talks (the vendor roadmap talks of course, but also others) had shared material across these conferences, I attended such talks in the user conferences and the talks in rival threads in the Smalltalk Solutions conference. I have therefore combined these three conference reports into a single document. They follow in the order in which conferences occurred.

Style

'I' or 'my' refers to Niall Ross; speakers (other than myself) are referred to by name or in the third person. A question asked in or after a talk is prefixed by 'Q.' (sometimes I name the questioner; often I was too busy noting their question). A question not beginning with 'Q.' is a rhetorical question asked by the speaker (or is just my way of summarising their meaning).

Author's Disclaimer and Acknowledgements

These reports were written by Niall Ross of eXtremeMetaProgrammers Ltd. No view of any other organisation with which I am connected is expressed or implied. They are as accurate as my speed of typing in talks and my memory of them afterwards can make it; please send comments and corrections to nfr at bigwig dot net. I thank all the organisers, and the speakers and participants whose work gave me something to report.

Cincom Smalltalk User Group conference, Frankfurt, 5-7 December 2007

I came back from this conference with an umbrella, an excellent choice of freebie for a visitor from Britain and particularly so for one who had left his own umbrella on a tram during his last visit to the continent exactly one month before. :-)

See http://www.cincomsmalltalk.com/scripts/UserConference2006 for slides and other conference information.

Cincom Smalltalk Update

Cincom Smalltalk Roadmap, James Robertson, Cincom

Cincom Smalltalk started with regular releases to correct the ObjectShare-inspired impression that the product was dying. The product was going toward known goals but a particular release might or might not contain the completion of any particular goal. The new focus will be on specific feature releases pre-announced in the 13-16 month range, plus one annual bugfix release. Jim then discussed some candidates for these specific features.

Which goodies are supported, which are contributed? Icons indicate but more clarity is needed.

They aim to support ‘projects’, a higher level of control than bundles and packages. Users will be able to define kinds of projects, load all project prerequisites as a single operation, etc.

Who has used DLLCC? (many hands) Who has enjoyed using it? (two hands were raised; “You people are weird.”) DLLCC tries to do too much: it offers to parse the header file and do everything but it cannot actually deliver when the header file is non-trivial. Many people have told James that integrating with C libraries is harder than it needs to be. Integration with C can save you reinventing a wheel. For example, BottomFeeder needs to display data in a WithStyle widget from XML but the XML from the feeds is often ill-formed and LibTidy is a C program that cleans it up. Using this was a lot quicker than writing a tidier and let James concentrate on other things.

Ruby, Python, etc., people are used to messages, objects, etc., but they are used to writing code scripts and running them. Jim wants to let such people try that and so get them in easier than ‘dive into the image, you’ll be fine’. Letting people carry on with file-based support while they are adjusting to Smalltalk would also be helpful - never the recommended route of course but possible.

Creating an executable is also harder than it should be. RuntimePackager by default quits when it sees a DNU (Jim overrode this to carry on in BottomFeeder). More generally, building up an application would be better than stripping down.

Q. Pollock? First release will be winter 2007 for Pollock and Splash. Not all tools will be redone as Pollock tools then. (The transfer to the current GUI framework took several iterations up to VW 2.5 so they expect the same this time.)

Q. Threads? There are many reasons why VM will remain single threaded.

Q. Full unicode support? The goodie parcel does it now. Jim will provide a fuller answer on blog after he has talked to the people doing it.

Native widgets in Windows will be done via OS and so not soon.

Moving to ObjectStudio 8, Mark Grinnell, Andreas Hiltner, Cincom
There are various syntax differences (and one difference of opinion between Georg and Mark over whether unary minus is one of them :-). VW does not allow assigning to method parameters (but VW can if a VM setting is changed, Vassili tells me). OS is stricter in preventing duplicate class or instance variables, references to undefined Globals and changes to immutables. Mark will make the OS Smalllint tool available again (was on wiki but has disappeared).

OS 8 uses the VW Magnitude and Collection hierarchy. OS7 had conversion behaviour in collections so that

```
(Set with: ‘2006 12 3’ asDate) includes: ‘2006 12 3’
```

returned true in OS7; it does not in OS8. Collection classes use methods (e.g. =) from VW so do not use any OS-specific equivalent methods. The VW collection requirements (e.g. that subclasses of Collection which add instVars must reimplement `copyEmpty`) must now be met in OS.

OS single-parameter collection methods return `self`; VW's return the parameter. You may need to add `yourself` to various calls.

Attempts to create invalid dates no longer succeed. Months are case sensitive.

OS 7 used native threads. The VW VM does not and neither does OS 8; they find this is an improvement. Developers have spent ages debugging native thread issues; Smalltalk runs far better than native threads and they are far less expensive. You cannot specifically exploit multi-processor CPUs but otherwise it is all gain. (Jim mentioned that spawning multiple sockets in Smalltalk threads is far safer than spawning via native threads; BottomFeeder would bring any Windows machine swiftly to its knees if it spawned native threads.)

OS SendHooks were evaluated for every method and slowed any system 200%; they are now replaced by MethodWrappers.

DLLs will need recompiling with the same flags as OS8; an example makefile is provided. NULLs and 0 no longer work as nil equivalents.

Q. VW has limit on symbols per method? No, not any more.

Q. Dictionaries could not have nil values in OS? This non-adding of nil is still true if you do `osAt: aKey put: nil` but not in Dictionaries as such. (Questioner felt it was an improvement that nil values are now possible.)

Q Native access to SQL server? No we use ODBC which is the 'native' access now; the old DLL is not going forward. Generally, they aim to call APIs directly, not via DLLCC.

Q. Oracle connection has two flavours: proC and OCI (this uses generic DLLCC). In old OS, OCI was very slow due to slow garbage collection; in VW, GC is better and it should be faster. (Some discussion of dynamic SQL which Andreas said was used by both proC and OCI so was not a factor.) Generally, precompilers generated C code, not C++ code so were a problem for the port.

Current and Upcoming Store Developments, Alan Knight
In 7.5, Store will be internationalised.

Shadow compiling: Store may partially load code because it detects unloadable code (superclass missing), because loading causes a walkback, or because both old and new versions are internally consistent but a half-loaded state is not, e.g. loading a new Oracle Connect version from an Oracle Store database may crash half-way through.

If a file-in errors, it DNUs: the prior code is in the image and you must usually quit. Parcel loading reads all the code into internal data structures and applies it all once finished; if there is an error, no code is installed. Parcels also support uninstalled code; extensions to classes not in system will be remembered and applied if and when those classes are loaded.

Store has

- source loading: fairly like file-in but cleverer; it handles unloadable code better and gives you a UI in which to view and work on it
- binary loading: very like parcel loading

Shadow Compilation is like a file-in in that it compiles source, but like parcels in that it compiles elsewhere, creating a mirror of the namespace hierarchy to ensure that variable bindings refer to the right thing. From the user's point of view, everything works as before but loading is more robust. (Envy calls this Atomic loading. The main developer dislikes the word 'Atomic': does it mean 'cannot be divided' or does it mean 'causes huge explosion'? :-). The support code for this was included in 7.4.1. It is all in 7.5. but not turned on as there are still some serious bugs (you can execute `useShadowloader: true` if you are willing to risk them).

Q. Reaction of system when an error happens? Alan demoed by loading a version of AT Parser Compiler that he knew would not load without the parser. It caused a walkback. He closed the debugger and showed that no AT Parser Compiler code had been left in the system.

Q. Can hook for nightly builds. Alan and I answered: yes, you can build overnight, can handle errors and react (by increasing memory and restart appropriate context or whatever is appropriate to that error) resume or quit with log as appropriate.

Q. Impact on speed? Superficially it looks faster but no benchmarks have been run. Speed enhancements were not the aim. (However a speed gain was achieved by only flushing the change log at the end.)

The merge tool has been improved. It offers a hierarchical view (package to class to methods). You can hide image-only mods to see only changes made from the common ancestor in the version you are merging into your image.

Q. Can you choose the base ancestor instead of always going back to the latest common ancestor? No but Alan thinks that merging into the image would handle the situation correctly (he will check). There was more discussion of this at the BoF.

The merge tool also hides full names by default as being rarely interesting. The icons are better and there is help on them. Operations can be done on nodes in the hierarchy (e.g. merge all for this bundle/package/class). Class extensions on renamed classes are handled better, detecting and advising you of method remove-add pairs that look like just a rename of the extended class. You can zoom the text view as in the standard browsers.

Q. Property handling? Much as in old tool.

StoreGlorp (c.f. Niall's talk) gives a better object model and handles changes to the schema better (Store complains when fields are renamed or removed). Because Glorp runs in other dialects, you can use it to talk to Store from other Smalltalk dialects. StoreGlorp provides replication: copying code between databases without loading it. It can also publish loaded code and Alan has been using it to publish his code for the last 3 months. It can load and compare and merge (the work of linking to the merge tool is incomplete; it is integrated but does not yet give the right answers). It offers Envy-like class versions (in rough code that is none too fast but it works). The Store workbook has useful expressions to guide you towards writing your own ad-hoc queries as needed.

Autoreconcile walks your image and detects which versions of database pundles are in your image and reconciles them all to the correct versions.

Projects and Streams addresses configuration management. This work is still in an early stage. The team are no longer yelling at each other about what is needed so now you the users are invited to do so. The aim is a higher level than pundles.

- One part is a tool that does not just show a huge list of bundles and packages; Vassili will work on this.
- The other part is how to manage and deploy.
  - A motivating use case is VisualWorks; the components are all versioned in Store but how do you find the December 01 release build of VW in terms of all the components which cannot all be loaded at once (loading all would take too much memory and in any case some are incompatible and cannot be loaded together).
  - Another is wanting to load just some components but always in the context of an overall project of many more components. These projects should be deployable in several different ways and must be able to include non-Smalltalk artefacts.

Another aim is to have explicit branches, not just relying on conventions for naming versions (e.g. 1.3.4.2). The aim is to have explicit branches, which will be called 'streams'. Then you have an explicit notion of synchronising. People who come to Store from Envy miss the idea of an open edition; people release into an open map and you update by reloading. However this destructive modification has problems. Explicit streams will have a version history, unlike the finally-versioned Envy map that only knows its start and end state (snapshotting in Envy did something like this). This will enable much more automation of merging.

Advanced Object-Relational Mapping with Glorp, Alan Knight
Alan will use a Ruby on Rails metaphor to motivate his talk. Glorp is an open source OR LGPL(S). Ruby on Rails comes from the school of opinionated software; you can go very fast where the rails go; you cannot go anywhere else. It is a reaction against J2EE/.NET ideas. Ruby is a scripting language that uses many Smalltalk-like tricks.

For persistence, Ruby uses ‘active records’; a very simple object-relational framework. Active records and Glorp are very different. Glorp has explicit meta-data for what the classes are, what the tables are and how the two connect. Active records use conventions; for example, class Person expects to map to a database table called People (Ruby speaks English only :-).

In Glorp you have a single broker, the session, which you ask for the objects, which does the implicit writes. You can have multiple sessions. In Ruby, the classes are the brokers. There is no object identity and only one database session for the whole app. Writes are explicit.

Glorp’s typical use case is that you have 300 classes and tables, neither of which you can change, and you must make them connect performantly. Active records aims at simplicity with minimal metadata, very fast to get going but very restrictive.

Alan wondered if ‘Hyperactive Records’ could be provided in Glorp to give the same speed of getting started but with a graceful transition to the more complex real world.

The first aim is to read the database schema and map. This is now done for Oracle and PostgreSQL. Most databases expose the schema as tables. This is a standard i.e. different but similar for every database. Thus Glorp can read these information tables and map them to an information schema.

For database fields, matching is simple; create a direct mapping between the field and the column, except for primary key. In Smalltalk, this is a boolean. In the database, the primary key has a one-many relationship to those columns that make up the primary key i.e. a two level column with a subselect we want to map to a boolean. We therefore use a block (common in Glorp) `[:each | each primaryKeyConstraint notEmpty]` where nonEmpty does the subselect and primaryKeyConstraint is less complex than it looked on Alan’s slide (which it filled) and does the joins to discover the primaryKey-contributing fields. `beForPseudoVariable` lets Alan avoid adding a primary key instvar to his Smalltalk schema while still letting him refer to it in queries. Subselects map to Smalltalk collection protocol: `isEmpty`, `nonEmpty`, `select:`, `anySatisfy:` and so on. Thus any simple Smalltalk-like expression can be converted into very ugly and complex SQL without the developer having to worry about it. By the time foreign keys target fields and source fields, the generated code will have five nested joins, giving you even more reason to hide it behind Glorp.

This has let us do something complex without changing our schema but it has involved reading only. Writing has its own challenges.

Glorp uses `descriptorFor*` methods on the classes to hold their table metadata. So active records for Glorp needs to create these based on starting conventions. The Inflector in Ruby on Rails does the mapping between singulars and plurals, etc., and was very easy to port into Smalltalk (there were a couple of Regex expressions that Vassili’s Regex code did not handle in the same way but they were easy to fake). RoR also has hints

(hasMany, hasAndBelongsToMany, ...) that he handled as class attributes.

Alan brought up the Glorp workspace: it is aware of the Glorp session, etc., so you can trial expressions in it.

```
session
  beginUnitOfWork;
  register: person1;
  register: address1;
  commitUnitOfWork.
```

He showed the volumes of SQL this generates scrolling on the transcript. A more RoR-like API is `session save: person1`. In RoR, there is only one session. In Glorp, `ProcessEnvironment at: #glorpSession` associates a session to the thread running this window manager workbook.

You can modify on the fly e.g. send `byExclusive` to the address attribute mapping so it will be deleted if the person whose address it is gets deleted. RoR cannot handle a tree node connected by a table to another tree node. Glorp can; he demoed retrieving tree nodes by primary key.

He then looked at doing this in a web context. Seaside examples include a sushi store demo. Alan made it persistent via Glorp by writing 11 methods and two new classes WAGlorpSession and the new abstract WA superclass that all the demo classes derived from. He had to rewrite `findItem:` to replace `match:`, sent as a method, to code that was translatable to SQL. He got a new session. He also made `allItems` get a VirtualCollection: this defers retrieval and creation of a real collection until the full query is constructed; select, collect, etc., methods sent to it do not immediately retrieve the collection but instead are stored to modify the final query. (He had the usual demo hiccough but Seaside let him get the debugger, change the offending method to `commitUnitOfWork` and proceed.)

He can write deferred tweaks to generated mappings, e.g. saying to change the type in the image of a mapping from String in the database to an integer in the image, and these will be applied when the mapping is generated.

Alan generated a schema from the Cincom HR demo with RoR plurals true and showed it finding the schema (with one case where plurals rule was not obeyed in the schema).

Q. Release status: preview.

Q. Use for taking one-off datafills? Yes that is one possible use case (may be large hammer for small nail but if you have little time and effort to spare, a large hammer may be what you need).

Customer Experience Reports

Exploratory Modelling, One Team's Approach at SAP, Heinz Roggenkemper, SAP Labs

They are a team without direct product responsibilities. Their task is to look at areas where there is mismatch between customer desires and what their products can do today.

Customers need to adopt best practices just to keep up but they also need to differentiate themselves from their competitors. The second task requires creativity, fact and need finding, holistic thinking and ‘design’. When software is involved, there is rarely one best design. Design is a complex and iterative process. They practice a ‘design-led’ software engineering process.

It starts with fact-finding from users. In the past they thought putting developers in front of customers was the way to do that but this task needs skills that are not conventional software engineering skills. When users tell you something they may be projecting from what they already know. You need to put the users’ remarks in context.

The second step is to debate this user information in a conference room.

The third is iterative prototyping. They start with a very rough UI design with some functional code behind it in critical areas.

Finally they stay engaged during the formal creation of the delivered product.

Example: invoices in accounts payable systems. Do invoices get paid twice? Can the company recover them. This is a serious business. Service companies make a business out of recovering such invoices. “Just let us scan your data and for every dollar we recover for you we will charge a modest percentage.” They built an ERP system to discover such duplicate-payment invoices. They quickly built a working prototype that analysed large numbers, ranked possible duplicates, etc. Customers were very happy (and sent letters saying so). They themselves were less happy; they did not feel they had fully understood and so fully solved the problem.

At that point, they got a call from Georg, “Have you ever thought of doing something in Smalltalk?” They honest answer was “No” but they decided to try it for this problem. A three week joint project with Georg Heeg made surprising progress - so much that they were past where they had best hoped to be in less than three weeks. After two weeks they felt they understood the data a lot better. Why? Because in that short time they had gone through many iterations.

In the review, he asked his team, “OK Georg’s people are good. Could good people have done the same in our environment.” The answer was no, because it cannot iterate as fast as Smalltalk.

Then they looked at supporting shipping managers in selecting carriers, service products and service options. The data that users base their rules on varies greatly. They implemented a prototype in Smalltalk, benefiting greatly from the speed of iteration between customer visits. On becoming a product, the rule maintenance was done in a Java front end but the rules were run in the Smalltalk: the prototype became the deliverable.

Heinz then handed over Georg for his view of the work. He looked at the

duplicate analyser. Naive rules don't work. Suppliers who are paid twice often do not complain :-/). The first thing they found (usual for non-Smalltalk customers) is the classic schism of computing: thinking in data and in processes (see Georg's dentist example in his Smalltalk Solutions 2006 talk). So they started modelling predefined notions without considering data. What is an invoice? The answer is *not* some rows in a database. So they transferred the data into this model and verified it thrice a day. They modelled and remodelled a duplicate invoice; the users found they could control the concept of similarity "as if with a joystick" giving a high-speed iteration of the model. They found more duplicate invoices.

The carrier problem started with an inconsistent table of rules (some carriers care about weight, some about volume, etc.) The rules were very haphazard. As before, there was confusion between rules and tables. They implemented a prototype for rules and iterated with the product manager and then with short cycles (two a day) to correct and to make more precise.

In 1807 Thomas Young used energy to mean the product of the mass of an object and its velocity squared. He went beyond tangible objects to define a new concept. The idea of energy was key to industrialization. (The idea that capital must not decrease might be called the second law of thermodynamics in business :-).

Smalltalk is the best environment. Why? Because every notion can be modelled in Smalltalk. Georg has been doing this for 23 years and even smalltalk's enemies have never denied that it is the best prototyping environment ever. Today, the gain is that people are also realising that Smalltalk also makes a fully functional product without loss of quality. Finding the right models is key in business.

Heinz then resumed to talk about key success factors in SAP. He looked at some issues that newcomers to Smalltalk can meet.

Smalltalk's need for its own environment is sometimes an issue but it was a plus point for them; they could easily put it on a laptop and have the whole thing available when travelling, in meetings, etc.

Prototyping in Smalltalk and then building in another environment does in fact make sense in this case; it is very wise to get the model right before moving to the slower-to-iterate environment, which is used by many customers and will live for a long time. (Niall: this is an example of my Smalltalk Solutions 2005 talk on 'first make it run, then make it right, last make it fast' where 'last make it fast' can mean trading refactoring flexibility for a range of factors - deployed platform compatibility, organisational convenience - as well as or instead of speed.) Thanks to understanding the problem, they can choose the most suited of their existing rule engines to use and populate that engine with the right rules.

They will not use Smalltalk for everything. They will certainly use it again (and again and again). Other SAP teams are showing some interest; the learning curve may delay their take up.

Introducing SAP/NetWeaver Connect for VisualWorks, Rolf Ehret of SAP and Taylan Kraus-Wippermann of Georg Heeg
A year ago they started, following on from the carrier project Heinz and Georg described, to integrate their Smalltalk rule engine to the SAP system. Since SAP use Smalltalk for exploratory modelling, can SAP customers profit directly from this by using Smalltalk for extra functionality.

Clients are logical systems that share data. They wanted to integrate VisualWorks Products with SAP Solutions, and thus enable backend functions powered by Smalltalk. They used SAP-endorsed integration technologies: WSDL/SOAP, Remote Function Calls and eXchange Infrastructure Adaptor. RFC is a standard for accessing different computers and destinations.

NetWeaver Connect makes Smalltalk a first class citizen in SAP NetWeaver. Customers can use Smalltalk as easily as Java and ABAP, and the tight integration means they can use Smalltalk modules with modules built in these other languages. Connect talks via RFC between VW and SAP NW. It uses an enhanced VW web services model. Service execution is session-based.

They showed an example of planning truck loads. Routes may be seasonal (the shortest route uses a pass that is closed in Winter). They connect the SAP order system to a VW transportation zone system. SAP/NetWeaver modules get the order and finds the business partner, then VisualWorks gets the transport zone for that order and uses the transport zone to plan the truck loads. They opened an SAP window and searched for a business partner. (Usual demo hiccough - a VW debugger timeout warning; see below.)

The algorithm to find the transportation zone was complex and would not have been easy to do in SAP. It needs the order and also the business partner so a second server, to get the business partner from that SAP component, was needed (that was what timed out while they were demoing).

They opened VW and showed an enhanced web service menu. A ServiceModule acts like ApplicationModel to let them create modules with services. He showed the business partner demo service - implemented in a method that creates the service object and demoed (usual demo hiccough; they had left in a breakpoint?). The response was the business partner object.

You can model a service and then store it in a WSDL file. A SessionRepository manages Sessions to the BusinessPartnerModules which talk to BupaWebServices (abstract class subclassed to proxies and implementations). Service classes are generated from WSDL. Service modules configure the service and maintain their state. It is a symmetric approach (Smalltalk can call or be called by other SAP modules) and the tool support makes usage easy.

This will be available in VW7.5. Future work will provide access to SAP BAPI repository, automatic access to RFC function definitions (must know and type attributes by hand today) and access to the Enterprise Service Repository. Integration with the XIA will support business process management.

Smalltalk's strengths are now available to SAP's 34,000 customers. 500 SAP partners can replace a white space in their SAP business map with certifiable applications built in VisualWorks. Thus VW is now a first class citizen of SAP.

Q. Transactions when calling and being called? Yes, there are two transactions but you are controlling their joint commit.

Q. Communication is synchronous? Yes, but this is not a restriction.

Q. Two-phase commit in the future? TBD.

Q (Georg) how many of you are from organisations that also use SAP? One third of the audience raised their hands. A BoF was organised.

Q. Release? It is not on the VW75 CD but you can download it separately and Cincom customers do not have to pay extra for it.

Entity Control Boundary: Architecture Patterns in Computer-aided timetable construction System RUT-K, Jochen Eckert, DB Systems
They are partners with DeutchBahn, the German rail system. RUT-K plans train timetables on heavily used tracks. (When two trains seek to occupy the same track at the same time it is called a conflict. If they are travelling in different directions at the time, it is called a serious conflict. :-) Seven independent regional business units run trains in Germany (and there are also border stations). Each of the 7 has some 50 users of RUT-K.

They are delivering a new release in 7.5.1 next week. A very fat client handles visualisations, and conflict analysis and resolution. The server holds the data.

Excellent visualisations are key to RUT-K's tasks. He demoed. The system is very interactive and uses all three mouse buttons. He logged in, selected the time period he would work in and loaded the track data (tracks, signal locations, bridges, speed limits, etc.), the train objects (35 in his simple demo, 20-30,000 in a real system) and various business objects. He showed the rail network map for Frankfurt. As he moved the mouse over the tracks, popups showed train stations, etc. He chose a part of the network and loaded the train running information for it. He expanded some tracks and showed train paths in one window and train running times in the parallel window below. He showed a conflict in the lower window, a parallelogram of the two signal passing points between the two train running-time lines. He then resolved the problem on the parallel track diagram above, routing it on a different path, and re-timed. (A background business rule finds optimal paths.) The system calculated the new running times and showed

the impact (new running time 4.18 minutes longer than before).

The system supports many ways of visualising the data. He showed some of them, zooming in on a view of very heavily used track and examining and altering train connections, dragging the timelines and invoking subviews and operations. Constructing timetables is a hard problem and can be time critical; a cargo train may demand a timetable to let it start travelling half-an-hour later. Of that half-hour, the timetablers may only have five minutes to use the system to compute the solution.

The persistence framework layer is well encapsulated so both the databases and the persistence layer itself can be replaced if it were ever advantageous. It uses various techniques to minimise network load, e.g. accumulating events.

Q. (Georg) How do these half-hour-warning trains get passed to the actual drivers, etc. and how are delays, engine failures, etc., handled. They produce coarse-grained views that are passed to the outside world via a special file-based interface. Many other train systems use this interface and this is how the data is passed to those who need it. The Operation Centre has other systems, even more complicated and not in Smalltalk (RUT-k: 20 developers in 3 years, other system: 3 companies, hundreds of developers, many years (still not finished) and using RUT-K's models.) The other system handles the actual conflict of a late train and offers a proposal for solving it.

Between Desert and Jungle - On the way to an efficient and maintainable test framework, Torsten Happ, Uwe Liebold, AMD
AMD build smart micro devices. You all buy AMD products. CEI baseline is their project for automating chip production. Advance Micro Devices was founded in 1969. They have 15000 employees world-wide. Their HQ is in California but 79% of their sales are international. Recently, memory and microdevices were split into two companies. Uwe works at AMD in Dresden; the factory has been quite a success story and is ranked as one of the 5 best employers in Germany. The site is large (3000 AMD, 3000 in other companies co-located).

Semiconductors is a fast moving industry. Automated precision manufacturing is one of AMD's core competencies. APM is a large suite of components that effect the 800 process steps, taking 2 - 3 months, that create a microprocessor, starting with a wafer. 700 such wafers are started every day. Wafers in process have great value invested in them so downtime is very low; only a few hours per year. They have to have an hour downtime every year just to handle daylight saving time - please remember this next time you write software. :-/)

Uwe works on automation. The Factory Control System has many Equipment Interfaces. Their generic EI is customised for the many specific EIs that connect recipe management, data mining, etc., via the many protocols used: file, corba, MQ, HTTP, SECS. This runs in VW7.2 and has 3000 classes, 45000 methods and a 27 Mb runtime image.

Recently they put much effort into expanded their testing and are seeing much benefit from this. They can test everything except the actual equipment which they must simulate in Smalltalk or via an emulator supplied by the manufacturer. Their control process is very strict so changing on the fly is not done. Each baseline release must go through a manual testing process. They have been developing SUnit tests since 5i.4.

Running all automated tests before integrating to the baseline sounds easy but in fact at starting the tests were not well enough maintained. This is now addressed. Now a CEI baseline is never released before all SUnit tests are green.

They have developed a matrix test to make it easy to test that given input pin values return given output pin results. A table maps set to get vectors; the test applies these to the component. He demoed a test passing, then failing when code was changed. It is very useful for straightforward components but there is a combinatorial explosion if too many parameters are needed for the component. The initial set up of a large test matrix can be very expensive.

The RuntimeTestRunner runs the tests in their stripped GUI-less image, reporting any failures. In future they will refactor SUnit tests to use a uniform set of test resources, especially as mock objects. They want to be able to inject errors.

Their new process is agile with short cycles.

Q. On the fly changes during debugging? Our apps are headless so they rely on their detailed error reporting to let them analyse production issues.

Complex MultiMedia Applications: Secure Deployment for Mass market, Andre Schnorr, Cognitome GmbH
Cognitome is a startup in Hamburg, Germany, specialising in desktop apps for professional music production and education. Their customers are ‘prosumers’: some very professional music composers and many more enthusiastic amateurs. Hamburg is a ‘silicon valley’ for professional music software world-wide with Apple, Steinberg, Ableton, Native Instruments and MAGIX. Such music software retails for from £300 - $1000.

Their product is called Music Prototyping Studio. It is the first product in the world that can understand and reinterpreting musical performances. It visualises the theory of harmony in the Harmony Navigator. It aims to be to music making what 3-dimensional modelling is to physical design.

He then started the product. He started some music and then used the visual harmony display to analyse its harmonies. He then brought up the clefs and showed dragging and dropping note sequences to revise and play music. These scores can be loaded from any music on the net. He played various tunes, including some Bach reworked ‘for a more contemporary sound.’ If the client asks you for an advertising tune “in the style of Bach” then there you are in four minutes.

Typically programs like this take 12 -16 years of effort. With Smalltalk the first two products were completed in 4 years (there was also some 3 years of underlying scientific research not included in this figure).

CxStates: a dynamically defined state model not based on the state pattern, Alfred Wullschegel, Swiss National Bank
(See Alfred's talk at ESUG 2006 in Prague for a prior version of this.) They wanted to use metadata rather than classes for states and to communicate with the outside world easily. The ANSI event model guided the implementation. CxBaseState defines a state with name and transition table. CxTransitionEntry defines transitions by symbols and the new state. It can have transition actions. CxActionSequence and CxAction define these actions for execution of methods in the 'outer' system. CxStateEnsemble holds the model. Transitions are delegated from the base state to the transition and finally to the action. The receiver can be polymorphically set during this process.

Their environment needs a rapid evolution of state models as their customers discover new requirements. They built a model with 36 states, 42 transition symbols and 415 transitions. It was very easy to make errors but also very easy to correct errors. They need to modify StateEnsembles from time to time, which they do by adding a new state, by adding a new transition symbol or by changing transitions using become: (important because the base states are wrapped into larger 'states' that maintain the base environment objects). It is in production supporting workflow of Swiss National Bank handling messages from all the banks that must supply data to it.

During the last month of testing with users just before and just after going into production, they made 6 major changes in the model. This meta-programmed model would be hard to implement in a non-Smalltalk environment.

Q(Georg) When are state machines like these appropriate and when not? Apart from workflow, which was their application area, Alfred is not sure. They chose it to meet their practical need and have seen they can use it for three other small problems to date.

1) An ITIL-compliant CMDB Solution (2) Standard Software (3) Developing Standard Software for Different Platforms, Uwe Danzeglocke, Steynmayer Net Intelligence
He sent in three possible topics and saw that they had become a single title for his 15 minute slot. So he will talk briefly about all three. Steynmayer have used Smalltalk since 1994 and have 100+ customers.

IT Infrastructure Library is a best practice idea from 1986 and is now the basis for ISO 2000 (BS15000) so if you want the compliance certificate you need to know what it is. The Configuration Management DataBase stores the network data MACs (Moves, Adds, Changes), etc.

Standard software for controlling ITC networks should document all

network types (LAN, traffic light network, ...) and all workflows that these networks require and asset management for all their nodes (computers, traffic lights, etc.) They therefore store all this data in a single universal database. Steynmayer have various products: Cable-NSM, Task-NSM and Logic-NSM, and Compact-NSM for small enterprises. All their products live in one image and share one data model; licence keys determine which modules function for given users.

They migrated to VW7 and a customer said no, make it look like the last one (VW3) so they had to provide a VW3 look in VW7.

They have many interfaces to other programs (GIS, ERP, CAFM). A colleague used their program to model the London underground so he could compute routes when he visited.

They provide very configurable products but do no customer-specific development. Instead they listen to their customers and set their roadmap accordingly. Then they concentrate on delivering the expected features.

They use an Oracle database. Multi-platform delivery is very important to them so VW is valuable here too.

Web Techniques and Experience Reports

Web 2.0 Seaside, Michael Bany, Cincom

Michael Bany works for the European Professional Services and Support group; his office is in Geneva. Seaside was written by Avi Bryant of Smallthought systems. Two years ago, a customer wanted a web interface and Michael looked at Seaside. The customer found Seaside very easy to use and they were very successful.

Lukas Renglii gave a presentation in Prague recently at the ESUG conference; Michael's presentation borrows from his.

Web 2.0 has a big constraint; it must work in IE. Therefore it has to be based on old technology. Javascript is ten years old. Web 2.0 aims to exploit some features that have been in the browsers for a long time; it is not a big bang, just a lot of small bangs, using XHTM 1.0, CSS 2.0, Javascript, XML, RSS, HTTP.

Seaside supports all these. Seaside lets you say it all in Smalltalk; no mix of HTML and Smalltalk in your code. It guarantees XHTML conformance. Seaside's rendering metaphor is of a canvas painted on by brushes.

Michael then demoed. He brought up the XHTML generation workspace: it showed the writing panes, the rendering code, the generated HTML and what it looks like. `renderContentOn:` is the method that renders Seaside code. Its argument is the canvas on which you paint. You ask the canvas for a brush and you then send it methods to make it paint. Michael showed the code and then applied style sheets.

He showed how blocks are used to nest HTML elements. He then changed

to an anchor brush and asked it for a callback, showed a text input field and how to echo what was typed in it elsewhere in the page, etc.

```
html anchor callback: ... with: ...
html form:
  [html textInput
    value: data;
    callback: [:v | data := v].
html div: data.
...
```

He then showed real Seaside applications written in VW. The developers wrote all the Smalltalk and hired a professional to write the CSS, and that is Michael's advice; don't do the CSS yourself, get an professional to do it. He showed what the app looks like in another style. He then went to the Zen Garden website and showed a few styles. There is also a Swedish guy who publishes open source CSS demos and does SeaChart demo styles (SeaChart is an open-source Seaside goodie for adding various graphs and effects to your Seaside application). He then showed CSS styling and then exploiting Javascript in your Seaside demos; fade-in for slideshow. He went to the slideshow Smalltalk code; add the slides to the container, implement `children` method to return them. Another example was a Smalltalk app supplying numbers to Javascript that ran in the browser, not on the server which only downloads the values.

Ajax is advanced Javascript that sends requests to the server. It sends standard HTTP requests and gets responses, thus allowing update of pane elements rather than updating the whole page to make a simple change. With Ajax, you can have the client do trivial work and the server send just the data needed. Ajax has several libraries of which Seaside uses two: Prototype library and Scriptaculous. So Ajax can make your UI look much more responsive. Be aware that the Ajax libraries may slow the initial load of your page (Seaside caches as much as possible to minimise) and it uses more browser features so some browsers may fail on them.

The Seaside download includes the latest versions of these libraries and its script.aculo.us support hides the Javascript for you; you can say it all in Smalltalk (but you should be aware you are using Javascript and know its features to use it sensibly). Michael showed Smalltalk code that generated Javascript, not HTML

```
html effect shake
```

generates

```
new Effect.Shake(this)
```

He then showed how to shake someone else instead of yourself :-), pressing a button to shake a text field, then changed the frequency and duration of the shake. Then he showed using `onClick:` to make events such as clicking on widgets cause responses.

```
html updater id: ...; callback: ...
```

lets you update an element with some data. He showed a simple example then one that also removed the prompting button from the page after the update. He demoed causing visual effects, toggling values, scrolling

sliders, drag and drop, all without having to go to the server.

He then showed the Seaside class browser application without and then with AJAX. The AJAXified one was noticeably faster to update as you browsed, generating much less server traffic.

Debugging in Seaside will usually be done in the Smalltalk debugger. If you need to debug Javascript use Firefox and Firebug even if that is not your main target platform; its debugging is much better.

Comet is another technology that pushes info from the server to the client, e.g. to update stock prices in a display. Comet uses server push onto a long-lived HTTP connection. He brought up three counter apps and evaluated code in the Seaside inspector workspace; comet updated all the counters. He then demoed a comet chat application, using push to send an 'Administrator: please logout' command to all in the chatroom.

Web Security, Martin Kobetic, Cincom
Martin created a simple web server (see his slides for the code), viewed the response - some HTML that displayed hello world - then focused on to authentication. Certificates from a site are signed by someone else: a third party whom the user trusts; that is how they work. This third party is a certificate authority.

```
caKeys := DSAKeyGenerator keySize: 1024.
caName := Name new O: 'TrustMe'; L: 'Palermo'; C:
'Italy; yourself.
ca := Certificate new
  ...
  issuer: caName;
  subject:caName; "these are usually the same"
  ...
publicKey: caKeys publicKey;
    "never use privateKey in cert, of course"
  forCertificateSigning
    "what is the purpose of this certificate"
```

You sign a certificate using the private key

```
  caSignUsing: caKeys privateKey
```

The signing *by someone trusted* is what matters; the certificate as such is only a convenient way of packaging all this. Server certificates have a similar protocol. `forKeyExchange` lets you establish a common key between two parties, which is what a web server usually wishes to do.

Public keys are embedded in certificates. Private keys must be stored and retrieved. There are standards for this.

```
...
[sKeys privateKey writeOn: file password: pwd]
  ensure: [file close]
    "openssl pkcs8 -inform DER -in server.pk8"
```

or

```
[sKeys privateKey writeOn: file] ensure: [file close]
  "openssl x509 -inform DER -in server.cer -text"
```

Martin showed the list of the really secure cyphers:

```
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_DES_CBC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
```

and explained how, in ssl, to switch to HTTPS via contexts

```
sCtx := SSLContext newWithSecureCypherSuites.
sCtx certificate: sCert key: sKeys privateKey.
...
```

Using this context, you can replace a simple HTTP transport configuration with an HTTPS transport configuration, copying the old marshaller, etc.

```
configuration soReuseAddr: true.
```

tells it that you will reuse (e.g. on Linux you set it up but might not be available for brief time). He then returned to "Hello World" and got a typical certificate warning that the signing authority was unknown (of course, as Martin has just created the certificate). He browsed the certificate in the browser and then looked at the browser's list of trusted authorities (quite a long list; did you know you were trusting all these people?).

VW does not ship any list. Martin strongly recommends not to have a registry with so many entries. Choose your trusted people deliberately and wisely. Put them in the registry. The registry verifies the chain of trusted authorities that connect you to the original issuer of a certificate being offered to you.

```
registry := X509Registry new.
registry addTrusted: ca.

chain := Array with: sCert.
registry validateCertificateChain: chain.
```

Martin revoked and revalidated the certificate and demoed the behaviour in the two cases. You can populate your registry from the files in Firefox, from the Linux file and so on as you wish. You can also write VW certificates to the Firefox registry and elsewhere.

```
sCtx := SSLContext
  newWithSecureCypherSuitesUsing: registry.
...
```

Q TLS (PLS?) as well? No, just SSL. It would not be hard to do. It appears on the to do list every year but somehow we never do it.

Q. Webtoolkit? This is OpenTalk toolkit. Webtoolkit only uses some of OpenTalk today; the integration should reach this by VW7.5.1 (but Martin noted that Alan smiled when he said that :-).

Do not use a key for multiple purposes. If you use a key for signing do not also use it for encrypting. There is a straightforward attack if this is neglected: an authority gives certificate to John Doe Inc in a legitimate role but they then reissue it to Bank of America in another role. If Peter hands you John's certificate that does not prove that Peter is John. The CN field

is the one that clients compare to see that the certificate matches the server they are trying to reach. Emails can also be validated:

```
transport clientValidator:
  [:nm | '*@oogle.com' match: nm email]
...
```

They do not have pcks12 support yet so use

```
openssl pcks12 -export -in client.pem -inkey
```

(At this point, while importing into Firefox, the usual demo hiccough occurred; it would not accept his password for a while.)

All Martin's slides will be published in the Cincom public repository Presentations-MK bundle contains all his slides and the slide framework so you can download and try his code.

Web Services, Martin Kobetic, Tamara Kogan, Cincom
Tamara Kogan is a Cincom expert for web services who helped Martin greatly in preparing this talk. Martin may not be the prime evangelist for web services but he can make them work. (He can also make screen projectors work - after an epic struggle. :-)

He used the VW Web Services Wizard tool to generate the VW classes and methods needed by the google search web service. The wizard's first step is to show you a workspace with suggested calls for the various services so you can exercise it. You must supply a google key to use the engine; he put this in a shared variable. He supplied arguments, guessing from the method name what the parameters should be ("safeSearch sounds like a boolean - and a good idea") and obtained a WebServices.Struct, a slightly web-clever dictionary. You can then create the classes suggested in the script. With the class created, you now get instances of it returned instead of a struct.

He then read in a WSDL file for a service that reported the current time (rather easier than GoogleSearch to implement during a demo). From this he generated a server class, put `^Time now` at the right point and showed the client receiving the time from the server. He told us to remember to do `server start` and `server stop` when using the workspace, noted that he would probably forget during this demo, so also did `passErrors`.

Now Martin considered an application that you want to expose as a service. No specification exists a priori. He looked at a unit converter application. The wizard leads you through specifying the class and its protocol for the service. The wizard prompts you for result types and parameter types for the protocol's methods. Complex types raise a dialog in which you find their classes. Simple ones provide a menu of simple classes. Types can be defined as collections of such classes. It then asks you to define the complex types' classes' aspects in terms of simple types.

He then generated the server and client classes, and used the workspace to start the server, start the client and get a conversion; it worked!!! He then requested conversion of an invalid unit pair (how many feet in a kilogram) and got a SOAP client error (client error means you can probably fix and

reissue the request. Server error means the server is probably in trouble and resubmitting may not help). (Niall: if you know too much science you can become less certain which unit pairs are inconvertible; for example, in relativistic units, the earth weighs 9 microseconds.) Martin then added exception types (UnknownUnit, UnknownMeasure). He demoed these but found he had forgotten to stop the server, seeing DNU 'address already in use' after he had thrown away his reference to it, so evaluated a expression the expression at the foot of the workspace to find and close all servers.

```
server := UnitConverterServer new start.
client := UnitConverterClient new start.
client getMeasures.
client getUnits: 'weight'.
[client convert: 7.5 from: 'mm' to: 'yd' measure:
'length']
  on: UnknownUnit, UnknownMeasure
  do: [ :ex | ... ].
[client getUnits: 'volume']
  on: UnknownMeasure
  do: [ :ex | ... ].
client stop.
server stop.
```

Header processing was added to their server support to avoid polluting domain code with common stuff such as login and password handling, etc. The wizard has a four-pane page for configuring this. You define the class as usual (ClientAuthorisation in this case) and then add input headers and output headers to appropriate operations. He updated the schema and went back to the workspace. This now had a

```
RefactoringBrowser newOnClass: ClientAuthorisation
```

to browse and fill out the class (which had some methods already defined, e.g. `addInputHeader:transport:`). He then restarted the server (had not stopped it again :-/) and saw first an OK pass with right password then a DNU soap server error with his message. It should have been a soap client fault as the client had the wrong password but the error he had coded was a generic `self error:` not a defined one. He changed it to raise the error BadPassword which he defined in the Wizard for the ClientAuthorisation header, set BadPassword type, created a BadPasswordProcessor and regenerated. The repeat demo raised a BadPassword error.

Q Resumable exceptions? Not easy as several processes are involved. Resumption does not really make sense when you would need to resume a server process that has completed by returning the error. (Jim: and the server might not be a Smalltalk server; resume might make no sense to it.)

How to scale a Smalltalk Server without any Planning, James Robertson, Cincom

Jim started BottomFeeder with no experience of standard web development. To use the term architecture would be overstating the case at starting. All the code is in the public repository; look for Silt*. Early on, he had too few classes doing too much. Running on an ancient Linux box with 256 RAM, he only ever had one crash when an article of his got startling exposure in three major outlets on the same day, and in that case, the HTTP

and comms server failed before the Smalltalk server did. Persistence was never a problem in size (he writes ~ 10 blogs a day maximum so bossing a daily file was fine) but searching got slow as the blog grew. He starting caching the current page just before a major interest post occurred; the caching kept the server alive just long enough for him to push the relevant page into a static Apache cache.

He started by putting more Smalltalk code on the page but then had versioning problems between his page files and his Store versions, so he moved to minimal hooks in the pages with all behaviour in the Smalltalk server. He started with a single class with a singleton pattern. Then Michael Lucas-Smith asked for a blog and after three months of thinking it might be hard to change, Jim tried and found it was very easy. There are now 27 active blogs.

As his daily readers rose to the hundreds he profiled, noticed that each request read the same few files 4-5 times, so added a cache of all the posts on the front page. Each post can have a category and category searches required a search of all posts. He replaced this with a cache of which categories were in which files; only opening matching files speeded things. Looking at his blogs he saw many frequently-repeated search keywords so caching matching files for them also speeded things. He has just noticed that google are now introducing some technology for that so he may replace his by theirs.

If a name is only mentioned once it did not become a keyword, so search still had to search all posts. The server would stop while the same-priority search thread hogged the processor. He used the class Promise to fork off the thread at userBackgroundPriority in a minimal code change. Searches were still expensive but Jim noticed that only newer posts changed; he does not edit old posts and he switches off comments on older than 4 days posts to prevent spammers from cheating their google refs by commenting in your old unwatched posts (for the first 4 days, he monitors comments via the RSS feed). Thus he can cache, relying on old posts never changing.

Spam was a major problem. Trackbacks he just turned off; it is a spam garden. Jim monitors who referers to your site to see who keeps coming back and why - did they come from a rival site? He no longer offers a public referers page because that would include much spam-inserted stuff that Cincom would not wish to appear to present on their page.

No more than 8 HREFs per post is a good rule; comments with more are almost certainly spam. Any site that hits you multiple times in a short space he blocks; it is probably an issue. He has a word blacklist which occasionally sounds false alarms. His referral log was being read every three hours and slowed the server (Apache lets it grow to 90Mb before caching) so he moved it to a separate image run by a cron job.

He never had a scaling problem that was the fault of the Smalltalk server. Every problem was in Jim’s code. He solved these problems as they emerged and fixed them while the server was running. He twice made basic

mistakes (from which he learned) and had to take the server down and reload, but otherwise he has done major changes on the fly without problems (for example, he added CSS styling without any takedown).

Porting Techniques and Experience Reports

Changing the engine while the garage is in motion: Porting to VW7, Niall Ross, eXtremeMetaProgrammers

Long ago I read an article in The Smalltalk Report by Wilf LaLonde called 'Changing the engine while the car is in motion' about how Smalltalk lets 24 x 7 systems upgrade while they are running. If 'changing the engine' means evolving a system to meet new business needs then porting a system to a new platform must be like moving the garage while you do it.

The fantasy scenario for porting is "Sure, we'll freeze all development while you port." When porting to Java or .Net, an infatuated management may actually do this (see e.g. Karen Hope's talk at Smalltalk Solutions 2006) but when porting to a Smalltalk dialect, you are more likely to have frenetic development continuing while you port. So how to handle this?

One commonly suggested approach is to do trial runs then 'suddenly port.' This can work. However such a strategy can face difficulties.

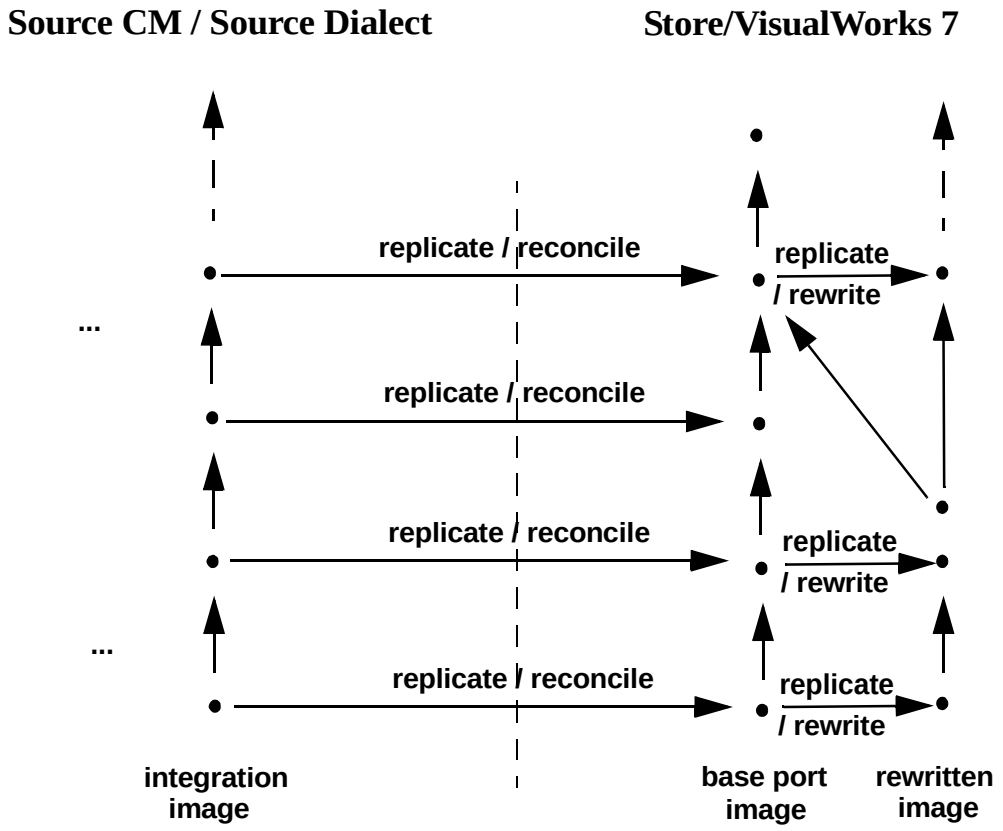
- Technical problems: a large complicated evolving application with associated tools, system extensions and tests can simply pose too many big port tasks to do in a single step. Code that handles essential performance and multi-user or multi-threaded behaviour may be adapted (perhaps unknowingly) to the source dialect and platform, needing serious exercise and study on the new platform. The port team may need to deploy in-situ, test and revert as needed.
- Work interruptions: the frenetic development may suddenly need the port team.
- Commercial or political issues: stakeholders will be ready and happy in their own time, not at the port's convenience, and may change their anticipated schedules with little warning.

Any of these can derail the port schedule unforeseeably: then the port preparation work decays as the system moves on.

An alternative approach is to accept you may not be able to port suddenly and instead arrange port work to be agile, working *with* these pressures, not against them. You can keep the port synchronised with the evolving system by using StoreGlorp (c.f. Alan's talk) to reconcile and replicate each integrated system increment from your source' CM (e.g Envy) into Store on a regular (e.g. daily) basis. You can assist this by minimising code changes through deferring them wherever possible.

I spoke first about these deferral strategies.

"NameSpaces are about *in*visibility." (Georg Heeg, ESUG 2006). When porting from a single-namespace source dialect to multi-namespace VW7, you know that every name in the source dialect has a single resolution, so



every name the port cares about *must* have a single valid resolution in VW7. It is a necessary truth that you can define a namespace (call it `MyApplication`) with appropriate imports to offer the valid resolution as the first found, suppressing unwanted duplicate binding errors with

```
NameSpace>>conflictingImports: key firstFound: bind
  ^self == MyApplication
    ifFalse:
      [super conflictingImports: key firstFound: bind]
    ifTrue: [bind]
```

- Define a package in VW7 with (only) the namespace definition and the above code, and prereqs DefaultPackageNamespaces (use 7.4.1 release version or later, or an appropriate version from the Cincom OR; these have an essential fix for the case of ExternalInterface methods). Assign your namespace as the namespace of all packages the port creates.
- Define global `MyApplication := Smalltalk` in your source-dialect application and rewrite any references to `Smalltalk` in your application to use `MyApplication` instead.

With these changes you can defer all resolution-code changes. Code like

```
MyApplication at: aClassNameSymbol ifAbsent: [...]
```

just goes on working and by using `namesAndBindingsDo:` you can provide VW7 implementations of any `allClassesDo:` or similar methods that are not already there. All your application's methods (including, thanks to DefaultPackageNamespaces, all methods that extend

base classes) will be resolved with respect to `MyApplication` so will find the correct bining for the port. The effect is that the port can proceed as if from one single-namespace dialect to another.

A change you can not so much defer as manage is class definition coding style. Store began life before NameSpaces; it still has class-side definitions and so can still load classes defined in the old style. A few fixes are needed - DatumDescriptor needs `isMeta`, some Store loading routes no longer bother to set the class-side package instVar and this must be reverted - and you need the old class definition protocol, which is in the base image or in FileOut30, or is easy to add (to ClassDescription *and* to ExternalInterface *and* to ExecuteCodeChange). With this, your daily synchronisation can have a loadable intermediate target stage (see diagram above) in which the ported class definitions are still in the old style, simplifying change comparison with today's source, and with yesterday's source and target.

How clean is your application's configuration management structure? Have developers been ignoring warning 49 for years, or worse? In the fantasy scenario, the application builds cleanly in its existing dialect with no warnings. In reality, the build may take two forced loads during which the Transcript shows screeds of warnings, or a living image may preserve an application whose clean rebuild would be a major undertaking. Obviously, such applications will not load and initialize package-by-package from Store. Equally obviously, taking the time to clean up the application in situ before porting would be a frustrating additional cost. By tweaking Store so that the replicated packages have a special blessing level which defers all package initialisations to the outermost bundle, the problem can be deferred and the port loaded in the VW7 environment. Loaded code is always easier to fix and since the repackaging of code is one of Store's strengths, it will almost certainly be easier to track and clean code here than in the source' CM.

I also covered some minor matters. When a loaded Store bundle's parents in two different databases contain different sets of packages, switching between them can confuse the image's understanding of what packages the loaded bundle has; have the fix for this loaded when doing this work. The `MyApplication` namespace/global is an example of the general fact that mapping between classes, globals and namespaces can avoid the need to rewrite code. For example, CharacterEncoderPool is a namespace in VW7; making it a class in VW3 minimises code rewrites in backports to VW3 of utilities that use it and forward ports of tweaks to them.

The central idea is to use Store for Glorp to replicate and reconcile between the source and Store, exploiting the Refactoring Browser frameworks. (Sometimes you can also replicate the base system. You cannot load this of course but it can sometimes be useful when comparing your changes to the source base classes with the cross-dialect base changes.)

Glorp is a framework for connecting SQL databases to Smalltalk. It is declarative, eliminates impedance mismatch and has a natural Smalltalk style of coding, e.g.

```
matchingPundles :=
  session read: StorePackage where:
    [:each | each name = aName &
           (each currentBlessingLevel =
             self replicationBlessingLevel)].
...
session transact: [session register: newPundle].
```

See Alan's talk for (much) more detail on Glorp. Store for Glorp uses Glorp to read and write to Store databases. It is available in several dialects including VASmalltalk and VW3. It is wholly distinct from the Store code as far as the VW7 image is concerned (be aware of this).

The Refactoring Browser is in most dialects. I exploited three of its frameworks in this work:

- RB UI extensions let you shadow-browse code in Store databases from the source image (a BrowserNavigator views an environment; making it a StoreForGlorpBrowserEnvironment enables shadow-browsing).
- BrowserEnvironment framework extensions define what code is to be exported. A BrowserEnvironmentWrapper is a view of a subset (of a subset of a subset ...) of the image. In normal use, the last environment in the wrapper chain is a BrowserEnvironment, representing the whole image. When exporting, it is a StoreForGlorpBrowserEnvironment, representing the whole (source) image and also the target database.
- Refactoring framework extensions enhance the BrowserEnvironment extensions to let you refactor code in an RBNamespace and export it *without* having to recompile these changes in the source image. An RBNamespace inserted at the penultimate position of an environment wrapper chain can presents a refactored view of code in the image.

These were natural frameworks to exploit. BrowserEnvironments are usually the easiest choice for code management work because they are cross-dialect, easy to construct using standard RB functions (the source dialect's code units, e.g. Envy Applications, will usually already have environment subclasses adapted to them) and their UI and tools come free. (Where a dialect lacked a bundle-equivalent or the idea of extension methods, I exploited the little-used MultiEnvironment.) RBNamespace already offered most BrowserEnvironmentWrapper protocol, so it was straightforward to add the polymorphism needed by the exporter utility. I had to add a little protocol to handle the fact that RB environments care who includes a class whereas CM systems like Store care who defines it.

(In VW7's RB, environment wrapping has been eliminated; I restore it via

```
RBProgramItem
  RBNameSpace
    RBRootNameSpace
      RBNamespace
```

where the final class in the hierarchy is added by my extending utility to wrap browser environments. Note its similarity to the second class' name. Keeping the VW7 final class' name the same as the equivalent base RB class of all other dialects simplifies writing cross-dialect scripts.)

With this machinery, replication and reconciliation can be done. The utility adds suitable environments to the replication list and a subclass of the standard StoreGlorp replicator lazily creates StoreForGlorp pundles from them during the background replication process. Aggressive reconciliation ensures that the daily run captures only new code from the source.

Once replication and reconciliation are working, the next stage is trying to load and run in VW7, and resolving the issues that arise. Minimising code changes during the port was stated to be the ideal. Perhaps it would be better stated as minimising *the effort of managing* code changes. Rewriting code to be dialect-neutral is as usual the ideal, but not always realisable, solution: whenever possible, feed back the changes VW7 needs to the source image (via the shadow browser or the FileOut30 utility), test them and integrate them. However sometimes these changes would not give the same behaviour or would not be performant. Then you can capture the changes as RB changes and run them in the image before replicating. This gives easy code browsing and is fully reversible (executing refactoring changes returns their undo change equivalents). However, some changes that VW7 needs may not compile in the source image, or if they could they would break it. These can be run in an RBNamespace that the replication environments wrap.

While discussing the above, I demoed using the utility bundle to replicate itself from VW3 (non-Envy, to show that the utility did not depend on a source CM system) to VW7. I then reran, demonstrating that replicating identical code gave an exact reconcile, then effected changes and reran, browing them with Store's comparison tools from a VW7 image. I also browsed a database of the VW7.4 base reconciled to the VW3.1 base, showing how you could use it to see how an application's changes to the base might interact with dialect or version base changes.

The experience of rewriting code between dialects suggests refactorings for those dialects. It makes sense to capture as many dialect differences as possible in run-once refactorings instead of handling each individually. Some are trivial and specific (rename this class, that method) while some are general. I walked through a simple example: a refactoring that converted all class names in UI specs to qualified names, useful when mapping utilities from pre-namespaces VW3 to VW7.

To replicate and reconcile 1000+ application package environments, each with an average of 15 classes, 225 ordinary application methods, 6 *large* application methods (can be > 64k), and 2.5 base overrides / extensions can complete in 3 hours when running from a performant client machine to a dedicated Store database server machine. The same replication takes over 6 hours when a low-spec machine with other tasks is both the client and the server. Thus the natural frequency of use is overnight replication and build.

Summarising, a port needs to be as agile as a standard development or more so. Large complex systems have large complex ports. The customers are the developers, not the end-users, i.e. the porting activity is at the bottom of the food chain and must endure all the changes created by higher layers.

When porting to VW7, you can deal with this by exploiting extensions of the Store Glorp utility to keep your port synchronised daily. Since you are starting without namespaces, you can and should keep them invisible till the port is finished. Solve problems where and when it is easiest, i.e. in whichever image or CM system best knows what a given rewrite needs.

(I spoke briefly of test style techniques that assist porting. All that material, much better presented, is in my Smalltalk Solutions 2007 talk; see below.)

Store more like Envy, Jan Lukes, Gehe

They are porting from VW3 to VW7 while still developing, so hundreds of changes will occur. Coming to Store from Envy, the notice that Store lacks the 'always on' state, versions of classes, version colouring, the application manager browser and some comparing tool aspects. They could modify Store protocol or just modify the tools. Their solution was to publish automatically on accept, creating a sub-version. They use one package for each class. They hide packages completely; thus bundles are more like applications in Envy. They created a hierarchical comparing tool.

He showed the old and new (coloured to show state) version browser, then the published items bundle, with one more pane on the right showing the class/app contents of the bundles (its layout was inspired by the Envy config manager). They use the RB's first two panes to show the application manager, with bundle version displayed by its name as in Envy. He then showed the old comparing tool, which of course did not suit a style of one class per package. They again modified the RB so the code compare tool did the compare with the left two panes showing the two lists.

Q(Georg) Config maps and override-handling? No and they've not looked at it particularly.

Q(Niall) Use these tools but not have just one class per package? Yes; the system packages have many classes of course and can be browsed in these tools. (Niall: I know that many people coming to Store from Envy miss class versions, as I did at first, but I found it was better to generalise my CM approach, and so discover that what I thought was basic was in fact just an adaption to Envy's model. Methods and class *definitions*, not whole classes are the basic units of an ideal CM system. Higher level groupings are also essential but groupings based on classes are not particularly fundamental; indeed, I find Envy's limiting - Store's too, of course.)

History and Process

Small Matters can Matter a Lot, George Bosworth

(See my write-up of George's talk in my report of Smalltalk Solutions 2004 for George's background. Since I arrived during this talk, I only managed to take a few notes during it. Thus the following paragraphs give just a few choice remarks that I culled from it.)

Back in the early 90s, IBM developed a product in Digitalk and demoed it in an announcement. They had to restart during the demo so the Digitalk splash screen appeared. Digitalk were promptly called by the press: "IBM

is displaying Digitalk in a major announcement; what's the story?"

Necessity is the mother of cost-effective invention. In one printer project they faced a major disaster when suddenly paper had to be rotated 90 degrees at one point in the flow a few days before release. Moving paper is a hard task; paper is not aerodynamic. An engineer built a trivial solution out of a pin and a tube that evening. If the deadline had been 3 months away, they would have taken three months to build an elaborately designed and costlier solution.

Digitalk customers were often 'people who program' not programmers, people who deeply understood their problem domain and found that Smalltalk did not get in the way of expressing that understanding to solve problems.

The Open Unified Process, Scott Ambler
Scott is practice leader for IBM's attempt to become agile; is his group part of the problem or part of the solution? Let us see. This talk is not about Java; to prove this, he showed his talk overview written in Smalltalk: `presenter tellLameJoke. presenter giveStandardWarning.` and so on.

In many organisations management will fight the very thing they desperately want: to save time and money. Scott has some data that might help convince them to stop doing so. Scott is selling ideas and Scott is a firm believer of telling it like it us - one could say he is blunt at times. For example, he sees writing a requirement spec early in a project as a spectacularly bad idea and an indication of bad practice. You should pick those of his ideas that you have a chance of selling to your organisation - unless you like seeking employment elsewhere of course.

Agile development is iterative, incremental, evolutionary, highly collaborative development with just enough ceremony to produce high quality software that meets the changing needs of its stakeholders. It values individuals and interactions over processes and tools; working software over documents; customer collaboration over contract negotiation; responding to change over following a plan. It is not that the less preferred items don't matter; it is just that they do not matter as much.

Agile reduces the feedback cycle. For 30 years we've known that the later you find a fault the harder it is to fix. The agile cost of change curve is flat? No, Scott argued, it is exponential just like elsewhere but short cycles keep you near the early flat-looking initial part of the curve. (Niall: I would also stress the test suite's role in making later refactoring much less terrifying, so I would not wholly agree with Scott in denying the relative flatness of the agile cost-of-change curve versus those of its rival methods.)

The concept of a repeatable process has absolutely nothing to do with success. Repeatable results are what you want to aim for; not repeatable process.

There was RUP. Then in late 90's Scott introduced Enhanced RUP which became the Open Unified Process and there are lots of other *UPs out there. A 3 person team working together for 5 months will have a different process from a 40 strong team working together for years. RUP has often been adopted by companies in a not-so-agile manner but this does not mean it cannot be used in an agile manner.

A method framework has a base process with plugins. People are working on a Scrum plugin and others, so maybe a Smalltalk plugin would be welcome. (Plugins can be sold commercially or made open source; Scott recommends open source but it is your choice.) IBM donated a portion of RUP as a starting point; it has now been much refactored. There is a round table of roles in OUP (like King Arthur's round table - all roles are equal). The various roles in OUP are *not* positions; because you need stakeholders, analysts, etc., in OUP does not mean that a person must try to be just a stakeholder or an analyst, a common mistake in original RUP.

EPF (needs JVM to run; endure it) can be downloaded by process people and used to get and see the plugins. Many people are involved and this is the most active area in Eclipse today. He opened a browser and went to the OUP home page and navigated to the 'run developer tests' task, then to the architecture tasks. People who know what they are doing will never look at this but new people in your company may look at it and others may check it to compare with their own process. OpenUP is free, tailorable and the tailoring tool is also free. It is agile and you can tell managers you have a well-defined complete process. Get it from www.eclipse.org/epf/.

Developers rarely love process. For most developers, process is what gets forced on them. If it is going to be forced on you, OpenUP may let you get in front of the problem and manage it. Large organisations who adopted RUP may have adopted it poorly (a polite way of saying it when waterfall people are tasked with introducing it) and OpenUP may be a way of remediating (escaping from) such situations.

Feedback and BoFs

Feedback from users, James Robertson, Suzanne Fortman, Cincom

Of the categories offered to solicit feedback, the results, in order (most to least), were

- CCom better
- VW as a DLL
- Tools

and more minor

- Support for scripting: minor in this conference but Jim would get different answers at a conference of non-Smalltalkers (Alan, "Is that like the bindings for Vi issue?" Jim, "Anyone sitting next to Alan feel free to strike him hard.")

(Various other offered categories were mentioned.) In the 'Other' category, people had suggested:

- Making Store better and faster,
- Oracle non-EXDI: assume this means direct socket connection like some Java but Cincom cannot because it is a proprietary protocol and Oracle will *not* release it (they have asked)
- New look and feel for OS: we will look at whether Microsoft WDL work can be slotted in
- easier-to-create COM stuff, don't use DLL calls, etc., all about making C connectivity easier (like the first point above)

Alan asked the audience about C++ as well. It is harder to do owing to the less standardised compilers but Swig is being used elsewhere and Ian Upright did a Smalltalk version. Who would be interested? A few hands were raised. Who has heard of Swig; almost all the same hands were raised.

Jim asked who knew about the screencasts he has been doing almost daily. These take 3-5 minutes to describe how, starting from a base non-commercial image, some function can be done, utility can be used, etc.

Q. StarTeams? The star team members are the product experts in Cincom's product lines. At this conference, tell the Smalltalk star team people what is good, what is not, what is needed.

Tom Nies decided two years ago that as Smalltalk was doing so well with almost no marketing effort, what might it achieve if they put publicising effort behind it. Suzanne was hired soon after and has spent 18 months on the road talking to Cincom Smalltalk customers and to Cincom customers about Smalltalk and generally publicising Smalltalk. She wants to hit vertical conferences. Write success stories; we will print them.

George Bosworth was very impressed at this conference to see so many amazing applications written in Smalltalk.

Suzanne values all information from customers, negative as well as positive, and is grateful to those customers who have told her about various situations that occurred even five years ago (and grateful they told her instead of yelled at her) and also to all those who have told her about things that pleased them.

Alfred has learned about the number of people still using VSE. Is there a roadmap for how they move forward. VSE fared ill in the ParcPlace-DigiTalk merger. The VSE team did Parts for Java which was sold to SingleSoftware in Atlanta, who no longer do anything with it. Cincom needs all its effort to support the products it owns. Cincom cannot move forward a product it cannot own. Pollock's match to VSE UI means that migration to VW/Pollock will be a feasible way forward. They have one customer moving from VSE to OS8 and will monitor feedback from them. A company in Europe have talked to SingleSoftware / Seagull about buying the source. In summary, they lack resource and Suzanne spent hours with Tom Nies reading the contracts and remained unsure whether they would have the IPR to do anything even if they had the resource.

Q. The Mac product does not look ideal for those who download it? We are working on these and will have a stable VM, good fonts, etc. It will not look entirely like a Mac product but will be stable usable and similar. We are not ignoring the platform. (Alan) significant usage of Macs in the Cincom Smalltalk team. (Niall) ESUG uses Macs a lot and is growing young people into Smalltalk.

Q. Workshop sessions? Monika is preparing some, deciding which areas and what is needed: introductory or advanced. Some want a workshop in their native language. Some want it internal and customised to their team. Some wanted it but at another time than early December (German attendees miss having St. Nicholas day with their families). Times are tricky: September is ESUG, October is OOPLSA, November is Thanksgiving, December is Christmas, January is NetObjects, February is not a popular month, March is Cebit and so it goes. In the next few weeks Monika will make some proposals.

Q. Since the last CSTUG, have you acquired new users? Yes; we are seeing a lot of growth. Current customers are building new applications and new customers, including new 'name' customers (just last week a product manager phoned her "I have 5 new named customers and two new partners and I don't think you even know these people"). After last year's Smalltalk Solutions that marketed Smalltalk to 100,000 instead of 250, Suzanne got calls and new customers.

Q. Geographic distribution? Some are in Germany (a few new name customers and more new applications), many are in the US, a new name in Canada, India has a brand new customer and Chascart (measures India's energy usage). Giorgio mentioned that India has the problem of keeping workers because Smalltalk is in demand; after a year, a trainee Smalltalker moves to a higher paying company. Jim mentioned the job postings. Jim includes a 3 minute report from James Foster on new jobs in Jim's weekly podcast.

A customer in Europe has just hired 8 new Smalltalkers. One in the US has just hired 6. Suzanne has found herself putting Smalltalkers in touch with customers. Monika mentioned that Helge is a contact in Europe for the same thing so if you are looking for Smalltalkers or looking to move tell Helge. Helge has recently been looking for some Smalltalkers for a Mac application. Heeg mentioned two Universities using Smalltalk, one north of Hamburg, "the professor is here" (he stood up), the other in Kothen, and others are discussing it. OOPSLA had a dynamic language symposium with many young people demoing Seaside and other Smalltalk things.

Q. When will you have one logo? Cincom changed their corporate look and feel last year. Most products just want a corporate logo but Suzanne wants also a Smalltalk-specific logo at events so they still have it.

Suzanne on taking feedback, "I heard this was supposed to be painful. This was pretty easy so far."

Q. Why not every year? (Monika) given time conflicts, she thinks biennial is the right timeline. It may be that a changed format (more customer and partner presentations) would suit annual. So give us those feedback forms.

Someone mentioned that in Frankfurt every two months they have a

it prompts for another image to connect to for debug.

OpenTalk exposes objects via a naming service or whatever. You can then use a regular inspector and via that show objects on the web or wherever.

Early OpenTalk embedded IP addresses in connections so had problems when an IP address no longer fitted your context. This is fixed in VW7.4.1 (start server in DMZ zone, unmatched address advertises external firewall address) but even in 7.5 if you use an ssh tunnel, e.g. from your local office to a remote office, you do double translation of IP addresses and this does not yet work.

There is a load-balancing application that also handles making the servers it load-balances look like a single cluster to the outside world. There are no analysis tools to let you see what is going on when you turn a client-server application into an OpenTalk application (there was a distributed profiler but Florean's MultiProfiler overtook it; it may need rework). Minimising the number of messages is more important than minimising their size. He hopes that OpenTalk is faster than DLLCC marshalling but he has not seen a comparison. A user thought he saw ColorValue passed by reference; Martin believes most colour stuff is passed by value. Martin described an OpenTalk application: trader images communicating via OpenTalk to a server with a cache of portfolios, front-ending an SQL database.

Early OpenTalk was one-way but connections are bidirectional. That was needed when a machine behind a firewall spoke to an internet service that found it needed to send back a request. It is robust to dropping connections.

Q. Compression? OpenTalk has no compression; he assumes compression will happen in the lower network layers it runs on, so it is not clear there would be any advantage, plus some data does not compress well.

The SSL layer can be wrapped round any stream (but it makes little sense unless the stream is a SocketStream). The only significant issue against pulling back the security layer into OpenTalk is configuration: should *every* OpenTalk node playing a server and client role have a certificate?

Q. Multicast? Yes.

Q. Relationship between Opentalk and WebToolkit? We wrote OpenTalk to replace Distributed Smalltalk so focused on CORBA and Smalltalk-to-Smalltalk. It soon became obvious it had many similarities to web services. Different protocols and marshallers were just plugged-in to the generic OpenTalk framework to do functions of VisualWave and WebToolkit. The older parts are still there for some stuff, e.g. VisualWave uses OpenTalk to create the connection but then hands it to VisualWave to use directly, not via OpenTalk-HTTP. This will be replaced with full use of OpenTalk.

Jan Lukes of Gehe remarked that implementing in OpenTalk was very easy if you use sockets. If you use something else, e.g. a serial connection, the assumption of sockets is at several levels so is non-trivial to reset. Martin

agreed. Andreas has also noticed that there is no need to have object tables for simple cases; work on this should be done.

Q. MQ library? No. MQConnect was done by Heeg for AMD. It is a thin layer around MQ Series DLLs. It could be integrated with OpenTalk but only serious customer interest would make it happen.

Tuning is done by setting a low limit (become slow to accept connections) and a high limit (stop accepting connections). There was discussion about transaction monitors. In the end, application logic controls transactions so is there a generic answer? The 'standards' disagree with each other. Discussion with Giorgio and others looked at building headers to effect it.

VSE-to-VW Porting

My notes about the VSE-to-VW BoF at CSTUG in December 2006 were later typed up from rough (and sometimes obscure :-) hand-jottings. There were five groups of participants besides myself.

- Christian Haider, smalltalkedVisuals: see the VSE-to-VW porting talk he gave at ESUG 2004 in Kothen
- Daniel Poon and other staff from Romax, based in Nottingham, UK: see Daniel's talk on Romax at Smalltalk Solutions 2005 in Orlando.
- Debica Insurance, based in Coblenz: I noted down the name of one of their staff as something like Roel Adid (very scrawled in my notes)
- Guy, from the U.S. or Canada IIRC: I can't find his last name and company in my notes
- Jerry Blinton of Caesar Systems: he gave a talk on his system at Smalltalk Solutions 2002; see my StS 2002 report on the web

Debica needed to put their system on the web so ported the model layer to VW and used VisualWAF, working with Heeg (Heeg have also done a port assessment project for an Italian customer, who IIRC decided to wait for Pollock / Widgetry). They had to handle some base class and method differences but it was not bad. They had some Delphi DLLs. They simply rewrote their DLLs as they could not migrate DLL handling between the two (but the DLLs rarely change so one-off rewrite was not a problem).

Meanwhile their VSE system, much to their surprise, ran OK on Vista in an experiment, removing one porting necessity.

Whereas Debica have a web interface, for Romax, GUI is *the* issue. They had thought of porting WindowBuilder to VW but think it might be comparable to the ObjectStudio port in the amount of work. They had also thought of porting some GUI code and displaying VSE widgets inside VW windows. There was discussion of a utility from SmallcomX (a small company near Munich, N.B. *not* Smalltalk/X) that lets them create such widgets, exploiting the directness of VSE windows widget handling and so simplifying their GUI port. The approach worked for wholly self-contained widgets (demo IIRC) but how to interact with subcomponents of the widget from VW remained TBD.

The near approach of some size limits motivates Romax' port. They are pushing 32 bit (must configure old-space to avoid customers seeing OutOfMemory warnings, and are seeing 1 Gig file-outs of model and analysis files) and can see 64bit becoming essential.

They have automatic DLL-conversion via a method that holds a context. These method create input descriptions for DLLs that can be called to create the strings they need. These methods are created by a generator in VSE and work immediately in VW when ported across. They still have some file-reading and copy layer stuff to rewrite (mention of COBOL copy book method) before they port. Romax use the GH change list and FileOut30 to read file-outs and assess differences.

Serialization was an issue for both Romax and Guy. Romax have users with existing files that must boss in to the new system (perhaps with conversion but must not be too clunky and must not be a do-it-all-at-once as these files are large and dispersed and many will never be reused but others will be; which ones is unpredictable). Guy must also reuse existing object files. Romax experimented with SRP but found it was slow and had problems for large files (may have been an understanding or configuring issue). SIXX (XML-based) has no VSE version.

Guy's port is motivated by rivals putting their apps on the web. They would port tomorrow if Pollock / Widgetry were there. Otherwise, there is no motive as “it just keeps working” and they are not yet near any size limits.

There was discussion of the possibility of putting non-VW-graphics within VW windows. Getting the window handle for a rectangle within a window is the key issue. Pollock has form rectangle so getting window handles for that *should* be doable. OS maybe could help. (Other idea: vxSqueak port).

Glorp BoF, Alan Knight, Cincom
(The Glorp BoF was held on the last day and I had to leave before it ended to catch my flight.) Several people listed applications where Glorp might be of use. For example, Jochen Eckert has no love for the persistence framework of DeutchBahn’s timetabling system (see his talk above) and would like to replace it. Gehe have an old VW system using Toplink that they might migrate to VW7 and Glorp if it were supported. Alan thought Toplink would migrate to Glorp easily. Cincom considered declaring support for Glorp in 7.5 but decided against because it depends on DefaultPackageNamespaces; I urged them to support that as well.

Michael Lucas-Smith provided TimedExpiringProxy which can improve many garbage-collection issues in Glorp. Recently, Alan has been improving the mappings. David Shaeffer implemented strategy-controlled mapping of object hierarchies to tables; strategies are filtered (one large table), horizontal (all concrete leaf classes are in their own tables, so joins are not very performant) or a root table pointing to subtables (most like Smalltalk in structure, but writing a large bushy object to some 20 tables can also be unperformant).

Glorp has a two-level sort: topological (based on foreign key constraints) and minimal perturbation (based on instance-level relationships, e.g. an employee's relationship to another employee who is also a manager).

For Oracle, Glorp uses Array bindings to do updates. In Toplink, having the cache distinguish insert and update was one of four configurable policies. If you create the same objects (same primary key) in different images, e.g. from Gemstone, and then wrote them, Glorp would need a pluggable policy to tell it when objects not in the cache were nevertheless already in the databases.

Q. Replication depth? (Don't want proxies that are resolved one by one instead of in a single round trip.)

```
q retrieve: [:eachTrain | eachTrain car asOuterJoin]
```

is good for latency but bad for bandwidth. `useFilteredRead` means that `q read: Customer where: ...` will get the order for all customers at once, not one by one.

Q. Any value in defining views to help Glorp? Alan thought not. It might be easier to describe the mapping and a materialised view *might* be faster but generally it will not help you.

Q. Can you specify tuned SQL instead of the default style? Yes for reads, where it is most needed, not for writes. It's not an especially elegant syntax.

VirtualCollections tend not to be kept up to date, though they could be. Glorp avoids tracking the domain objects (saw the great dangers of doing so aggressively in EJB). A compatibility layer to let existing Toplink or Lens applications use Glorp unchanged might be useful and might need it.

Q. Can you unload things from the cache? Yes. You can remove objects explicitly and you can tone the cache policy on a per-class basis.

Q. Glorp and StoreGlorp port status? Alan last used the VA StoreGlorp port at CSTUG 2004. Niall has worked with it since (found some bitrot, much of which he has remedied). Niall has also fixed and improved the VW3 port. Noone has worked with the Dolphin port for some time. Radislav Hachichack was maintaining the Squeak port and Tod Blanchard said he would look at it again.

Q. StoreGlorp state in VW7? Today the Store schema is brittle because it uses the EXDI tables * -> instVars trick so one extra column will break it.

Other Discussions at CSUG

I had discussions about porting approaches with Jan Lukes from Gehe in Prague (they would like to port an application from VW3/Envy), and from Janos Kazsoki of Cincom who would like to port the Smallbrain utility. I met Troudo Manz again (she was at CSTUG 2004). She would like Glorp for SQLlite. The lack of transactions in SQLLite is a problem for the tests.

Smalltalk Solutions 2007, Toronto, 30 April - 2 May 2007

I spent the days before and after the conference with a friend at Oakville, 20 miles west along the lake from Toronto. The weather was beautiful when I was there but colder and rainy during the conference itself - which was the ideal balance for me.

This year's Smalltalk Solutions was combined with the IT360 conference (renamed and somewhat changed from the Linux World and Network World conference that we paired with last year).

I have sorted the talks I attended into various categories:

- Keynotes, Initiatives and Overviews
- Application Frameworks and Experience Reports
- Coding and Testing Patterns
- BoFs and the Coding Contest

followed by Other Discussions, Follow-up Actions and my Conclusions.

As there were usually two parallel Smalltalk programme tracks (and several others), I could not attend, still less report on, all I wished to see. James Robertson's blog posts cover some talks I missed, as do those of other Smalltalk bloggers. David Buck blogged Thomas Stalzer's talk.

Keynotes, Initiatives and Overviews

Ruby on Rails for Smalltalkers, Chad Fowler

Chad has been in Ruby since before most people think it existed. In 2000, Chad decided to learn Smalltalk after reading 'this feature was borrowed from Smalltalk' in language after language that he had learned. However while waiting for Smalltalk Best Practice Patterns to arrive, he decided to look at Ruby - so much for learning Smalltalk (but since then he has played with it enough to understand it).

When talking to non-Smalltalkers there is much he has to explain that he can skip with us. Up to 2003, everyone in Ruby conference knew each other; less so now. (Tomorrow he will speak at a conference keynote on why this is no longer so; why Ruby, who in 2003 thought they had 'the thing' but the stupid industry couldn't figure it out, has broken out from there to become a hot thing. Slides are at http://chadfowler.com/sts07.pdf)

- Smalltalkers are at a huge advantage in learning this hot technology.
- All of it is what Smalltalk can already do (better in many cases)

So you need to understand why it sells.

Windows has a one-click installer (http://rubyforge.org) and likewise for Mac; others install from source (do not use Debian packages!) in unix-style. The program is `ruby` and runs on command line (so very ugly with respect to Smalltalk "and he has put it on a green-on-black command prompt to emphasise how archaic it is." :-)

Ruby 1.86 creates an AST and walks it so is strictly interpreted. Ruby 2.0 will do byte-code compilation. There has been discussion of porting ruby to run on top of Smalltalk VMs as they are better than the ruby VM (Q. has been done by Brian Davis and others to the 'it works' stage; Dave Simmons also did it).

It used to take all afternoon to install rails. Now you use rubygems (not yet shipped with ruby) from rubyforge.org/project/rubygems. This gives you another program gem that lets you do `gem install rails -y` and suchlike. Thus ruby gem is a package manager.

Q. (Bruce) Tension with Debian project will be resolved? Some people have been working on it for two years and it is still broken. I could give you my opinions on why but you might quote them. :-)

Then you run `irb` to get the 'workspace' for ruby; type ruby expressions at the prompt and get the results echoed back. There is no browser, there is no IDE so the `irb` is the IDE. Ruby is not image-based; any changes you make in another copy of the interactive ruby process will not be reflected here. If you want to find methods available on a string you can do e.g. `"asdf".methods` and they are echoed. "Show this to Java programmers and they say 'Wow!'; show it to Smalltalkers and they say 'So?" Likewise you do `"asdf".class` and `"asdf".class.superclass`, etc. The class of Object is Class, as is the class of Class, etc.; there is no Metaclass.

Rails is a way to get people into ruby. Companies are getting used to jumping on the best available language; ruby could be a gateway into Smalltalk.

Ruby has four different kinds of variables:

- Global: $name - this is very rarely used
- Local: name - scoped to method or context you are in; declared wherever used, not necessarily at beginning

```
class MyRubyClass
    local_here = "masked by def below"
  def a_method
    local_here = "whatever"
  end
```

(no compiler warning of masking)

- Instance: @name - scoped to method or context you are in; declared wherever used, not necessarily at beginning

```
class MyRubyClass
  def a_method
    @local_here = "whatever"
  end
  def b_method
    @local_here = "same var as before"
  end
```

(variables are initialised to nil if referenced before written, as in Smalltalk; nil is an object in Ruby as in Smalltalk; there are no primitive types)

- Class variable: little used
- [] is an array, {} a hash literal, i.e. association or dictionary in Smalltalk

An assignment starting with an uppercase letter is a 'constant' (in fact a variable but you will be warned when you reassign it). Generally, in Ruby as in Smalltalk, you can "cut your feet off" unlike Java which "is written for stupid people, i.e. not that Java programmers are stupid but the language assumes they are."

Q. Debate in Ruby community about whether programmers should be 'protected from themselves'? Often new programmers say they like Ruby but 'I really miss static typing; let's add that' and the Ruby 'anti-bodies' deal with this; the newbie either runs away or stays and if they stay then six months later they will be an anti-body attacking the next newbie who says 'let's add static typing'. When the community was small, it really needed these anti-bodies; now it is larger, they maybe need to calm down a little.

Class names are therefore constants and can be reassigned (but of course reassigning String to a non-class object will cause problems quickly; you can do it and then quickly do String = "asdf".class and so recover).

`AnObject.new.a_method()` is Ruby's equivalent of Smalltalk's `AnObject new aMethod` and similarly methods with parameters are `AnObject.new.a_method(a_param, b_param)`. You can collect parameters into arrays; `def a_method(a_param, b_param, *rest)` gets the remaining params into an array. Thus the same method can take an arbitrary number of parameters. You can inspect as in Smalltalk - he did rest.inspect in the method definition to show us both the array and inspecting (which popped up pure text window; no interaction).

Ruby syntax has implicit self calls: in `self.a_method()` you can omit the self. But `self.` is public so a private method called inside another will always use the implicit self as an explicit will always be treated as if it were being called not from within an instance of that object. (Use private keyword to identify such methods). super is always explicit but must always be a call of the same method as that in which it is defined i.e. you write `super`, not `super a_method` - the `a_method` is implicit.

```
def a_method(a_param, b_param = 123, ...)
  ...
end
```

means if `b_param` is missing in the call, it will be set to 123. All optional parameters must be after all compulsory. Ruby has no named params as yet (it will in 2.0). Thus you must use documentation (if it exists!) or read code. Ruby 2.0 is worse vapour-ware than perl 6 but is finally seeing some real work and is due for beta release this year and will have keyword arguments.

Methods are not objects but you can get a reference to an object that encapsulates a method and you can invoke it via call, assign it to another object, ask it for its arity. It is not as lame as Java because you can create these and manipulate them at runtime but they are not true objects.

"You talk to Java programmers and their eyes glaze over and they do not understand why this stuff is cool. You guys understand why it is not cool!"

Avi Bryant has been playing at translating Ruby to Smalltalk. Any VM that can handle continuations can translate and run Ruby code.

In Ruby, conditionals are syntax:

```
if 2 > 1 puts "Y" elseif 3 > 2 puts "N" else puts "oops"
```

They have TrueClass instance true, and FalseClass instance false, but no superclass BooleanClass. They also have syntax variants:

```
puts "yes" if 3 > 2
puts "yes" unless 3 > 2
while 3 > 2 do aBlockOfCode end
for i in (0..2) do | num | ... end
```

These are all keywords. Blocks are objects in Ruby as in Smalltalk, delimited by `do...end`. Blocks can have local arguments (same syntax as smalltalk). You can assign blocks to variables and call them or pass them as parameters. However you cannot serialise a which, combined with the fact that this is not image-based, is limiting.

```
the_block = lambda do | num | puts num * 2 end
1.upTo(10, &the_block)
the_block.call(123)
```

(The ampersand is Ruby syntax to say this parameter is a block, so you cannot pass blocks and non-blocks to the same parameter. You can pass an array argument containing a block or non-block.)

A proc is an anonymous block; `self yield` executes the one (only) anonymous block passed (actually just attached). Ampersand makes a real block object that you can assign to, etc. The 'yield' pattern was first and is quicker to type (Ruby tolerates having more than one way to do things). The result is that Ruby does not have `detect:ifNone:` and suchlike collection protocols.

After a break, we moved on to Rails. Avi showed him Seaside and he thinks it's the coolest stuff; Rails is not going to make you abandon Seaside. However it is worth knowing about because while Smalltalk has a technically better framework, Rails, unlike Seaside is very popular; it has done the better marketing. So let's look at some of the social aspects by building an ugly functional application in front of you.

```
gem install rails
```

Rails is a wrapper on top of other frameworks, activerecords and actionpack are the core items. activerecords implements Martin Fowler's pattern for object-relational mapping. actionpack implements all of the web stuff in an MVC way - actioncontroller and actionview. Running rails generates an empty application for you.

```
rails addressbook
```

A rails buzzword is ‘convention over configuration’. There are many ways of doing things in ruby but there is one published way for doing things; this saves a lot of project set up time. It generates a default structure for your app. Rails is a little less green-screen than ruby; it has a UI with checkboxes for the parts of your app. You select them to see the green screen. `server` starts up a server on localhost:3000 and you can see it - a page telling you what to do next. I typed one command and I have a web app running (less exciting to Seasiders than to others). Now we create a model

```
generate model Contact
```

lists what it creates. Rails provides code for creating tables incrementally.

```
class Contact < ActiveRecord::Base
  def self.up
    create_table :contacts do | t|
      t.column :first_name, :string
      t.column :last_name, :string
      ...
      t.column :contacted_at, :datetime
  end
end
```

The class Contact extends (i.e. subclasses) ActiveRecord. ActiveRecord assumes you will have an id field that will be automatically implemented. That is created when you generate. Rails produces the SQL from the code above and assumes you use mysql (can use another program to generate your DB if you want). Files are named with version numbers 001 and thereafter as you change your rails code and regenerate.

He opened the green screen and looked at the Contact class in ruby (vast numbers of generated methods). The column name have accessors; the = syntax provides setters.

```
myContact.first_name = “Niall”
myContact.last_name = “Ross”
```

will assign to column names, or you can use a hash (like a kind of keyword function - Rails wanted keywords so they use hashes a lot in this way)

```
myContact.create({:first_name => ”Chad”, :last_name =>
”Fowler”})

myContact.find(:all, conditions => [‘first_name’ = ?,
”Chad”})
```

Meta-programming tricks gives you a range of detailed methods like

```
chad =
myContact.find_by_first_name_and_last_name(“Chad”,
“Fowler”)
```

You can then set values on what you’ve found and commit.

```
chad.contacted_at = ...
commit
```

He recommends keeping the log running all the time when programming Rails. Like many high-level abstractions, this leaks all over the place and you need to understand the SQL being generated when things get complex.

The above is how to build the M part of MVC. Next he showed the C part.

```
generate controller Contacts

class ContactsController < ApplicationController
  def index
    @contacts = Contact.find(:all)
  render :text => @contacts.inspect
  end
end
```

He then went to the web app and showed the textual inspect output of the contacts list (only one created so far). Finally he created a view by embedding ruby tags into raw html code.

```
<html>
<body>
  <h1>Contacts</h1>
  <% @contacts.each do | contact | %>
  <li><%= @contact.first_name %> <%= @contact.last_name
%>
<% end %>
...
```

the %= embeds (a % alone does not embed). There are other pluggable view-layer programs you can use instead of the default he showed. Rails is instantly 'gettable' by JSP, PHP, etc., programmers and saves them from much timewasting activity.

You can use a script to generate scaffolding that will add some 'obvious' things'; view to show all columns, add drop-downs for dates, etc. He ran this and showed the raw form to add a new contact, to paginate contacts (e.g. 10 per page). The methods `show`, `list`, `new`, `create`, etc. are mapped to URLs .../show/id and suchlike.

```
<% @contact.send(column_name) %>
```

sends the message `column_name` to the contact object. Helpers generate field setters. Why is

```
    <% text_field 'contact' 'last_name' %>
```

not

```
    <% text_field @contact 'last_name' %>
```

He does not know but the second is a better guide to the meaning although the helper provides the first. He `raised` an error to show the huge keyword-hashes that pass objects from a form and parse it into an object.

Q(Bernard) Much code generation? Rails generation is about giving structure and creating 'opinionated software'. You can then change it but any regeneration will overwrite your changes. People use generation to learn and to get started but not to maintain their apps. (It was also a marketing tool; he could have built an address book app twice during this talk using the scaffolding and this speed looks good in demos.)

Finally he added address books to his contacts model. This created a class AddAddressBookToContacts which managed adding (and removing, if he decided later to revert his model) the foreign key identifying the address

book for a given contact. `Class AddressBook has_many :contacts` which is recognised by naming conventions as 'maps to a model Contacts' (`has_many` is a method, defined on ActiveRecord, not a keyword).

Q. Debug? He usually types `raise` into the text. There are debuggers but most ruby programmers don't use them, which correctly suggests that they are not very good; if we had a better debugger we would use it. ("Yes, Seaside kicks our ass.")

Bert Freudenberg, One Etoy per child
Alan introduced Bert as 'this guy I've never met before'. (At this Smalltalk Solutions, it is no longer the case that everyone knows everyone else; this is a very good sign. :-) Bert has been doing Smalltalk for ten years. For the last five, he's been paid to do it.

Bert showed the machine, a dinky lightweight but chunky-style (its for kids use) laptop, and passed it around. The one-etoy-per-child is an educational project. We all know the history of the Dynabook vision. Its emphasis was on 'book' not 'tool', and it followed on to LOGO, to Lisp, to cell-oriented ideas from biology and so on. (Read the "Tracing the dynabook" dissertation by John W. Maxwell.) The specs still read sensibly except for the RAM (they thought that 64k would do) and the processor speed (they would have been happy with less than 1Mhz).

The OLPC XO Hardware has to be robust (must last 5 years) and cheap (5 years of textbooks cost circa $100) and low power (must last for an 8 hour school day). Low power is achieved by making it hibernate fast (100ms) so that the machine can be powered down except for display while the user is reading a page.

It is the first implementation of the 802.11s standard for networking so each machine connects to any other in its vicinity with no configuration. It is always on so routes packages even when it is closed, routing based on battery-state of the machine.

The display is hi-res low-power sunlight-readable (goes to black and white in strong light but remains very readable) 200dpi *and* it is cheaper than a regular display. The colour is controllable on every third pixel, not on every one, which means that diagonal lines can sometimes become coloured, which is circumvented by anti-aliasing in hardware to spread out the colours over neighbouring pixels. In the usual screen the colour filter is above the reflective layer but theirs is below, which is why the backlighting shows colours but strong external light shows only black and white; the much greater strong-light readability makes this trade worth it.

They run plain Fedora Linux, X11, GTK+, Cairo for rendering, D-Bus for comms. The device drivers are written in Forth. The power-on GUI is called Sugar and is written in Python. Because all apps must run in full screen (because the screen is small) so they use the Nokia cell-phone-oriented OS as it does that. They are working on the security framework, called BitFrost, which puts each app into a separate virtual machine, a

lightweight one that does not emulate a whole PC but only separates the application spaces from each other.

They call their applications Activities because they do more than regular apps: they can be shared and they save their state automatically. The machine has limited RAM so activities must be able to be swapped out and in easily. Example activities are a journal, an Evince-based reader for PDF, DejaVu, an Abiword-based writer for Crossmark, a Firefox-based web browser, an app called TamTam for music synthesis and, of course, EToys.

Why is the UI stuff in the machine done in Python? The implementation effort is led by RedHat, not by Alan Kay; they used Python because they were used to it (maybe Alan did not fight this because he never liked Smalltalk-80; he thought it too restrictive). They wanted to take things apart and reassemble them differently. (They plan to have a view source button to let you look at it for whatever activity you are in.) If there had been EToys in Python they would use it but the Python community is not building Squeak-like things such as EToys so they need us.

EToys is a collaborative multi-lingual (English, German, Japanese, etc.) media-rich authoring environment in which authoring is always on. Designing your software for children ensures it is simple and so adults like it too. EToys was influenced by LOGO, Smalltalk, HyperCard and morphic ideas (Self's originally, which became the basis of Squeak's UI).

In morphic you are actually scripting instances in effect (implemented by subclass, change object to subclass but this shields the end-user from having to learn the class concept). A metaphor of players and costumes for them guides the scripting and you can clone to start variants. Work is done in projects. He demoed bringing up the right menu-list with position data etc. so you can drag the object and see the menulist change or change values in the menulist and see the objects move or change.

Q. In all languages? He changed to German and all the menupicks etc. changed? Then he had to change it back, which meant finding the now-German-named menupick: "OK who speaks German? Oh yes, I do." :-)

He showed scripting by the tiler and by writing Smalltalk directly (but there is no back-mapping to the tiler if you write direct Smalltalk). This is because it is a restricted environment for children with much unexposed (e.g. they have 'if' but no looping constructs).

Projects are basically desktops. A tree of objects is saved as ImageSegments (found by the GC's marker) and this is very fast. Because this uses the actual layout of objects in memory it does not let you transfer to another environment; they are also working on a higher-level save e.g. in open-document format.

Tweak is a next-version EToys. It extends EToys scripting to the system level. It replaces polling with notification; every change to an instance variable generates a change event (no more need for self change) but you

don't want these events to be handled synchronously; updating three instvars of an object needs not to block on other processes looking at the object, so these events are added to the event loop.

Sophie is a Squeak program whose UI is in Tweak. The Sophie project is run by the Institute for the Future of the Book. They are finishing the 1.0 release; you can download it for free. Plopp is a $15 (free on Linux) kids drawing program with UI in Squeak that won the ESUG award last year. It is the only program that uses a tube colour mixing paradigm.

Q. Mix Plopp with scripting? Scratch is a Squeak tool that lets you draw (poorer than Plopp) but you can also script the graphics. Scratch is made to learn programming. EToys is not made to learn programming. Generally, yes, if there was any money in kids software they would have done this already.

Tweak is not on OLPC because it needs a high end machine for these 'notify on every instvar update' change events.

They are implementing D-Bus in Squeak to eliminate a Python wrapper. They must integrate with the journal activity and the clipboard for non-file cut and pastes.

Q Fullscreen drag and drop? Frame button lets you see all apps and drag from one to another.

OLPC forced them to re-license Squeak. The original licence forced you to share any base system mods but anything above your base you could keep. This was slightly non-standard and so they re-licensed to the Apple Public Source Licence (took 2 months to get through lawyers) and then the open-source people started muttering don't really like APSL and so they re-licensed the original Squeak 1.1 again to Apache 2.0 which can live with everything. All subsequent contributions are now being re-licensed to the MIT (free-est of all; just says 'don't sue me') so everything will be Apache 2.0 or better. ("Squeakers who have not yet signed the letter, please do.")

They want better ability to send messages direct to the hardware (c.f. Michael's talk). See vpri.org.

Q. Classroom experience? They have 10 years of experience. One problem is that EToys is a blank sheet of paper. They are working to provide example projects to get started.

Q.Will OLPC take more Squeak projects? Each country can choose what software to install on the machine. It is very easy to download stuff from the web with a one-click. Go to laptop.org and become a developer (only way to get your hands on a machine right now).

Q(Yann) Use SqSquare for collaboration? Time will tell which is best.

Q(Bruce) How to keep up with OLPC code evolution? Download from

wiki on laptop.org; it's findable. To see what version, you must open a shell and (finished answer offline).

Q. Audio chatting? Yes, and camera and microphone.

Q. How is it sold, marketed? They plan to ship millions to developing world governments (by year-end it is hoped; the date has slipped from this summer) and that is how they keep costs low (no distribution cost) so selling in the west is an unsolved issue.

GLASS: Gemstone Linux Apache Seaside Smalltalk, James Foster, Dale Henrix, Gemstone

Ruby on Rails is giving buzz to the Ruby community. Seaside is providing similar buzz to the Smalltalk side. However Seaside, although it does what it does much better, is not as complete a story on the database side. The Smalltalk Enterprise Application Server Integrated Development Environment provides the layered abstractions over HTML and HTTP that lets you quickly build and modify highly interactive web applications. Because these applications use the web, they do not need you to build a UI and so what has otherwise been a limitation of Gemstone Smalltalk is less important.

Avi created a Ruby framework and then ported it to Smalltalk because the IDE was so much better and the class libraries so much more mature. However he almost went back to Ruby as it had built in continuations - until he realised that in Smalltalk you can add them in 10 lines of code.

Building complex web apps is hard because HTTP is designed to serve static pages and so is stateless. Other frameworks tend to have to marshal all state and the back button is a problem. In Seaside, each served page is associated with a continuation of the stack, which leads to a far more natural coding style. James then reviewed the WACounter code and behaviour (see the many prior Seaside talks written up in my reports an d elsewhere) stressing the various renderers, brushes, etc.

Seaside originated in Squeak and has been ported to VW, Dolphin and now Gemstone. Seaside needs continuations. Squeak and VW just added a little code. Dolphin released a new VM that supported them. VASmalltalk roadmap is to add continuations. Now Gemstone supports them.

Continuation is a class with methods `class continuationDo:`, `valueWithArguments:`, etc.

All other dialects are single-user, non-persistent, so the user must add multi-user and persistent behaviour when their website needs it. You can persist in the image, which is very easy but you risk loss of data if the image quits and it is not shared across multiple images. Another choice is a BOSS-out but object identity is lost on file-in. Putting it in an external database involves coding and impedance mismatch. Multi-user can be done with several users per VM (scaling and query-directing code), or multiple images (needs coordination through file system or whatever).

Gemstone provides multi-user persistence by default.

Seaside requires a web server. Smalltalk has Swazoo, Hyper (a Swazoo branch), Kom (formerly Comanche), etc. External servers require FastCGI; you can use Apache, Lightppd, etc. Smalltalk servers are great for development. You can deploy on them or on external, which may amortise an existing server on the host or whatever.

James started a Hyper server and opened the SuchiStore demo application, showing the HTTPServer instance handling the connections.

The latest official version of Seaside is always in Squeak. VW and Dolphin maintain ports. Porting needs you to export file-out from Squeak in suitable format and then import to your dialect. They preferred Monticello, which is Squeak's distributed concurrent versioning system. Monticello is repository-independent so can use Gemstone; they made the very important decision to port Monticello, not just Seaside so any seaside app versioned into Monticello would automatically port.

Method overrides are an issue in a multi-user system; you cannot let just anyone load a method that changes a kernel class for everyone. Session methods add a sessionMethods dictionary decorating the methodDict, so that method lookup looks in each class' sessionMethods dictionary, then in each class' methodDict, then looks in the super's sessionMethods then in the super's methodDict and so on.

Q. How to remove? Override with DNU call or (better) raw super call.

Q. Like Method Wrappers? Yes, we think. (Niall: in session methods, two dictionaries are consulted in turn and the session method is not expected to call the masked implementation. A method wrapper is a replacement value inserted into a single method dictionary with a pointer to the compiled method that it has replaced, which it need not but frequently does call.)

Thus you load Squeak directly from Monticello, making some mods to the code. (They were modifying _ to := but this makes it hard to see what interesting changes were made so they have allowed underscore. Someone in the audience noted that Squeak is now dropping underscore.)

SqueakSource is two things: a web domain and a Seaside application (the web domain happens to run the Seaside app).

Most Seaside apps use a Smalltalk server but production ones often want to run another, especially if the programmers and the IT department are separate. Let Apache manage static pages, SSL, load-balancing, fail-over backup, etc. He stopped and restarted an apache server, briefly showing where the things you configure lived.

FastCGI communicates between a web server and an application server: mod_fastcgi.conf showed localhost:<port> for his demo but in general this is of course very distributed. He then demoed using topaz scripts (which he

noted are “Rubyish”) to test the basic functions, launched from a web page.

Dale built a framework (Tsunami) to do denial of service attacks against Seaside servers and studied when the CPU saturated. Squeak saturated at 30 pages per second. VisualWorks saturated at 60 pages per second (with Swazoo). Gemstone saturated at 60 if it committed every page and at 200 if it committed every 200 pages. (Disclaimer: for Squeak and VW the continuations are kept in memory whereas Gemstone dumped them to memory, so the VM GC was not spending lots of time on it. They did not tweak the VW memory servers, which Jim knows can make a huge difference. Gemstone was writing 6Gig per hour as the alternative to GC.)

In Gemstone, each VM gets shared access to the data so scaling is easy and close to linear by adding VMs. (They have customers who run 1500 VMs on 200 hosts.)

James opened the SuchiStore example and showed two browsers to two different ports. He showed the two seeing the same data which he changed and showed both saw it. If you commit on every hit then the continuation objects are saved and so any VM can continue. If you want to assign affinities to distinct VMs, Apache can do that. He assumes Apache-capable people using this will assign the appropriate patterns for doing this; they have not explored that yet.

His Squeak image has access through the foreign function interface to a library that can login to Gemstone. He thus gets a transcript in which he can execute and browse Gemstone code. This is a very watered down version of what GBS provides. Squeak does not have a rich toolset yet but may grow some, however it will never have transparent replication (not for free at least :-).

If you do not have a 64-bit machine (or don’t have time to install Gemstone) but want to play with it, apply for an account on http://seaside.gemstone.com/ (where you can also find the slides). Use the mailing list at http://www.seaside.cst/Community/MailingList/ sending subscribe-gemstone-smalltalk at earth dot lyris dot net. Send emails to James dot Foster or Dale dot Heinrichs at GemStone.com.

(Later discussion with Jim: the main way to tweak Seaside memory in VW is to increase Eden and SurvivorSpace by 10x - 25x because you do not want the many objects that Seaside creates surviving into oldspace; they die young so let the scavenger collect them. Another is to optimise the settings for memory growth; make them aggressive, not procrastinating. Lastly, make sure your external server serves anything static.)

Application Frameworks and Experience Reports

Seaside Experience Support, Boris Popof, DeepCove Labs

DeepCove is a small technology group in a larger company (120-130 in Vancouver and Ireland). They like XP and use lots of test-driven development, no code ownership, frequent small releases, on-site-customer (they share the floor with their customer) and 40 hour weeks.

Their product is called Raven. It processes international payments in 150+ currencies in more than 65 countries in many types. They do millions of transactions a month, so they are significant but not huge. They are an all-Smalltalk shop running on 7.4.1 (7.5 tested) talking to a Microsoft SQL server using a homegrown minimal OR mapping. Clients can batch requests or submit them one at a time in REST style.

Raven Online is a web portal for clients. Clients need to initiate payments and to obtain detailed reporting. The project started soon after Avi's and Andrew's hands-on Seaside session at Smalltalk Solutions 2006. Boris had not seen Seaside before. On the flight back he built a simple prototype in 3 hours (he showed the simple login and upload transaction page he created). To do that on a plane with no access to tutorial or whatever impressed him.

Their clients were then sending them WebDAV files. This was awkward for many clients and Windows built-in support for it is flaky; the uploaded files sometimes don't upload so you see what is not there, etc. (and it is even worse in Vista). Generally, the push model for getting these files was a problem (client might accidentally delete file then think they had not downloaded it); they wanted to shift the obligation to get files to the client.

There was also a fear factor delaying web projects as there were only 4 of them in the group and servlets and SSP seemed harder than they should be.

The next stage was a mock-up which took 8 days. They regarded this as about learning Seaside; there are few and old tutorials.

Q. Changes rapidly; what impact on your code? Could be significant if you want to move to the latest version from the old API; otherwise, the changes are compatible. It is important to talk to the Seaside community and not just use the old tutorials which will point you at deprecated code.

Visual design matters: a professional look motivates you. So make it look good early. Also, layout affects functionality which affects layout and so on. By pure luck, he saw a book on CSS and saw an author photo showing a Vancouver background so contacted him; it turned out to be David Shay, author of ZenGarden. Do not let developers do a web designer's job; let the one who knows how to do web art do the web art. Boris asked Avi how to interact with the web designer. His rules were: express what you want to get done, not how, showing existing brand materials and ideas if any. Hopefully the designer's initial patterns will be reasonable and then you need little interaction, just iterations of him sending you the latest CSS and you loading it and seeing what you think. Now, 12 month later, he showed various (very professional-looking) CSS stylings of the application.

Q. Javascript used; if so, whose responsibility? Yes we used it for one feature and it was Boris' responsibility. Some web designers might offer help on that, others would not.

They needed two-factor authentication. RSA or Verisign are the two main players; they used RSA because Verisign would not even talk to them (they

were too small). They found RSA very flexible, willing to negotiate based on projected volumes. Their devices start at ~$80 including six years of service fee (after which it costs a few dollars per year).

Seaside is all about components. A root component has children which have children and it really was that simple. HTML generation is not magic; you must get your hands dirty and you have to know what you are doing. However it is all messages, not templates, so you reuse and refactor instead of endless copy and paste. The new render API is good. You get your brush (`html div`), cascade settings onto it and finally call `with:` which closes the brush or starts brushing with it.

Callbacks are blocks and are for actions, input fields or anything that returns a value. The `value:` and `callback:` methods are key, the callback being what happens when the `value:` is changed.

```
html form:
  defaultAction: [self process];
  with:
    [html label
        toolTip: 'Minor units with decimals e.g. 19.95';
        with: 'Amount'.
      html textInput
        class: 'required';
        value: self amount;
        callback: [:value | self amount: value].
      html submitButton
        yui;
        callback: [self process];
        text: 'Process'].
```

They wrote `yui`, a call that lets you use yahoo styling on your buttons. They will open-source it if they have the time. Ajax effects let them flash green for approved, red for rejected and so on. scipt.aculo.us is used by ruby on rails, Apple, Digg and Gucci because it has been much objectified by Lucas.

He found that late binding is good. He ripped the system apart and put it together again several times so the fact that you will not get it right first time is no cause for worry. Seaside makes refactoring and reuse easy: smaller components give you less coupling.

Initially they deployed a headful image with all resources talking to Apache and SSL. Then they moved all the resources (CSS, Javascript and images) to Apache. Next they made the Seaside headless and cloned the whole setup beneath a load balancer (Microsoft NLB - somewhat flaky but it works). They also cloned the SQL DB to mirror it.

Then they put all their resources into Amazon S3 for 24 cents/month, while the load balancer talked just to two Apaches in front of two headless Seasides. Amazon only have a north America centre for S3 today but they are in north America so it was not a problem for them. This was easier than mapping resources from windows-based seaside servers to linux-based web servers when they deployed.

They created an automated build process that they can run once all tests pass. They upgrade in the middle of the night when there are no sessions since sessions running when they upgrade disappear. Exceptions terminate a session but leave the image running (they have their own ways of handling these).

They insist on IE7 or Firefox because they can (in fact Opera also works OK last he looked). Boris demoed. What the client's contract supports is what you see - anything else simply does not appear in the page. The `yui` makes their popup responses look nice. It also had nice dynamic-look charting capabilities to show things their clients should monitor. They use fusioncharts (like DabbleDB) for these effects. He showed the Smalltalk that created the chart. There are many searching and reporting options; he showed Ajax flashing effects. Raw table resports were what they started with but they soon realised they wanted grouping and substructures.

Q. HTML and CSS; how much did the designer change from your mock-up? All of it; designers know what they want. Boris was not a designer so had not set ids to suit. He sent files each evening and he had them integrated the next morning.Code reuse meant they only needed to change things in one place.

Tons of credit goes to Michel Bany who keeps Seaside ports up-to-date to within a day or so.

Interactive Visualization in Widgetry, David Buck, Simberon
A client asked him to do components in what was called Pollock and is now called Widgetry. His project was called Eagle as it was about seeing things from high-level or in detail.

His first task was to build a histogram, showing arbitrary number of bars and of elements in each bar. Thus it shows the objects that are being counted into a bar. He used it to show all the classes in his image, categorised by initial letter of name. Moving the mouse over a bar showed which class was that point of the bar, with separate horizontal and vertical scrolling. He showed paging and dragging the screen; a home button lets you return when you get lost. He has implemented his own clipping of the histogram to the screen (to give good performance). The leftmost vertical edge is set by the label length.

The announcement mechanism is much nicer to use than the old event mechanism. Events were symbols and you passed parameters. Announcements pass the pane, the group and the item index, and announcements can have their own useful behaviours.

Q. announcements are synchronous? Yes, exactly as for events.

Lines between elements are only shown when you zoom in far enough; at high-level, the elements are not sensibly distinguishable or selectable and so they disappear.

The next component was the hyperbolic graph viewer. A large graph displayed on a two-dimensional sheet soon runs out of space: as low-level nodes have many children, the high-level nodes must space out more and more to accommodate them. You solve this by using non-Euclidean space. Hyperbolic space has more space as you move out from the centre:

- close nodes appear large
- distant nodes appear small

He displays it on a three-dimensional sphere (other projections are also possible). A 2D hyperbola is the set of points where $x^2 - y^2 = 1$. He projects down to the y = 1 line. In 3D it becomes a saddle $x^2 + y^2 - z^2 = 1$ (rotation of 2D). The y = 1 line becomes the z = 1 disc and projection works the same way. The 4D case $x^2 + y^2 + z^2 - w^2 = 1$. The z = 1 disc becomes the w = 1 sphere and that is what they project. “If you’re good at imagining 4D spacetime, this is a no-brainer.” (It is just the negative of the Einstein spacetime metric, with ‘w’ the time dimension. I did a lot of work with these metrics and geometries when I was younger and had great fun discussing them with David offline. He regretted he had not known of my experience with this before as he would have liked to talk it over with me when he was learning this geometry.)

The widget calculates the spanning tree for each node, then does layout: think of the central node as a circle surrounded by child circles surrounded by grandchild circles and so on. The math was challenging as VW does not have sinh, cosh, tanh (but they were easy to do via e^x) and all David’s geometrical intuitions were wrong in hyperbolical space. Translating a triangle required two reflections about an intermediate point. Distance calculations are different.

His first implementation was rather sluggish (30 secs per frame) because he was drawing all of the nodes, no matter how far away they were. He improved by drawing the centre node and its neighbours, truncating when things are too small to see. He had to animate fly-over events (otherwise you lost track of where you were) and assign translation (to left-drag) and rotation (to right-drag). Labels had to be greatly restricted to centre and immediate vicinity only.

(Q. There was a suggestion about using clifford algebras for geometric computing. I suggested two-spinor decompositions. Discussion continued offline. I omit highly technical details of interest only to geometers.)

The next component was a world map, with arbitrary pan and zoom, landmark labels, etc. He demoed, zooming in to the point where country names start appearing. (“Implementing this has given me a whole new respect for google maps. It is hard to do all this well.”) As one zooms, there is sometimes a brief delay as it loads extra detail that has just become visible. Latitude-longitude lines become more detailed as you zoom.

Data source: He started with data from the 1980s (country boundaries have changed a lot since then) and then found ESRI (company name and data format name). It is a binary format, very well-documented so it was easy

for him to write a reader for it.

He used the ‘descriptor method’ pattern (google that phrase for his blog article on it). The method defines the file format, saving you from duplicating the knowledge of the format in a reader and also in a writer. The format implements description methods the reader and writer can use. The header tells the reader what to expect and the writer what to write, e.g.

```
header
  self
    bigEndianInteger: #fileCode value: 9994;
    bigEndianInteger: #unused value: 0;
    ...
    littleEndianInteger: #shapeType;
    ...
    mark: #endHeader“tells us to switch to body”
```

Colouring the country: we all know the four-colour theorem but the U.S., like some other countries, does not consist of a single contiguous area; fortunately, his client did not insist on four colours. To see which countries are adjacent, he looked at bounding polygons, not at every point on the boundary which would have been far slower.

Detail versus speed was always an issue. Displaying all the detail in the high-level view would mean drawing many points to the same pixel, and would be very slow, so scaling what information a given view displays was key. He divided the world into 12 x 6 regions and clipped polygons to them via the Sutherland-Hodgeman but this leaves degenerate edges (a polygon can become two subregions when clipped, connected by a degenerate edge along the clipping region’s side). He solved this by breaking the polygon into clipped and the rest every time he crossed the clipping boundary (if the rest the produced a wholly external polygon, he just threw it away). He then summarised the data by ‘average point’. He is still not fully satisfied with this solution (and it has increased his respect for google maps).

Landmarks: did you know that the centre of the United States is in northern Africa? For a multi-part polygon, you put the label in the centre of the biggest part.

Q. How long to do? 150 hours of work (200 hours all told included a couple of other minor widgets and features).

Q. 100% VisualWorks? Yes.

Q. Why did the client want it all done in VW? He does not know. It may be they desire to modify these widgets or interact with them in fine-grain detail. He believes they will use them as a framework.

Q. Other projections? They requested mercator projection, which he did, and others can be done; you plug in a ‘camera’ object to do the projection.

Q. Using widgetry? It was easy to build a UI programmatically, which is very difficult to do in wrapper. He found that pushing the left and right

arrows on the menubar left to make room for his close button required him to override a method. He also wanted to subclass classes which had platform-specific subclasses already so he could not. (C.f. discussion of this in my write-up of Sames presentation at ESUG 2002.)

Q. These widgets are now available to us? Up to Cincom and the client.

Cairographics and Smalltalk, Travis Griggs, Cincom
His slides were (impressively) rendered with Cairo; he programmed the whole thing and every time he hit page-forward the next slide was rendered; except for some buffering, there were no shortcuts. Travis wrote his first graphical program in Fortran IV. He has been doing graphics in Smalltalk since 1992. He joined Cincom last September. Before then he had worked on ExtraEmphases and wanted anti-aliased fonts. His goal is to make cool graphics in Smalltalk fun again.

Cairo is a two-dimensional graphics library that supports multiple output devices (XLib, Beos, Win32, Quartz, etc.) and platforms (Arm processors, various cell phones, Windows and MacOS X, etc.). It has many language bindings including a Squeak one (called Rome, which also includes other stuff) used in Sophie and Firefox. He has had much help from the Cairo community and from Michael, Holger, Joachim, John Sarkela, Sean and Andreas. Get it from the home page (http://www.cairographics.org) plus the libraries.

He started with ExternalInterface LibCairo, parsing the functions in standard DLLCC style (from cairo.h version 1.2, hand-maintained since) making the ENUM types into classes. This is where most language bindings stop (e.g. the ruby bindings, which he looked at). He went on to build an object model. He tried to use OsHandle but had issues so made his own, using weak links for finalisation. They kept the method names the same (despite some being unintuitive if not plain wrong). Methods hide the `memberAt:` calls. Opaque structures are UninterpretedBytes subclasses. For some reason, fractions are not automatically mapped as integers, doubles, etc., so the message `dllccDouble` deals with that.

The first thing you do in Cairo is create a surface, in formats a1 a8 rgb24 rgb32. You can do this wholly in Cairo (it manages the bytes):

```
ImageSurface format: CaitoFormat argb32 extent 100@100.
```

or by creating a ByteArray in fixed space and telling Cairo to use that. Pixmaps surfaces are longlasting but you should recall the cairoSurface (to synchronise - pick up bounds and etc.) every time.

```
aWindowOrPixmap cairoSurface.
```

When you have a surface, you then create a context (often abbreviated as cr or aCR), obtained by `aSurface context` (like a GraphicsContext but it has more state, its shape. (Yesterday, Michael and he made it possible to map any GraphicsContext to the equivalent cairo context.)

The source of a context is a pattern, (`aCR source`) which is a solid colour,

a gradient (linear or radial, and it can reflect at boundaries), etc. Travis has created VW helper methods like normal colour setting. Travis then showed some self-rendering code; the beautiful pattern made the code text hard to read but hey it's so cool :-). (Gradients are as fonts were for apple users in 1982. :-) A second self-render (of the Smalltalk balloon) was hard to read even in large font but showed what you could do with not too much code.

Paths are built via connect the dots: moves, lines, curves (all bezier quadratics - the wikipedia article has a great animation and explains well) and closes. Paths can be disjoint and are infinitely thin, not 1 pixel wide. Paths must be closed to be filled and must be done explicitly: a path not declared to be closed can return to its start point but it will be unfillable. Paths are pen-down moves: `lineTo:` or `relativeLineTo:`.

Enumerating over paths is tricky as the various elements are so unlike: moves, lines, curves and closes. They wrote a case-like method as the easiest way to express an enumeration. He then showed a cincom logo self-rendering slide (which was very readable). If he had done this in VW not using Cairo, it would have taken more code and the rounded edges would be less smooth (no anti-aliasing).

Cairo has some 'verbs' which you can apply to a stroke of a path, cleared at the end of the stroke unless you explicitly preserve: paint, paintAppha, clip, fill. He showed a VW and Cairo slide, showing how Cairo's infinitely thin lines avoid fill overlap in thick boundaries of filled shapes. He showed various clever things on the Smalltalk balloon leaving the island background: subtractive clipping, masking.

Cairo is a different way of drawing in which you can apply affine matrix transforms for everything ("use the matrix"). This is good for doing e.g. translate to centre, rotate, translate back in a single operation.

Contexts can be saved to a stack and popped again to let you unwind things. You can group a temporary surface and then pop it; this is handy for double buffering (can also use it for animation; no significant speed gain).

Cairo has a 'toy' text API; he wrote a minimal Pango interface for this slide; raw Cairo was too tedious.

This is not necessarily faster but it is a lot more attractive (Joachim's train simulations look much better but run at the same speed).

Smalltalk in Semiconductor Test, Mark Petersen, IBM
If you have cellphone or an xbox, you probably have an IBM chip. They are an IBM systems and technology group, but are using VW, not VA, due to an accident of history. They use Smalltalk to characterise semiconductor behaviour, not just to test them. (IBM has a site in Bangalore and they are now pushing Smalltalk into the group there.)

Semiconductors are made on 300mm wafers which are then diced. Wafers are fabricated, and then tested to understand the behaviour and the process,

perhaps as part of a production process, perhaps before the product is built. DMACS is a Smalltalk/DB2 application that tracks this testing.

Data mining is an issue. For a wafer barcode field, he would like to know what products it was used in, whether it had failures in the field, etc., which at present is data IBM has - but all in separate silos. If Smalltalk can give that service, it would greatly motivate its further use.

He showed some views that represent the device behaviour. They have extended VW's business graphics to let them mark and edit graphs and to add 'smith charts' (an unusual electrical characterisation chart which they make out of two business charts superimposed). Images (.png) are saved to the database and pushed into HTML documents.

They have not used Store in the past, just writing some code to their DB2 database. Now the team has grown larger than a couple of developers, they will use Store.

They have a Smalltalk web server that presents objects as XML on a web service. They also provide data via ftp and email. They added some exponential formats (1.0E4, 1.0e+4 would neither be correct in base 7.4).

General Purpose Interface Bus is another communication protocol. They have built a Smalltalk utility to talk to the gpib32.dll (from National Instruments) and the NI gpib card (from Intel).

Why Smalltalk? A programming language should be chosen for how well it solves your problem. They are engineers, not programmers, and find that Java (both the language and its overhead) makes it too hard to map their domain into the code.

Q. (Will Loew-Blosser) Why are engineers programming (his company has merchants doing contracts who are bright but they do not program)? One is raw cost, another is that they can only ask IBM to program for them and they cost (their web services cost them $100,000) and take time (the web services used waterfall and took a year to gather requirements and he could have done it in a month). Some managers are impressed that he can give them things in a day or two, whereas they are frustrated by how slow the standard Java route is but IBM has a corporate strategy.

He showed their language comparison matrix in which they evaluated Smalltalk against Java, Matlab, C++, LabView along 16 axes grouped as essential, very important and important. Smalltalk scored best.

Q. LDAP interface? The LDAP parcel has problems so they used a perl script.

Q. Training? They have two courses of one day each to get people aware of the application. People struggle with Smalltalk, and with OO generally.

He ended by mentioning things he would like: scripting (maybe S# like),

FileManager with read/write features (they have done that and Alan mentioned that latest VW has it, native for windows and a portable one which you can also set to have for windows).

Home Automation, Thomas Stalzer, Object Dynamics
[I was presenting in the parallel thread. David Buck wrote up this talk on his blog and these are his notes.] Thomas said that this was the first project that he did for himself and just for fun. He may commercialize it later but no definite plans have been made yet. He started in Smalltalk in 1990 and worked for IBM and later Enfin. When he left Enfin, he started Object Dynamics working in Banking, Insurance, Production and other sectors.

His interest in home automation started when he remodelled his house. He decided to add automation to help make life a bit better. For example, when he opens the dishwasher, the light above the dishwasher turns on. There are many things he can do from lighting to controlling appliances that make things more convenient around the house.

His application was developed in VASmalltalk. It controls appliances (stove, dishwasher, etc.), power and lights, network based equipment (www, e-mail, weather, radio, stocks, etc.), custom systems (garage door, alarm, A/C) and multimedia (MP3 servers, AMX, Bose, Russound, uPnP).

These systems each have their own unique interfaces and there was no integration. His system gets these devices to work seamlessly with each other. He does this by abstracting the physical layer from the application layer. Each device has a driver that controls that device but to upper layers presents itself as attributes, events and actions that can be wired to other components in a similar way to the Composition Editor of VASmalltalk. For example, the ‘Door Open’ event of the dishwasher can be connected to the ‘Lights on’ event of the light above it. The name of a song playing on the MP3 player can be connected to a display that shows that name.

Features he has programmed with these event-action or attribute-attribute connections include:

- opening the dishwasher door turns on the light above the dishwasher
- dishwasher can be turned off with light switches
- if the house is locked, all appliances, music, etc., are turned off
- the lights in the living room blink when the dishwasher or washing machine are finished
- if a motion detector is triggered, the lights will be turned on (50% brightness at night)
- if the house is in ‘security mode’ then ringing the doorbell or a motion detector can cause the sound of a barking dog to be played from an MP3 player inside the house
- a sensor on the floor of the bedroom detects when you step out of bed and turns on LED lights along the floor to help you get around without blinding you

- when an e-mail is received, a light in the room flashes

The system is programmed with a graphical interface that lets you draw connections between units shown as boxes on the screen. A 3D simulator with a model of the house lets you try out the programming without connecting it to the actual house controls yet. Once connected, the simulator can show and control the devices.

Q. Does the system ever gets messed up causing you to lose all control over the house? Functions are divided into core and luxury functions. The core functions will work whether or not the system is operational. You can always turn on lights from the switches, turn on and off the appliances manually and control the A/V equipment without automation control. The luxury features require the system to be operational. If you push the 'theatre' button, it closes the curtains, dims the lights and turns on the A/V equipment. This can all be done manually if the system is down.

Q. Does this affect your buying decisions for appliances? Most high-end appliances already come with interfaces for automation. In some cases, an appliance he would have liked to buy did not support automation so he bought another instead.

Q. Can other household members use the system? Using the system is easy. He is normally the one who does the visual programming to connect everything up.

Q. Any crashes? Only three he can remember. One was a hard disk crash which required a hardware replacement. The other two were caused by software problems.

Q. What kind of interfaces do these appliances have? RS-232 is a popular interface for appliances.

This was a fun and interesting talk.

Coding and Testing Patterns

There is no Spoon: Overcoming the Object-Relational Mismatch in OLTP Systems, Thomas Gagne, Instream Financial

Thomas is the CTO of Instream Financial. Instream Financial buys service obligations and turns them into financial instruments.

What does the title mean. In the film 'The Matrix', someone bends a spoon and explains how he does it by saying, "There is no spoon." He feels that as regards OO-relational mapping there is no spoon. An industry exists to handle OO-rel but in his systems he avoids the problem. This is a database heresies talk (like Avi's 'web heresies' talk; he gave the example of having to use his PDMA as a light recently; if he were the kind of person that thought Java's philosophy sensible, he would not have thought of it).

The database is the most important object in his system. It is the first thing to get back on line if everything stops. The next thing is the correct entry of data; all else is cosmetic. (A comp.lang.smalltalk discussion with Nick

Malik at Microsoft suggested the term ‘Semantic Persistent Store’ for a database that you put your app in; Thomas preferred ‘hypostasis’. An X.500 system is a black box with an interface i.e. it is an object.)

Alan Kay believed that having messages be objects was the main thing about Smalltalk. Similarly, it is tempting to turn RDB tables, columns, etc., into objects *but* the meaning of identity and suchlike are not the same. OO systems are proved empirically; RDBs are proved mathematically. In RDB everything is 2-dimensional whereas OO is N-dimensional. 10 DBAs would create 9 identical designs from the same requirements which is not the case for 10 OO people.

Thomas is not here to advocate OODBs or OR mappers or wrappers. Think of the database as its own independent object apart from Smalltalk. Their inStream application was constructed by analysing the business and then designing the database. They did web pages in python at first (why - irrational fear) and then rewrote them in Smalltalk.

He will spend lots of time thinking what is the least amount of typing he can do to achieve a task. He tries to hire people for whom this is even more true. In his system, database people are not separate from OO people.

The language should not control what you do. The business exists before the technology and the database provides memory and longevity. Why use stored procedures: because if people cannot grasp using stored procedures then will they grasp the application server? Smalltalk understands an object’s data being private; stored procedure access to the database’ data makes its data private.

You do not need an OO language to think OO. C code in OpenLDAP, CAL and idbLib/ctLib is very OO in style; you call to something via a handle you are given.

An Semantic Persistent Store (SPS) is like a Smalltalk object because

- it is a subclass of a more general database (c.f. subclass of model)
- it can dehydrate and rehydrate itself: save to disk, etc.
- it is a state object
- it is a singleton: the SPS is the single instance of the system’s current state
- it is reflective; you can ask it for its columns tables etc.
- it lives in a virtual machine
- it is cohesive
- it responds to messages

Lastly it reacts poorly to tight coupling: if you see objects with nothing but getters and setters to let other objects act on its data, that is poor code. So we should not access our database via direct SQL in our code.

Differences: everything is in fact public; if i know the name of a table, I can probably access it and you can interrogate the system to get table names.

Java did not exist before ‘95. The databases it works with mostly with existed earlier.

“Why stored procedures; I can create a view?” Views are select only, tightly coupled and too simple. Triggers lack parameters and can only use data in participating tables. Their system has only one trigger (for an essential unfigurable correlated change that must happen).

He then demoed, showing the SQL test script his system generates (see slides). Login creates a session key (he could do that in Smalltalk but that guy in accounts who hacks in to check something; will they use ST?). The transaction history is the most important thing in their system (and in most financial systems). Thus all changes can be associated with history by adding transaction history setting in the stored procedures; all callers see the same API as before, just as in a Smalltalk object. Validation changes are similarly changeable without changing the API.

They can use the transactions to reconstruct the past state of any object. By adding a ‘start and end day balance’ they made such reconstructions in reports run in seconds instead of in many minutes.

Their system has 613 stored procedures of which 76% are queries.

Many people think the web, or the mainframe-to-PC-change, caught everyone off guard, forcing much coding to move business rules from one location to another; not for them. The next disruptive technology will prompt more such work, but not for them. Your business rules have to be somewhere that will not change. In summary, treat your database as an object; respect it as an object; you will be more flexible.

Q(Bruce) Are you saying that in MVC the M should be the database? (The answer was a discussion that I missed; I think Thomas broadly agreed.)

Q(Davd Buck) Smalltalk handles scaling complexity? 20% of their system is in Smalltalk but that is the most important part because the database knows what happens but it does not know why. The Smalltalk knows why. So it is critical. Thus the Smalltalk has a model but it is a transient model. Whatever persists long term goes to the database. Their critical Smalltalk apps each have a different model, one for what to buy, one for how to settle.

Q(Bruce) hard to change DB vendors? We never have; we trust we never shall. We bought Oracle for a reason, to exploit its specific features not to avoid them. DBs like languages are not created equal. Follow-up Q(Alan) some domains sell to others who will run on their DBs? Thomas agreed.

Q(Andres) OO prevents small changes propagating everywhere? In a transaction-processing system they find they can change how things are represented to create new products. Example: Delphi went bankrupt in

2005 (declared over a weekend as usual), causing a worried management meeting on Monday morning (they were $24 million exposed). They had to change their system to pay 60% against invoices immediately, the rest when Delphi paid; within 1 hour they had designed this change and were in QA within 3 hours and were in production that evening.

Q. Using this with an OODB, e.g. Gemstone? That could be a problem for them because they would need to write more code for all the non-Smalltalk accesses. In future they will make their API an XML API.

Interfacing to C, Michael Lucas-Smith, Cincom
This is the thing he hates the most about VisualWorks. He has done far too much of interfacing to C (he will also talk about C++). He will explain where the rest of the world is, where we are and where we should go.

Why bother with C? Well a great many C libraries exist and it would be crazy to rewrite them all in Smalltalk; we should reuse what exists.

The state of the art?

- intravenous C: recompiling the VM with the library included or as a dll
- fast foreign interface: telling the VM enough to interface natively, the C artefacts may be modelled
- Inverted Interface: recompiling the library to support a specific API which provides the meta-data you need to interface to it
- Pushing the inverted paradigm: Python only does this by automating generating the stubs for this

VW has a C header file parser that uses K&R so usually doesn't work. So you must define the interfaces yourself (which is how almost everyone else does it as well). VW has nice error handling to get the callback error raised in the thread that called.

VisualAge uses pragmas to declare C to pool dictionaries.

Squeak, Dolphin and VW are similar (Squeak lets you write faster generating C code and swapping to the new VM). GNUSmalltalk has a very limited interface description (only a few basic types but this allows most things) and you must modify the VM yourself. Understandably not too much C interfacing has been done in GNU.

Smalltalk/X dynamically compiles C code into itself so you can write inline C code in your Smalltalk method.

That's how we do it. How do they do it. Python without boost is the most primitive interface: call to get back chunk of memory. With boost, it is much more powerful and in fact is the lead solution. Perl is like Python without boost. Ruby is second worst, only above Java. It is like very invasive GNUSmalltalk; you must use the inverted pattern but the Ruby guys endure it and have done a fair amount of integration.

Q(Jim) do they fall behind an evolving library? Michael thought no, they just check their code back into the project. (Non-open-source is a problem, especially if they cannot get source. They can make a wrapper library.)

C# does less than you would expect. It does not parse the header file and you must write for yourself the 'use this dll and these are the entry points'. There are some commercial programs that offer parsing of header files.

Java is by far the hardest. You must modify your original DLL, make an interface class and another class that uses it. He is amazed that so much has been integrated; comment on how many Java programmers there are.

Lisp is like VW and Vassili told him he had written a header-file parser in a previous job but he could not find it, or any other. Scheme is like Lisp but with different commands because they hate each other.

Forth, like GNU, has a limited description but they do have the first actually-working header-file-parser. They always compile the library into their VM.

COM and .Net? VisualWorks COM interfaces are very automatic, because COM was designed to provide a lot of interface info by default. .Net in VW subclasses DotNetObject which then transforms C-style calls to .Net calls.

Examples and Morals: don't reinvent the wheel. LibXSLT is very good so who needs a VW XSL library, especially as it was an old spec, since much changed, which can parse little XML code you will find today. A lot of open-source programs are cross-platform so you do not need to reimplement them in Smalltalk just for cross-platform. VW uses subclasses to hold the platform name (so must remember to throw them away when saving in case the image is moved to another platform) or ExternalInterface pragmas (better as you can now extend the superclass for a new platform and dispense with subclasses).

Counter example: LibTidy cleaned up HTML to XHTML. It was not available on Mac initially and when they finally got it working it was bad; it only really worked well for Win32 and Linux. They would have been better writing a Smalltalk version. Another LibTidy moral is don't use C interfaces. A python program called cwm transformed RDF to more RDF. They had to make ExternalProcessStreams and use VW's only gradually getting better stdin/stdout story.

Don't use C moral: LibASpell. You can make progress run much faster in Smalltalk than in C if you get the algorithm right. LibASpell was oriented to post-hoc checking and built an automata for each word, whereas the ImitationLevenshteinAutomata (only implemented once before, by its inventor) only pretends to make an automata to compare two strings and it was *much* faster. It could have been implemented in C but was easier to do in Smalltalk.

Use C moral: if a program works well on all platforms, use it. BerkelyDB

is a raw table implementation, offering various formats, btree, etc. and is used in gmail and other places. After using this, they could inspect a 4Gig byte array in Trippy and look anywhere in it with no delay. While doing this, they found all kinds of bugs. FixedSpace fragmentation assessment was reporting fragmented space as free space and tried three times before failing to use it; they also fixed a GC malloc bug. When Oracle bought BerkelyDB they changed every magic number in the header file and Michael took several days to upgrade for it.

FastCMethodPointers: pulling a pointer out of a struct and calling it was losing VM optimisations so they provided these to run as fast as optimised.

StrongCompositePointers: in a single-threaded call the VM stops when you call out to C. In a multi-threaded model, call arguments are copied to fixed space so they will not be garbage collected. However a non-argument will not be so protected (see slide example) so StrongCompositePointers will keep those referenced objects too.

Garbage killed it moral: every slow C interface he every created was always slow due to garbage. LibSDL does two-dimensional rendering and the interface creates lots of garbage. A streaming web server taking thousands of requests per second will also create garbage; for ZLib they created an flyweight pattern for the entropy pool creation which limited garbage and accidentally gave better compression because ZLib is smart enough to reuse the entropy pool. BerkelyDB was very fast at creating what VW was very slow at cleaning up, so much so that they left it single threaded; the process environment critical lock (prevents two processes modifying it at once) was just too slow.

Lessons learned: if you could magically identify that you should create stuff on the stack then you could solve this problem. Modelling C stuff as classes makes programming easier but can create more garbage, Structures that deallocate themselves in C when dropped in Smalltalk (Ephemerons) kill the GC because the VM doesn't know that this ephemeron is holding onto 10Mb and needs to be prioritised (this is the same for Python boost and everyone else).

Michael did what he needed: everyone else does this too. Michael listed some of the many many C interfaces that interface to *parts* of the Windows interface. If the VW C header file parser worked, this could be solved but no human will ever do all for almost any library; life is just too short.

C is a partially context bound grammar. C++ is a completely context bound grammar. GLR parsers alone can parse it and they are very hard to write. C and C++ has application binary interface problems. VW and VA have to make assumptions when it calls to C and you must compile the right assumptions for the platform. He described graphviz which has a switch to force alignment (to be fast on platforms that do memory alignment, its suppliers say). And in C++ you have all these issues and all the better-known C++ issues. Finally, the suppliers have learned their lesson; the latest Intel 64bit C++ compiler has the same ABI everywhere.

Simplified Wrapper and Interface Generator (SWIG) builds language-specific interfaced versions of C libraries.

Python.Boost is a better SWIG that runs only for Python and provides two way calling - the C++ can call back into Python as well.

GCC-XML outputs an XML description of GCCs internal representation so it is a way of dodging writing a GLR parser. Python has a new project Pyste (pronounce as you judge best :-) uses this with Python.Boost to build the fast foreign interfaces between Python and C++ automatically.

c++filt claims (and, from his quick examination, correctly claims) to find the real method name from the symbol table of a C/C++ table.

He closed with his wishlist. Forth can do ANSI-compliant C parsing so we can. Stack allocation would solve all our GC issues. All common and OS libraries pre-parsed and delivered with VW would be a 'Smalltalk is better' point. Automatic generation of classes from structures would replace code he has written over and over again; the more time we save creating the interface, the more time we have to optimise it.

The current state is a barrier to Smalltalk adoption.

Q(Thomas) making the things you bring in Smalltalky (e.g. making things into streams) is another issue; are there patterns? You could sometimes recognise that a thing was a stream but not always.

Q(Bruce) how soon to get any of your wishes? The C stuff could be fixed (e.g. by Pete Hatch) in a hop, skip and a jump. By contrast, the magic interface to C++ could fall apart due to ABI issues.

Q(Thomas) easier debugging, e.g. of memory allocation; it can be hard to see how the VM is setting up the arguments just before it makes the call. Michael just uses the C debugger. Yann Monclair mentioned wrapping a call to see what was passed and what was returned. Thomas stressed it was the middle layer of the VM he found hard to see. Michael pointed out that the marshalling stuff in the VM could be in Smalltalk. Michael wishes there were no graphics interface code in the VM (recent mouse example).

eXtreme UI Testing, Niall Ross, eXtremeMetaProgrammers
I started by offering a picture of a spiky Stegosaurus as a (somewhat questionable :-) symbol of agility. Stegosaurus' walnut-sized brain and huge spinal ganglia caused the dinosaur-hunter who first found it to think it had two brains, prompting a wit to claim that

> It could reason 'A priori',
> but also 'A posteriori'.
> If something slipped its forward mind,
> 'twas rescued by the one behind.
> And if in error it was caught,
> it had a saving afterthought.

We're all XP programmers. Most of us have been caught in error and saved by running the tests. (And most of us, looking at some of the errors thus revealed, have wondered whether our brains are the size of a walnut. :-) In my keynote talk in 2005, I described how the conventional programming philosophy ("Coding from a spec is like walking on water; it's easier when it's frozen.") was a doctrine for those who think they are clever (enough to be right first time), whereas the eXtreme Programming philosophy ("First make it run, then make it right, last make it fast.") was a doctrine for those who know they are not (likely to be right without trial and error). Test-driven coding means that, like Stegosaurus, we can reason 'A priori', not, as in the conventional case by thinking (wrongly), then coding (badly), but in the XP way of expressing our thoughts as tests, then coding (better). We can also reason 'A posteriori', not as in the conventional case by using the system, then redoing it (dangerously, if at all), but in the XP way of using the system, then running its tests while redoing it (safely and as needed).

So what about XP and UI Tests? Can UI tests play an 'A priori' and/or an 'A posteriori' role?

My first XP lecture was from Kent Beck in 1999. Kent's dictum was simple: ignore UI testing. "One of the great things about the web is that it's trained our users not to be so picky. They're now accustomed to seeing UIs that look like garbage and change without notice." (I think Seaside may be undermining this. :-) Some coding tools let you develop in a style like test-first coding. The Refactoring Browser framework's rewrite tool is an example: the best way to develop a metacode pattern is through step-by-step change of example code, matching on every change. SmaCC is another: iteratively parsing further and further into examples is one way to develop a grammar. So I tried to interpret Kent's approach as meaning: use GUI frameworks in this way i.e. after test-driven coding of the model layer, add the UI via a GUI builder, without tests.

My first XP project had no UI tests - and had problems because of it. The monthly delivery cycle was often fraught because we tested amongst ourselves but demoed to others and while the model-layer always worked, making sure that all the UI did was always a post-hoc mini-regression-test.

Abandoning the 'rigour' of Kent's approach, can we add UI tests via tools? Silvermark's Test Mentor offers click and type test generation for UI's built from standard VW or VA widgets (it can need adaption to application-specific UI frameworks). It locates widgets by name if they are named, or by generated name if possible, or by the path from the enclosing subcanvas or window when duplicate names occur.

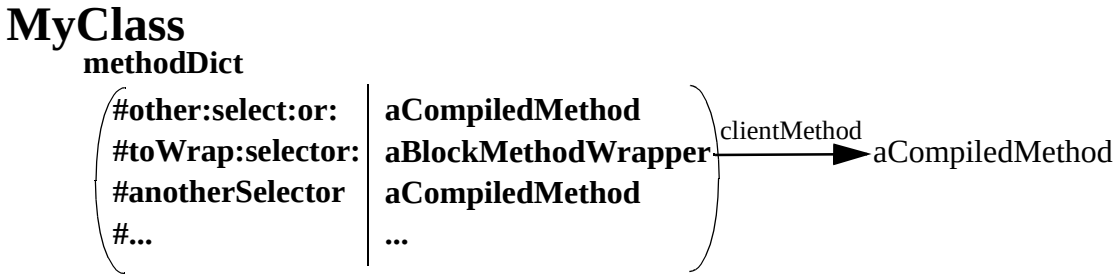
In David Buck's VWUnit, ApplicationModels are located by test-supplied, test-specific name. Widgets are located by relative location (to others and/or within bounds) and/or name/id. (See his talk at Smalltalk Solutions 2003. Also c.f. some similarities of style with widget location and manipulation in David Shaeffer's Seaside Testing Framework; see David's talks at Smalltalk Solutions 2005 and at ESUG 2005.)

James Foster has written extensions to WinRunner. WinRunner offers click and type test generation for a fixed (inaccurate) view of a Smalltalk application's UI. Its widget positions are fragile and its widget names are not unique. James' add-ons help you tweak your application to fit WinRunner's expectations better (see his talk at Smalltalk Solutions 2004).

My experience of using these tools is very limited (limited commercial experience of TestRunner, even more limited hobby/open-source use of VWUnit, none of James Foster's work) so the following remarks should be read with caution. I find them good for retrofitting UI tests to stable systems, and for automating acceptance-style tests, etc. They are also useful as examples of how to write widget-driving code. But somehow they lack the XP test-driven feel. In 'A posteriori' use, the tests tend to be brittle under refactoring. This brittleness effectively discourages 'A priori' use: you tend to build a stable UI *then* retro-fit its tests (and it's hard to click a not-yet-existing widget to generate a test :-).

With the above review done, I got down to the meat of my talk. The talk could have been titled 'Using Method Wrappers in Tests, with UI-oriented examples' as method wrappers are key to all its patterns. I first reviewed wrappers as not everyone is accustomed to using them. A method wrapper is the decorator pattern, applied to the values in Behaviour's 'methodDict'.

MethodWrapper classInstVar 'methods' holds a dictionary of integer-keyed templates. Each template is generated by the first request to wrap a method of that number of args. Instantiating a wrapper copies the (appropriate arg-number) template, rewriting its literal placeholder to self. (If the args are non-zero, the template also has another placeholder literal which is rewritten to Array.) If the dialect allows CompiledMethod subclasses to have named instvars (VW, Squeak) then MethodWrapper subclasses CompiledMethod. If it does not (VASmalltalk) then (following John Brant) my preferred implementation is to make MethodWrapp*ing* subclass CompiledMethod while MethodWrapp*er* rewrites the literal to itself in MethodWrapp*ing*. (So in VASmalltalk, the diagram below would show aMethodWrapping in the methodDict. Its compiled code literals would contain aBlockMethodWrapper, invoked when it was called.) This approach maximises shared code across dialects.



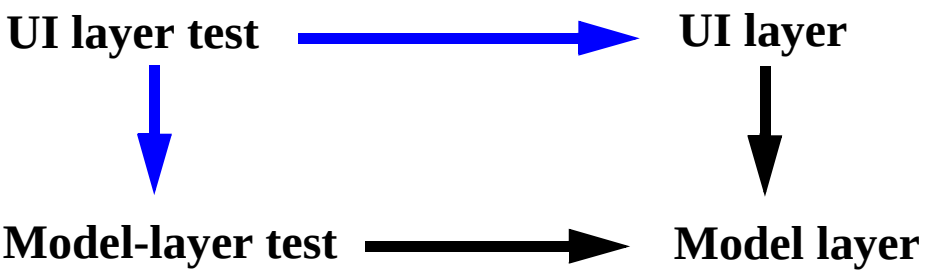
Wrappers modify method execution: they may run code before and/or after the wrapped method, or instead of it (perhaps only if a condition is met), or constrain its execution in some way, etc.

My earliest UI test ideas were to get at widgets by making UI operations

return things or by finding them in system caches or allInstances pseudo-caches. This soon reached its limits - unreachable popUps, unstable delays, complex UIs, etc. - and I gradually switched to using method wrappers.

Use wrappers to suppress side-effects, e.g. `ignoreHelpAround:`, to get and set values e.g. `setup:inClass:toReturn:around:` and (c.f. Mock Objects pattern) to prevent UI-launched tests going too deep, e.g. `executeRefactoringOfSameClassAs:invokedDuring:`, and to handle popUps and manipulate UI objects generally. Method wrappers naturally separate test and normal modes of use. In Smalltalk, wrappers are how to achieve what in other languages needs 'policy' code.

My next pattern was to derive UI tests from model-layer tests. The UI-layer and high-level acts on the model layer in a way very similarly to the way in which top-level model-layer tests (the most powerful and useful tests) act on the model layer. In normal use, the model layer gets its values from the UI and returns its results to the UI. Under test, the model layer gets its values from the test and returns its results to the test. We can therefore complete the commutative diagram ...



... by creating UI-layer tests as subclasses or delegates of top-level model-layer tests, e.g. SomeFunctionTest subclass: #SomeFunctionUITest.

- Inherit model-layer tests as UI tests.
- Override value setters/getters to set/get values in/from widgets.
- Override operation invocations to press buttons, select menu picks, etc.

This lets you add UI tests with little (and reusable) extra code. The tests use the *same* logic as the model layer ones, so only UI specifics need extra understanding.

I walked through a simple example, CascadeTest and its subclass CascadeUITest; CascadeUITest subclasses CascadeTest, inheriting its tests, almost all without overriding. Utility method overrides change test set up / execution to UI test set up / execution. Instead of presenting a subset of source code to the refactoring, they select it in a window. Instead of executing a refactoring, they find and invoke an item in a menu. A wrapper is key to grabbing the results just before they compile (you do not want to effect them but you do want to test invoking from the UI with no UI-related defer option set). Another simple example is SplitCascadeTest and SplitCascadeUITest (see the Custom Refactoring project's CS11 RC3 release for VW 7.5 and earlier, and VASmalltalk 7.5.1 and earlier).

Such UI-test-first code can help ensure the right order of adding new UI elements. For example:

- In VW7, all RBs become unusable if the UI refers to missing actions, forcing you to fix the RBCommand entry in the debugger (or in Trippy, whose class browser remains usable)
- In VA and older VW, the RB hides menu items that it thinks are not implemented. This was an optimisation of John Brant's, adapting common RB UI to multiple platforms some of which did not have all its model features implemented, but it can catch the XP test-first purist, who adds the code to the UI first and then cannot find it.

I showed example code of how tests protect me from adding such changes in the wrong order and so falling over the same trivial and tedious error again and again. They capture and enforce my safe UI coding procedures.

My final pattern was about how to test a UI that runs in a different process from its model or in multiple processes; the former is a common situation and the latter quite common enough. In ordinary SUnit, tests that fail outside the test process run green but raise walkbacks. This is bearable when developing tests, less so when running long test suites: who wants to come back to their computer after lunch, or next morning, and find thousands of green tests and a debugger notifier popped up by - well, I wonder which test did that? It's another demotivator for writing UI tests.

My cross-process test utility makes such tests safely runnable in suites. I walked through its code, which is straightforward. CrossProcessTestResult overrides and refactors `runCase: aTestCase` so that the method it calls `runCase: aTestCase forBlock: aBlock` can also be wrapped round any block spawned by the test. The top-level call installs a cross-process test strategy before running and waits (on a ProcrastinatingSemaphore, which can have negative excessSignals value) till the sub threads complete.

The chosen strategy wraps all test-launched other-process blocks in a `runCase:forBlock:` call by applying a method wrapper to the key spawning method: `Process class>>forBlock:priority:`. The CrossProcessTestWrapper passes the first-argument block to its strategy.

```
valueWithReceiver: aClass arguments: args
  ^clientMethod
    valueWithReceiver: aProcessClass
    arguments: (testStrategy wrapBlock: args)
```

The CrossProcessTestStrategy instance applies the test handler.

```
wrapBlock: anArray
  self isUnderTest ifTrue:
    [| aBlock | aBlock := anArray first.
    anArray at: 1 put:
      [[testResult runCase: testCase forBlock: aBlock.
      subthreadCompletionSemaphore signal]
        on: Process terminateSignal
        do: [:ex | subthreadCompletionSemaphore signal.
             ex pass]].
    subthreadCompletionSemaphore unsignal].
  ^anArray
```

The strategy controls wrapping and signalling of (normal and forced) completion, and how to recognise a test-spawned process in the dialect. Where the dialect supports stack searching (e.g. VW, VA), we check whether the stack calls `runCase:forBlock:`. (Where a dialect does not, the strategy must tweak a spawned process' value - e.g. its name - or capture it to a list. This has been coded and works but is clunkier.) The strategy also sets the timeout delay of the ProcrastinatingSemaphore (a timeout is essential during development and prudent during production; if threads spawned by your test never end, neither will the test unless you have a timeout :-).

Specific test cases can override the default TestCase-extending strategy for various reasons:

- The typical UI test opens a window, invokes operations on it and closes it. In VW, a test closing a window goes through `unscheduleWindow` which does *not* terminate the WindowManager process; the next UI cycle does that when it sees that the manager has no windows. So if your test was launched from a test UI (the usual case) then the process waits for the test UI to update before terminating and the test waits for the manager's process to complete before updating its UI - until the test timeout resolves this deadlock. The `wrapProcessesForUI` strategy wraps `unscheduleWindow` with an afterBlock to force the check:

  ```
  wm hasWindows ifFalse: ... wm checkForTerminate ..
  ```

- Blocks can be created in a test and added to an event queue running in a process not spawned by the test; in VW, block-holding Actions are used in UI code. They are queued by `DeferrableAction>>block:`. This can be wrapped to ensure that DeferrableActions created by a test are also run wrapped in its handler if the relevant strategy is used. (A trivial refactoring override of `DeferrableAction>>message:` to call `block:` is included in the VW implementation to catch all cases.)
- Special strategies can help debug complex multi-threaded applications once problems are found. These develop naturally during debug work and capture debug knowledge far better than the `Transcript show:` code that often disfigures tricky debugs of multi-process systems and is lost after every debug, to be hacked back in at the next one.

In summary, UIs can be tested in fully XP fashion, i.e. 'A priori' and also 'A posteriori'. Method wrappers enable such UI tests. The same tests can drive model-layer coding and UI coding. Tests running across multi-process UI and model can be added to refactoring suites. Simple examples are in the Cincom Open Repository in the SUnitUtilities bundle or the RefactoringBrowserTests bundle. (N.B. *not* the old 'Refactoring Browser Tests' bundle. Use with appropriate 'Tools-Refactoring Browser' version.)

Acknowledgements: John Brant and Don Roberts provided the RB. John provided Method Wrappers. John's tests are, or were prior versions of, or inspired, several basic wrapper examples. I pair-programmed with Katerina Barone-Adesi and Adriaan van Os on tests using the subclass pattern. Michael Lucas-Smith pointed out user interface issues for the

cross-process pattern and Reinout Heeck pair-programmed with me to analyse them and to write the ProcrastinatingSemaphore.

Q(Giorgio) The subclass-model-test-to-UI-test approach will tend to add all values to the UI widgets in a fixed order for all cases. His UIs may have 40 widgets and the model state updates as a value is added to each one. Different orders of adding may give different results and this needs to be tested. Boris suggested that the UI test's utility method that overrides the model test's high-level setter could be further overridden to rerun the test for all possible orders of adding. I agreed; I think a subclass of the UI test, (use a subclass to distinguish the run-all-orders case from the run-single-expected-order case) could override an appropriate method to effect running every test in every order.

Q(Peter) You can makes MethodWrapp*er* work as a subclass of CompiledMethod in VA by making its clientMethod an indexed instvar, dropping the MethodWrapp*ing* (of course, specific wrapper subclass instvars must also be indexed). At first glance, I prefer John Brant's approach: it keeps MethodWrapp*er* subclasses' code identical across dialects, and indexing errors could lead to horrendous debugs. Peter and I will exchange versions and review.

I invited discussion whether agile programmers should test UIs at all, and if so, how? There was much discussion. Most had survived so far while doing very little but, like me, several had experienced problems and delays as a result. Some plan to trial the 'test-driving via UI test subclass' approach. Yann Monclair, like me, has looked at using TestMentor in Kapital and noted the need for adaption to the Kapital UI framework. He mentioned its lack of cross-window drag-drop support.

Application Frameworks: an Experience Report, Arden Thomas, Cincom

(I only caught the first part of this talk.) Arden has been in Smalltalk since the 80s. At one point, he worked for a Hedge fund in Connecticut and in fixing things he needed an application framework. He was not authorised to build one but he was authorised to buy one.

Frameworks should make things easier, simpler and clearer. It should also not make it difficult to go beyond the box in which the framework makes things easier, simpler and clearer. In the early '90s, Tim Howard wrote a book that explained DomainInterface. Steve Abel built the ValueInterface (inspired by the slamdunk architecture). These framework both had a single domain and hid things. ValueModel put additional instance variables in the builder so the ApplicationModel was not cluttered by them.

A ValueModel is a model from which you can get information via a simple interface (value, value:) with dependency notification. The value, value: calls can be to blocks that recompute, to buffers, to many things besides mere data holders, and can be to other value holders and so on ("You can solve any computing problem by adding a layer of indirection" :-).

ValueInterface extends the ValueModel idea to ApplicationModel. This makes it easy to connect widgets to the domain and to react to domain changes. It also makes it easier to reuse applications. (nameChanged will be run when you change aspect name - issue when refactoring?)

The void object is like nil except that it disregards messages it cannot understand. (Niall: this is the message-eating pattern of which there is much debate whether it is good or bad.)

BoFs and the Coding Contest

Smalltalk Industry Council BoF, Georg Heeg, David Buck

STIC's task is to create and increase awareness of Smalltalk, to promote it as the superior technology for the construction of IS, to attract skilled human resource to it and (less important to Georg) to support creating industry standards.

Q.(Bruce) STIC should get VW, VA and gemstone to standardise simple things like files, sockets and so on.

Board: Gemstone (Monty Williams), Cincom (Jim, Suzanne, Alan and Joy) and Instantiations (Eric Clayberg, John O'Keefe and Mark Johnson). Bob Nemec has been executive director; Georg now takes over.

There are user groups (ESUG, CSUG, Frankfurt User Group, etc.) which organise the community. STIC's task is to care about strangers and to promote awareness in the industry. STIC organises Smalltalk Solutions and provides web sites.

"Smalltalk is greater than this R stuff": the ruby presenter.

Bob's aim was to move to where STIC could advocate Smalltalk from a solid base. A year ago the finances and the websites were in a poor way. (Treasurer's report: took a year to get sorted.)

- Mark Roberts worked on the Smalltalk central site.
- The Smalltalk Solutions site was to be a place for conference archiving.
- Bob Cherniak has a site where you can renew membership.

Chris Cunnington (a one-year Squeak newbie) set up a booth and is setting up a Seaside hosting site.

Q(Niall) include registration with conference? Requested last year, just doing registration for this year was so challenging that it was not done this year. They will try to do it for next year.

A contribution to Smalltalk productivity awareness is the talk available from Instantiations site. An existing Smalltalk project was asked by their management to convert their system for a paltry sum; Instantiations showed metrics that showed the sum was an order of magnitude too low, using statistics from real projects.

Lines per function point: a spreadsheet may have 6 loc/pt, Smalltalk has 21/pt, C++/Java has 53. A web based Smalltalk app went from 50 to 600 fps over some 8 months. he also looked at fp to effort over time.

Standard code metrics show ratios of elapse, effort and size (loc). Real data for actual projects produces a model for Smalltalk and then the equivalent in Java and other languages. In Smalltalk, it takes 50 - 100 months of effort to build a 3000 fp system. Similar size systems in Java take 200 - 300 months. The talk also shows Smalltalk, Java and Cobol productivity rates. Smalltalk has gone as high as 100 function points per month.

Smalltalk systems cost between $185 and $536 per function point. Java and C++ cost more than $1000 per function point. The data was displayed in tables. Smalltalk programmers are not specially expensive; there are plenty of .Net and Oracle programmers commanding $200/hour. The cost rations were serious.

Schedule overruns cause half of project cancellations. A short planned schedule in a language whose productivity does not allow it leads to poor quality which leads to a long actual schedule. Smalltalk has 0.14 defects per function point, whereas Java has 0.50 per function point. (Generally, statically typed languages are markedly worse than dynamically typed languages.) Figures for typical defect introduction and solution rates imply that converting a project will cost the same or more as building the original cost of building the Smalltalk system.

Q(James) over the last year there have been fewer conversion jobs? He replied that they are now offshored so might not be seen. Suzanne mentioned that she and James have helped to place 24 Smalltalkers with customers and she is definitely seeing less conversion. Vicky has also seen fewer conversions.

Q. Figures on Ruby? “Ruby is a gateway drug to Smalltalk?” is what the Ruby guy said. Unlike Java, enterprise customers will not deploy in Ruby and that means Smalltalk has a chance. However this is because the Ruby technology is still very immature, which will not last that long.

Bob noted that in 1997 he saw companies switching to Java because they perceived it as less risky. He hoped these numbers would prompt wiser decisions today.

When David Buck sees companies perceiving Smalltalk as risky they say they cannot find Smalltalkers and they cannot find training and if they found training they could not persuade their staff to be keen about it because they do not see Smalltalk as a good thing on their CV. There was discussion of how STIC could help make training available, and known to be available. David did a video last year on developing in Smalltalk.

Q. Just now the word in the Java community is that Java is the modern Cobol. People are looking for the next thing.

Jim mentioned that his blog started with 12 views per day but he now gets 15000 views per day. This is done by providing something every day. An article every six months is no longer very influencing. Example applications on the web are useful. BottomFeeder gets 200++ downloads every day. The wikinomics talk was about how the right context becomes more than what any one person can do.

Email stic_board@heeg.de with your offers and suggestions. www.stic.st is where the new entry point site. Old sites (e.g. whysmalltalk and other stic.org etc.), are proving hard to persuade to redirect. (.st is the domain name of an island. I suggested it was the island from which the Smalltalk balloon took off. In fact it is the island of San Tome. Suzanne volunteered to spend ~~a holiday~~ time there to establish our right to use it if necessary . :-)

Q. Still use source forge? Yes, source forge is an OK location for Smalltalk projects.

Monty has lots of customers who have found Smalltalk 'sticky' to get rid of; one CEO finally announced, after 200 years of effort trying to rewrite in Java, "If I ever hear about this conversion again I will fire everyone."

Georg won a free STIC membership for a year long ago and heard nothing from STIC that year. Should he email once a month? No, email when the website submission process changes, when STIC has a list we might want to add to, etc.

Q. Move conference? It is under discussion (and email Suzanne with suggestions if you know one that would be a good fit).

GemStone Bof
GemStone 64 release 2 has just shipped. It supports Seaside i.e. continuations (in GemStone 64 only, not in GemStone 6.1.5).

The GBS 7.1.1 release improves performance in object allocation (when only single objects, not big trees, were involved, it reduced allocation 90% and up speed by 30%) and large integer hashing (they gained several orders of magnitude improvement by implementing their own LargeInteger hash). Object identities are all ByteArrays not LargeIntegers (no math is done on them so no need for them to be integers; they are read from C buffers so it is quicker to read them as ByteArrays than to compute LargeIntegers).

The next GBS release will have single-round-trip: forwarder-send has one trip for execution, one to get the result and one to get the dirty objects; it will become a single trip, a big win if network latency is poor. `evaluate` will also benefit and the lazy versus immediate faulting decision is removed as there will be no difference. Concurrent traversal buffering (server can fill buffer as client is processing it) will speed up processing.

They will move all this to their VA version; they appreciate they are a little behind there just now but they mean to support their VA customers fully and they will catch up.

They are also looking at forwarding messages on the server, not understood messages being forwarded etc., in ways that are faster than the uncached forwarders being used for this today.

Q (David Buck) Going to server during C callback from OS in VA: get error if waiting on a semaphore so cannot go to Gemstone server? Yes, it's a tough problem; they can only workaround today but they would love to have better solution.

James Foster: Gemstone 64 version 2.2 has continuations (and SUnit tests for them), letting you run Seaside. Seaside 2.6 has been ported into the Gemstone server (see James' presentation on Wednesday at 11:00). Some character-encoding that HTML needed was slow in Smalltalk so they added primitives to do that. They also added an underscore operator so that Squeak source (old source - Squeak has now discarded it) can be easily ported. You can open a browser from Squeak on Gemstone (it's a basic tool at the moment; they hope the community will develop it). Monticello has been ported so you can use Gemstone as your Monticello back-end.

For a while, they will offer a hosted sandbox of the 64bit product at seaside.gemstone.com; ask them for an account. This is for 32 bit customers who do not want to go to 64 yet but want to explore it.

Kernel modifications can be tricky so they offer session methods: they only show up in your session, not in other people's sessions.

Q. Squeak or VW as preferred environment for Gemstone + Seaside? James would see Gemstone as being preferred :-). If you have VW or VA already, you will want to stick with those mature tools. Otherwise, choose as you wish. Of course, Seaside offers a web browser; you could use that.

Monty Williams: for years people have found Gemstone too expensive. Seaside under GLASS offers a Gemstone/S web version with 64 million objects, 1GB RAM and 4GB disc on Linux with squeak tools (for web clients only) for free. (If you want support, that would cost, of course.) You can go to the next stage of $7000 per year for 256 million objects, 2GB RAM, 64 GB disk with all clients and 20 hours support/year. When that is too small, you become a standard customer running on Solaris, HPUX or AIX and as large as the product allows. You can start by logging in to seaside.gemstone.com.

Q. Documentation? Seaside docs are on the web. You build your app in Monticello then login to seaside.gemstone.com and suck it out of Monticello. Gemstone docs come with product. (See the Wednesday talk.) Using Gemstone in general is complex because there are many things you may want to do. Using Gemstone in a seaside app is a much more specific case and you can set it up quickly.

Q. This is exciting as everyone else has to come up with replication mechanisms to SQL or whatever whereas this maps directly and naturally and is incredibly scalable.

Q. 2 connections for free, 10 for $7000/year means...? It is how many VMs you have connected to the DB, i.e. how many Gems (not counting garbage Gem, symbol Gem, ...). Thus you can have many web browsers being served by one Gem while another serves developers upgrading things. You can run at 1-5 requests per second and manage in the free regime. They trade RAM for disk to handle the 500+ objects that a Seaside hit creates. (In defining the proposal, they avoided Gemstone-specific terms for wide comprehensibility. Alan pointed out that 'database connections' sounds too like 'connected users' so recommended they call it 'server processes'.)

Q. If I own a Gemstone/S can I use the free to distribute? Yes but as there are no VW or VA clients in the free; it will not connect to GBS.

Q. Web client means ...? Gemstone has sockets, Anything that can listen on a socket, i.e. web browser, RSS aggregator, etc. They support swazoo, fast cgi.

Q. Modifications to Seaside? They subclassed WASession to modify only the internals of managing continuations so porting to future versions of Seaside should be trivial.

Q. Motivation for this? James has been pushing this for two years, along with Monty and others. They persuaded management that Seaside provides a way to get new people in easily and it is a perfect match with Seaside since transparent persistence is exactly what you need to take off. Bruce Badger's OpenSkills work was the start-point and an inspiration.

Q. Chances of a windows version of Gemstone 64 v2.2? Bring your cheque book to that negotiation. Or run in virtual machines; VMware will run Gemstone on windows.

They are new to Seaside; help from the Seaside community on seaside.gemstone.com is welcome. Please try it out before the official Q3 release. Existing customers who have a 2.2 licence can download it today. GBS for VW is available today; otherwise it's topaz today (so just like Ruby :-).

Seaside BoF
Seaside is the beach on the Smalltalk Island.

Projects: load balancer? The ones that exist work fine with Seaside. What is needed is a manager: you stop taking requests and wait till the sessions die and then do this maintenance and then go back in service.

Seaside is easy to learn but Carl has found scriptaculous and even Seaside Asynch harder to grasp. The Seaside asynch stuff has a better API and fit but does less: if you write the component carefully, it is easy to then turn it into an AJAX component. Scriptaculous is now stable but awkward; the awkwardness could be fixed.

What other Javascript libraries should we use? Tons but there is little you

cannot already do with scriptaculous and asynch so if someone chooses to make others available to Seaside that is their call.

Asynch is the first thing to look at because it is easier to understand. Use scriptaculous for what you need it for. Mixing these and with yui is no problem; Boris has found it no problem provided the libraries all load unconflicting. Seaside asynch uses the same render method as the standard Seaside so you can easily just change the calls. Seaside is stateful so you (remember to store the state and) have no problems.

Carl Grundel presented Run BASIC. Liberty BASIC was built in Smalltalk in 1992. Run BASIC is Liberty BASIC for the web. It is 'Seaside without Seaside', a Seaside app that lets you write web apps without knowing anything about Seaside. Its market is hobbyists, small businesses, students. The site takes you through simple examples programmed in Basic; you see the page and inspect the code that creates it as you step through it. Examples include a number guesser, a sine curve drawer, a hangman game, a calculator and a version of Tiny BASIC written in Run BASIC. Other pages let you write and run your own code, email a friend about it, save and load your projects and publish them to be served on the web.

They don't use continuations so could make it more performant if they could switch components off. They could use unregistered SeasideAsynch to suppress them.

They began adding subroutines to set CSS in odd moments over the past two weeks.

Q. How does the Basic create HTML? Everything maps to an object model; the program inserts their renders into the correct places in the component. The background (i.e. the Basic) program runs in its own process and when it finishes it waits on a semaphore until the next event. The foreground process is signalled whenever the background waits and renders the page.

After this demo, we discussed Seaside commercial use. One person is working with Amazon, one on an academic archive, one on document management for a company, and one (Boris Popov; see his talk) on a financial application. They are seeing attention from outside the Smalltalk community.

Q. Do clients complain you are not using something standard? No, the customer is our own company and they just want it to work.

Q. How to tell teams that use DreamWeaver to use CSS? If the developers are very unhappy with CSS then that is a problem but CSS is now mature and we should use it. Yes, sometimes no more templates scares them

Q(Niall) Present this at web designers talks? Avi's web heresies talk is presented at such conferences. Georg suggested going to the conferences that Ruby on Rails visits.

Boris showed his application again. Marketing: mention that David Shea, Mr ZenGarden, liked to work with you and Seaside. Boris intends to put the demo on the web to let the marketing department show customers how good they are but other people will be able to go to it to show how good Seaside is.

WATask does not render; it handles flow of the application. It is a crude version of web transactions.

Q(Bruce) Release S3 code for others to use? Yes, I can publish it to Store.

Q(Bruce) Can you set cache expiry in headers? Seaside's default URLs are not really meaningful. You can modify the URL in your component so it gets a meaningful URL.

Q(Bernard) Content management system for static pages? Can do but it would be overkill.

Q. Have a canvas that draws in morphic? The renderer does make assumptions about the HTML that is generated. If they are semantically aligned then that is OK; otherwise there might be issues. The question is more what HTML and AJAX can do: windows with overlaps and drag and drop is the kind of thing a desktop will have that web libraries would be needed for.

Coding Contest, STIC
This year's contest was run by Andres Valloud, last year's winner.

The first round was to write a program to play a simple memory game. Let an even number of cards, marked with symbols (Andres used the various icons in the VW UI) be face down on a table. The player can turn over pairs and if the two cards have the same symbol they can be removed from the game. Unfortunately, the player has a limited memory and can only recall 10 previous cards they have inspected. Andres implemented this situation as a computer with 10 slots, registers for the turned-over pair and one register to move things around. We had to write a program to drive his (via HTTP protocol) to a complete solution (all cards matched and removed) with minimal value for some combination of cycles and instructions (we had to deduce the exact formula he was using). This made quite a good first round as it was a task that was easy to do but hard to do well.

The three finalists were myself, Michael Lucas-Smith, and Leandro Caniglia and Valeria Murgia acting as a single entry pair-programming team (they do all their coding together so Andres allowed them to compete together; this gave them an advantage but, as you will read below, the result of the final was influenced by other impacts more significant than this :-).

Andres gave us a new program, driven by the same interface, and we had to improve our submissions to win against the changed - but he was not telling us how - program. We started, and started encountering oddities. At first, Andres assured us that surprising effects were to be expected but

gradually it became clear that we were all encountering a tendency for his program to run for a while and then suddenly decide that it didn't know us and it also became clear that this was a surprise to Andres. The final halted while he tried to debug this. Michael, recently arrived from Australia, caught up on sleep, I tried to help Andres debug, and Leandro and Valeria sensibly tried to understand what the intended surprise was from the scant data they could get before their runs hit the unintended surprise.

As usual, the problem was blindingly obvious after it was eventually found. Andres had run a final 'check it all works' test in his image before saving it and the workspace he ran it from was still open and VW's 'automatic workspace variable declaration' had caught him. (I *always* make myself write the temps explicitly in workspace scripts for precisely this reason.) His final test's server was held in the workspace's variable and so not garbage collected, so his image was offering two servers on the same localhost port. As we ran our code against his image, our program started talking to one server and then at a random point in the run suddenly was routed to the other, which was not in the correct state for it.

By the time this was found and new images distributed, the final was more than half over. We restarted and tried to work out what the intended change was. Leandro and Valeria got it right: the player's limited memory had now become unreliable, so at random intervals the slots no longer contained what they seemed to do (i.e. the player could no longer recall where the card was). They built a version that could handle this and complete in a reasonable time. Michael and I both started by guessing wrongly that finding cards had just become harder and that if the player kept seeking a slot they would find it. Michael submitted a program based on this which hung forever in the decider run. I realised shortly before the bell that this could not be correct, so junked it, instead coding something that was not so much 'the simplest thing that could possibly work' as 'the stupidest thing that could possibly work': it remembered nothing but just turned over cards randomly till at last it completed. It did not hang - but took so long to finish that it made little difference :-) - and as my code was still running my over-thorough 'prove it does not hang' test when the bell went, technically I did not submit at all. Thus Leandro and Valeria won, and I drew the tertiary prize: run next year's contest. :-)

Overall the final was very enjoyable for the contestants, who smiled when the contest manager, rather than the contestants, became the one who had to solve a problem to a tight deadline, but I fear it was less fun for Andres; I feel for him, having prepared such an impressively professional contest program to be caught out by such a trivial but hard-to-find bug introduced at the very last minute by his own 'let's just check it one last time' test. :-/

Contest Assessment: it is good to have a challenge that is easy to solve but not easy to solve well, nor with a single obvious solution strategy, and Andres' task was certainly that. I'm less sure about what one might call the Nebuchadnezzer approach to task definition: find the definition of the problem as well as the solution. True, the real world often offers problems where the definition of what precisely you have to optimise, or what a

changed program to which you must interface is now doing, is inadequate or wrong or non-existent. Equally, in the real world, things are rarely pure black boxes: it was not clear to me whether we were expected to decompile the challenge program or not. During the first round, after creating a respectably-scoring contest program, I did but solely to verify my theories about the scoring algorithm. During the final, I did not: I knew I had not deduced what the change was so did not feel entitled to 'look in the back of the book' to check.

Other Discussions at Smalltalk Solutions

David Buck has a customer who wants some DLL libraries wrapped in DLLCConnect. He also described a Gemstone system whose managers insisted that the audit trail be in an RDB, prompting three resignations and thereafter being an immense drag on development. Simberon is putting Smalltalk courses on CD; these will be available in the Autumn.

Securitas is a Swedish security company active in all Europe, also the UK, but only the Benelux site where Joost Bossuyt and Wilm work uses Smalltalk. They use VA (they have looked at VW porting but the VA cost model - pay per development seat, not a percentage of revenue - suits their development style better). They are interested in Smalltalk training.

Will Loew-Blosser, who works at Cargill, is interested in porting from VSE to VW. The approach used to port ObjectStudio into VisualWorks may be worth reusing. Code exchange and discussion happened at the conference.

Boris Popof works in Vancouver at DeepCove Labs which is part of PacNet. PacNet's Europe division is based in Shannon and DeepCove Labs have had remote sessions (long timezone gap) to them to support their Smalltalk fat client; the database is in Vancouver and the fat client is both there and in Shannon. (See his talk for details of the application.)

Peter Hugosson-Miller works at CDC in Stockholm on a Smalltalk system for warehouse management, helping supermarkets ensure they always have stock. A team of 6-7 has been whittled down to just him and for the last five years management have talked of 'will be rewritten in (language X)' but as the system has 60 years of effort in it, this may never happen.

Chris Cunnington is a Squeaker who is targeting Java programmers who are getting disillusioned and want to try Seaside as something new and do their own programming. In addition, I suggested web designers who want to put behaviour behind a customer's website and want to pair-program with someone who can teach them something quick and productive. One might seek such people at web graphics design conferences.

Yann Monclair's name is actually pronounced Ian Monclair; he's a Breton and his name is the Celtic name Ian, using an old Breton spelling. Just like me, Yann is accustomed to being addressed by various pronounciations of his name and happy to accept whichever is offered until/unless people ask.

Follow-up Actions

Niall To Do:

- Send notes of the VSE-to-VW porting discussion at the Cincom Smalltalk User Group to Will Loew-Blosser at Cargill.
- Send VASmalltalk Method Wrappers code to Peter and David.

Conclusions

A thoroughly upbeat Smalltalk Solutions:

- All three Smalltalk vendors are expanding and confident.
- Several new-entry-seeking initiatives: GLASS, SAP/NetWeaver, etc.

As I noticed last year, a Smalltalk-oriented evening event was what we always had in 2005 and earlier but lacked in the 2006 and 2007 'Smalltalk Solutions within a larger conference' arrangements. The evening sessions in the noisy nearby Irish pub were good, but I missed the usual early evening reception or buffet or event as a forum for industry gossip.

VASmalltalk User Group Conference, Frankfurt, May 24th 2007

Joachim Tuchel welcomed everybody to the conference for ObjectFabrik. Nicholas Gilman did the same for Instantiations, stressing Instantiations' commitment to VASmalltalk in Europe and to keeping it a living IDE.

SmallTalk Industry Council, Georg Heeg

STIC's purpose is to increase the awareness of Smalltalk, and create awareness where it is absent. We need to recreate the old awareness, to reinstantiate it (maybe the company could call itself 'Reinstantiations' :-). STIC also promotes recognition of Smalltalk's superiority, attracts human resources into Smalltalk and supports standardisation between dialects. Since 2001 (when it was held in Chicago), STIC has organised the annual Smalltalk Solutions. STIC works with user groups like ESUG (this year the ESUG conference is in Lugano, Switzerland), CSUG, Frankfurt SUG, etc.

Cincom, Gemstone and Instantiations are the three main supporters of STIC. Locate STIC websites from www.stic.st (these websites are being organised; they are not so at the moment). Please join ($40/year). (If you want to do more than join, STIC is looking for a secretary.)

In future, STIC might support Smalltalk utilities (STIC seal of excellence), etc. STIC will help you to create Smalltalk awareness in your environment. If you need help talking to your managers, contact Georg.

For the first time in more than 10 years, all the Smalltalk vendors are financially strong. At Smalltalk Solutions 2007, Cincom announced ObjectStudio 8 and VisualWorks 7.5, Instantiations announced VASmalltalk 7.5.1, GemStone announced GLASS and their latest version of Gemstone 64.

VASmalltalk 7.5.1, John O'Keefe, Instantiations

Q. What does the 'VA' stand for now? Joseph suggested it stood for Very Active, which John thought was an excellent idea; until that moment, it

stood for nothing, since IBM did not licence Instantiations to use the term VisualAge.

John started Smalltalking in 1987, joined IBM in 1990 and left IBM at the start of this year to join Instantiations. He is learning to introduce himself as “from Instantiations, working on VASmalltalk” instead of “from IBM, working on VAST”. He is delighted with his new role (new in some ways, not new but the natural continuation of his work in others).

Instantiations offer leading-edge tools for Smalltalk, Java (Instantiations are an IBM partner), etc. John reviewed their history, which started at 1984 and ended at 2007 and was quite involved. The message was: these are leading, mature Smalltalkers.

VASmalltalk 7.5 works with Vista, SuSE Linux, Windows Large Address (customer images were hitting the limit), Native Oracle, Windows XP/Vista themes, Web Services, RB and Envy browser tools.

Q. What makes SuSE Linux special? Not much makes any Linux build special but to say ‘VASmalltalk supports’ the team must test thoroughly. VASmalltalk will probably work on many other flavours (John uses Ubuntu as a test platform and he knows GemStone runs VA on Gentoo) but they have not tested them. If you want to run on another platform, let them know and they may be able to advise. The problems are usually set up problems; the directory structure is differently organised, the shell processors are different, the set of locales available is not the same, that kind of thing.

Windows Vista: Vista’s user account control security system is a wholly new security framework in which administrators run as standard users. Applications and processes run asInvoker (the default: privileges are those of the user executing image) or highestAvailable (all the privileges that the executing user *could* have) or requireAdministrator (the highest security privileges). These are set by XML manifests and John showed the VASmalltalk 7.5 manifest: emsrv.exe, abtntsir.exe and setup.exe need requireAdministrator because they write to protected areas (the service registry, etc.). To start and stop emsrv you must have administrator privileges. In Vista noone is special and you will have to push the ‘allow’ button for your laptop. It was the same on XP, but users are often the administrator of their laptop under XP so they tend not to notice.

Q. (Joseph) Need an emsrv user as previously? It was only on Unix you needed this; no change.

John reviewed the per-machine and per-user-file registry locations (the former need update privilege). ProgramFiles is a protected area and cannot be modified by the users so standard install must copy them to user-specific locations (if you install non-standard e.g. to D drive instead of C drive then you avoid this; of course you no longer have the same degree of windows security protection).

Windows Aero is the Vista visual style: transparent glass design, subtle window animations, new colours. VASmalltalk exploits some, by no means all, in 7.5.1 (not in 7.5; this is recent work).

The help system in VASmalltalk does not use windows help but some add-ons e.g. GF/ST came with windows help files. These .hlp files are now deprecated so they have converted them to the new format .chm (they still also ship the old files).

SuSE Linux (and all others): VASmalltalk is X-windows-based for graphics, motif-based for widgets.

Q(Marten Feldmann and John; Marten is a customer who had helped find these points) If using other linux, don't use unicode support and do use ISO-8859-15, otherwise you will have lots of problems. The Installer must use Cshell, so symlink to get it. Newer Ubuntus and others have changed the locations of the locale files so you need symlinks to the old locations. (See the discussion forum, FAQs, etc., on Instantiations website.)

Q. Mac OS10? AFAIK, it will not run; by all means try and tell us how you get on.

Q. UTF8 problem will have fix in 7.5.2? They know it is serious and needs addressing. Priorities for 7.5.2 are being discussed.

Using Windows Large Address space lets images grow to 3Gb of memory (prior limit was 1Gb).

Q(Christian) when will you support 64bit? It is a large development effort that has not yet made it onto the roadmap. They are considering it and will do steps towards it (e.g. an image that needs 64bit-size will have GC issues at the moment; they will provided e.g. threaded GC before they do 64bit).

They now support Oracle's holding LOBs in files, pointed at from the database, rather than in the database itself. They fixed some Oracle bugs (stored procedure issues, etc.) and provide examples (in product and on website) because they received many questions about SQL in Smalltalk.

Q. DB2 CLI interface work? Not in 7.5.1. It is high on their list of things to consider for 7.5.2.

Windows XP/Vista themes are not enabled by default for VASMalltalk apps. If you want it, use a manifest (this is the best way to handle the standard controls) or invoke it explicitly in code if you have custom controls (if your custom widgets are in C, you must add your invoking code it C). They ship manifests with obvious commented-out enabling code.

Q. WidgetKit / WindowBuilder widgets? Good question; John will add the answer to this presentation when it appears on the website.

John showed what an app looks like in Windows classic, then in Vista with

no manifest, then with the full Aero look (assuming sufficient hardware; Windows will downgrade its look if the graphics adapter or memory are limited). The differences were mostly subtle, sometimes less so (progress bars pulse now; some people like it).

The Refactoring Browser framework and UI, and the menus to invoke refactorings from the Envy browsers, are now released with the product; each VASmalltalk release ships the Custom Refactoring project's latest stable release of that time, coordinating with the project to test on the upcoming release. (I remarked that, in the same way, any releases that we post on customrefactor.sourceforge.net are likewise tested on the latest release and derived from the shipped release, so downloading a released .dat from customrefactor.sourceforge.net and upgrading should always be straightforward. As it happens, a new release has this week been posted). SUnit and SUnitBrowser likewise are now in the release. John thanked the various open source people involved in maintaining these tools.

The ENVY/QA suite (code critic, code coverage, code formatter, code publisher) is integrated with the tools. Envy/QA is an extensible framework so can be a place to start if you wish to add further QA checks (but they know people are busy building applications, not tools).

Q. Is there documentation on how to use this framework? None recent; they ship the original OTI docs from way back.)

The VASmalltalk browser icons are now external, not stored as ByteArrays in the image, and so look different.

The above summarises the current state. What about the future? Version 8 (no schedule decided yet) or sooner will support Seaside (they do not wish to be unique amongst major Smalltalk dialects in not doing so :-). The main challenge is that continuations use the reification of the context and contexts in VA are very different from those in Seaside's dialects. They could adapt their context to suit Seaside. They could use ServerSmalltalk actors. They are mapping the Squeak classes to VASmalltalk classes and will have a non-running code port soon.

Q. I have experimented on getting continuations to run in VA; simple changes to process primitives may enable it and are necessary. John agrees and he can change the VM so he can do that.

Q. Make VM source visible so others can help? The licence agreement with IBM means they cannot do that, alas.

A seamless all-platform one-click install will be provided. They will have a wholly new documentation system (the current one is built with IBM tools).

They've had 5000 downloads. They have 2000 active users, 200 commercial companies using it, 650 support cases (many fixes).

Use the Instantiations forum ww.instantiations.com/forum, not the old IBM forum which is now little monitored.

VASmalltalk has a free evaluation version, fully function except for a nag screen.

Migrating from VAST to VASmalltalk, Joachim Tuchel, ObjectFabrik
ObjectFabrik have financial and telecoms clients who do much creation of web apps in VASmalltalk, in VW (Seaside) and on the dark side (Java Struts work, etc.).

- Example 1: a social security application, begun in 1995 and in service since 1997, using 720 classes and Oracle 9.2.
- Example 2: a public liability application (car accident call centre) 3100 classes in 58 config maps support 17 commercial applications; migration from 5.5 to VASmalltalk 7 went smoothly.
- Example 3: A credit processing, collateral management banking company had 9500 classes in 210 VA apps and 18 config maps. This application was very easily moved from VAST 6.0 to VASmalltalk 7.0.

How did he migrate? Import all code, continue developing in old library while making the system work in the new library, then import a delta and get it working. That's all? Well, not quite. :-)

VAST 6.0 and VASmalltalk 7.0 are fully compatible. Start a new emsrv (because the new emsrv has bugfixes and because your old emsrv probably runs on an old box IT wants to discard). Do not use a multiprocessor windows machine (IBM decided this old bug was a windows bug, which may well be so; it is not an issue with any other platform). Then install your new client which must be able to see the new and old libraries. Produce a complete versioned line-up for a first port. Continued development on the old library (often unavoidable) will necessitate a delta port later.

Applications do not define the version of their prereqs whereas maps do. Use config maps, not apps, to express your dependencies. If you use maps and use the 'move all required maps too' switch, you may import all you want and much you do not want, so he hesitantly advises that you do not do it and just put up with a few 'map missing' messages making you do a few more imports. If you want to use it for the delta map, that may be wise. Generally, removing all references to base maps is wise. (Can exploit z:... stuff.)

What next? Next you see 'Load failed'.

Q(Georg) You see the 'Load failed. OK' dialog so its OK, OK? :-)

Load from the base layer up. Exploit the preferences setting `cancelIfMethodsDoNotCompile: false` and get used to grabbing Transcript load-failure-explaining text and using the shadow browsers to examine unloaded code. Then you search 'Locate Selected Name' in configuration maps. Suppose you find something is deprecated; how do you find what has replaced it? Search for class names to see where it lives

now. It may now be loaded in the base in another app, eliminating the need for a prereq. You may find that your maps reference inconsistent versions of a prereq map. Sorting all this out may produce many intermediate versions so you may wish to export the result to yet another library. Generally, the cleanness of your system

- non-circularity of prereqs
- freedom from base code prereq references
- up-to-dateness (e.g. no use of VAST 4.5 utility dragged forward)

will determine how hard it will be. Generally allow 2 - 10 days of effort.

One of his example projects migrated on feature release. This reduces effort (you have to do regression test anyway) at the cost of increasing risk; reduce risk again by doing the migration early in the cycle. Another example project shipped a purely technical release with no new commercial features; this was safe but less commercially valuable.

Don't plan the port; just do it. The moment you are in front of your browser, you are doing it: see the problems and solve them. Your developers are the testers for the migration; make them use the port.

Q. They moved their new code into the old library and ran both old and new against the one library. This lets them see the differences between the old and new libraries in the base and in their code. They also build a base image (i.e. of all utilities they use plus a config map of all their mods to the base).

Q. Oldest versions still in use? John has worked less than a year ago with customers who were on VA3 (and OS2). There are many 5.5 customers.

Q(Joseph) If you make changes to the base image, write test cases for them. Even a test that just checks 'any implementors of this selector' is useful.

Q. Can the 'Load fail' dialog tell us more: which map, which app or something? John thought this made sense. (Niall: mention Georg's 'class name in load failed widget' example in VW to John; ensure any such change is formatted to avoid that confusion.)

John stressed that they want to know what changes to the base code have been found necessary. They will evaluate them and so maybe you will be relieved of the burden of maintaining them.

X-Trade Experience Report, Werner Schirp, Maxess Systemhaus
X-Trade is a small ERP system for food and grocery retailers, now migrated to VASmalltalk 7.5. The company was formed in 1995, has 60 people, has 6.1 million euro turnover, and is located in Kaiserslauten, Graz, Hamburg and Barcelona. The core application is 100% Smalltalk (it has small Java apps at outskirts) with 800 persistent and 500 non-persistent classes, 1500 GUI and 500 report parts.

Why use Smalltalk? Because it is so productive. They have programmers who work in Java and Smalltalk and all agree on this. Their competitors are

significantly larger so they need this competitive edge. Performance is key in this domain and their system is one of the best. (Trumpet it: a Smalltalk application out*performs* its rivals! So much for 'Smalltalk is slow' claims.) Product stability and robustness is important and they find their customers do not have problems they cannot reproduce, unlike in other platforms. Versioning in Envy helps them manage the many specific features that each of their 10 customers want. They promise their customers that all specific changes will be rolled into the next standard release and without Envy this promise would be hard to keep.

They have made specific single X-Trade features into web services (customers want this) and now want to move to a wholly service-oriented architecture so customers can integrate X-Trade with their other software in detail. Next year they plan to move X-Trade wholly into the IBM service architecture (and get the certificate from IBM) so customers who also have IBM tools will exploit it with their other tools.

Q. Database access? They have an in-house framework, optimised for batch operations and suchlike.

VASmalltalk Web Support, John O'Keefe, Instantiations
Web Connection was the first web support framework shipped with VAST. Servlets and Web Services came later. Web connection serves web pages. It offers a high performance HTTP server, either all Smalltalk using Server Smalltalk Runtime if desired, or you run behind Apache, IIS or whatever. The web connection visual builder lets you create the pages much as the standard GUI builder let you build GUIs for fat clients, using visual parts that map to HTML markup, forms, etc. (You can also wrap externally-generated HTML files.) It offers non-visual parts to manage session data to track client state and suchlike.

The visual builder is not a high-powered web builder; it was designed to build forms pages. Therefore you can also build more expressive pages in other builders and then mark-up calls to your Smalltalk application in them via the usual `<% ... smalltalk ... %>` notation (and you can also wrap Java applets, active X controls, etc.).

Deploying behind an HTTP server (Apache, IIS) makes sense if many static pages are to be served in addition to your Smalltalk-generated ones. Install the server, copy and customise the .exe, .dll, .htm and .cnf files that Instantiations provides, then update the HTTP server .cnf and abtwsi.cnf.

Deploying a Smalltalk HTTP server (SSR) gives better performance if most of your web app is dynamically-built pages but Server Smalltalk lacks the load-balancing, security and redundancy features of e.g. Apache; it might be more appropriate for intranets than extranets. No setup is needed, just invoke as `http://<servername>/<classname>`.

Deploying behind a web application server (e.g. Tomcat, IIS) using the servlet interface: because CGI is loaded for every page access whereas the servlet interface is loaded once and then talks in-memory to the Smalltalk

image, the performance can be better. Install the application server, copy and customise .jar, .pro, .cnf, and, if needed abtwsi.cnf, etc.

Smalltalk is not prone to being infected by anything from the web; they have never seen an infected Smalltalk web app.

Q. Fast CGI? Not done at the moment; they tested it and perhaps could offer it easily.

Smalltalk Servlets are like java servlets. They take the standard get post head put delete protocol, run behind the HTTP server and are invoked in the same way: `http://<servername>/<classname>`.

Web services are described in WSDL, published in UDDI (if you do; maybe you will find the service in other ways; publishing has not gained the expected popularity). Find service by querying UDDI to get WSDL bindings and use this to bind a proxy to the service and so invoke the service over the network.

Web services have around since VAST 6. You can produce .wsdl files via `SstWSXmlGeneration forClass: ...`; alas, the tool is not yet very good. John went through code to deploy and invoke a service. He then showed the Web Services Demo, a tutorial to be used with the 'web services in 10 minutes' manual. (Niall: see Alison Dixon's talk at ESUG 2002 for an example of using this; reach my report from www.esug.org.)

Other Discussions at VASmalltalk User Group

Joseph is rolling out SCRUM to Nokia and to Swiss Telecom. He and Andres are doing further work on SUnit releases. Since he reported on it at ESUG 2000 in Southampton (see my report), Joseph's infection scenario software has been used in some 50 research papers.

It looks like Christian has another customer for smallCharts. I will put him in touch with Will at Cargill re VSE-to-VW porting.

Dirk Verleysen and Hans-Martin Mostner could use a complete Toplink decompiler (or Glorp) to move old applications forward. John has Hans' partial decompiler and others and has had discussions to try to resolve this.

Joachim's example applications (see his talk) are maintained by teams in (first example) Wuppertal, (second) Hanover and (third) Frankfurt, all of whom had developers at the conference.

I had to leave to catch my flight, so missed some BoFs.

Written by Niall Ross (nfr@bigwig.net) of eXtremeMetaProgrammers Ltd

* End of Document *