CS21 and ESUG 23, Brescia, July 13th - 17th, 2015

This document contains my report of the ESUG conference in Brescia, July 13th - 17th, 2015 (and brief remarks on the Camp Smalltalk on the Sunday before it). As there were parallel tracks, I could not attend all talks. At the end, I also report a 2015 UK Smalltalk User Group meeting I attended.

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 give my personal view. No view of any other person or organisation with which I am connected is expressed or implied. The talk descriptions were typed while I was trying to keep up with and understand what the speakers were saying, so may contain errors of fact or clarity. I apologise for any inaccuracies, and to any participants whose names or affiliations I failed to note down. If anyone spots errors or omissions, email me and corrections may be made. My thanks to the conference organisers and the speakers whose work gave me something to report.

My own talk is written up at much greater length than the others.

Venue

The accommodation was in old town of Brescia (called Brixia in Roman times). A capitol and arena dating back to Vespasian were on the same street as the conference venue, which was overlooked by the mediaeval castle. The Wednesday dinner was al fresco in the grounds of the Malvezzi in the foothills above the town (some of us mistook the first course for the whole meal, but we nevertheless managed to do justice to the rest. :-)). It was hot, even for Italy in July; fortunately all venues had air conditioning.

Summary of Projects and Talks

I give a brief Camp Smalltalk 21 summary, then the ESUG activities reports and the awards. Next I summarise the conference talks, sorted into various categories:

- Applications and Experience Reports
- VMs and Development Environments
- Frameworks and Tools

followed by the 10-minute talk track and Other Discussions. Talk slides are available at http://www.esug.org/data/ESUG2015/.

Camp Smalltalk 21

Camp Smalltalk 21 was in the Centrale Pastorale Paolo VI on the Sunday before the conference.

I worked with Miltani and Alexandre on Roassal2 in VisualWorks. The VisualWorks 8.1 release contains updated Roassal 1 parcels: the coverage tool now works as intended. We may offer Roassal2 parcels with VisualWorks 8.1.1; meanwhile, loading Roassal 2 version 2.21 into VisualWorks 8.1 works OK. (In MacOSX 10.9.5, the camera sometimes shows 4:1 ratio, a known issue, but in 10.10 it works fine.)

The Roassal2 version of the coverage tool has not yet been ported. (Allhandro Infante, another ObjectPeople staffer, rewrote Spy completely in Pharo; it collects much more data and is sgnificantly better.)

I also pair-programmed with Thomas Brodt fixing a Cairo bug in his complex VisualWorks application. It does very detailed health care monitoring and uses an impressive number of icons (nurses in given areas know the icons that matter to them). We then looked at his use of Roassal. Thomas embeds Roassal graphics in his application by using the ViewHolder widget in the VisualWorks UIPainter (as does the Roassal2 TRExampleApplication). We discussed providing a more specific UIPainter widget, with a pane to enter the Roassal script for the graphic.

Thomas discussed with the Roassal team his need for the Roassal popup not to be confined to the Roassal canvas. He would also like the overrides removed (I reviewed some with him) and more use of the Platform dialect-neutrality class.

(As I was very focussed on this work, I don't have notes of the various other projects going on at the Camp Smalltalk.)

ESUG Activities

Conference Welcome and ESUG Activities Overview, Stephane Ducasse

Stephane thanked the sponsors and asked how many were at their first esug: 10 or so hands were raised. He thank Luc Fabresse (ESUG's treasurer) and Damien Cassou, and the local organisers Glauco, Erica and others. He showed the schedule from when ESUG was last in Brescia (the first talk was Jay Almarode presenting GemStone, which takes it back; I met Lisa Almarode at a Smalltalk Solutions conference in this millennium but I last saw Jay in the 90s). This year they had 15 student volunteers.

ESUG sponsors SummerTalk projects. They sponsored 4 articles this year - same number as last. They sponsored lectures in the Ukraine, Prague and Cameroun. If you want ESUG to sponsor something, write a *short* email and ESUG will say yes or no. Stephane ended by stressing the main purpose of attending a conference: "Talk to people you do not know."

This year, ESUG had 94 participants, 15 students and 4 free entry. (The smaller number than usual simply reflects the unusually early date of this year's conference.) ESUG 2016 will be in Prague. ESUG 2017 will be in Lille or Barcelona.

Innovation Awards,
This year, LAMRC gave 3 prizes for best IWST papers. There were also the usual innovation awards for which we voted "en votre ame et conscience." The winners of both were announced at Wednesday's dinner:

- **GTSpotter** came first. (See Tim's description of it at UK STUG at the very end of this document.)
- **PharoJS** came second. It lets any Pharo app run on any platform that supports JavaScript, subject to some limitations: it does not support `become:` or weak references. It does support block closures, metaclasses and reflection.
- **RewriteTool** came third. This is a development in Pharo of the VisualWorks / VisualAge Custom Refactoring project work, making the RB's rewrite tool more accessible.

See http://www.esug.org/wiki/pier/Conferences/2015/Innovation-Technology-Awards for details.

Applications and Experience Reports

Cincom Smalltalk on a Tablet, Andreas Hiltner, Cincom
Andreas got a tablet that was not powerful enough to replace his desktop machine but much more portable. He tried out touch gestures. Windows support for this is not new. It works by sending events that can be configured for each item, each window - receive them or not. Gestures are Pan, PressAndTap, Rotate, TwoFingerZoom and ZoomGesture. They are sent through VW's Announcement system. He showed short video of zooming in and out by gesture on the new OST workspace, then another of rotating a pie chart (note labels on pie slices are redrawn on each rotate). There is a display change event WM_DISPLAYCHANGE sent when the tablet rotates. The class UIMonitor gets info about the available screens and the default screen (and the device name but Andreas can't see a use for that). The DPI is useful to know - you might need to scale up or down - the scale factor likewise. The motive was mainly to read more easily (c.f. the workspace example) and to rotate, because a tablet has a smaller screen than a desktop and is more often rotated, i.e. to make it easier to develop in OST/VW on the tablet.

Homological Smalltalk Algebra, Leandro Caniglia, Caesar Systems
Leandro uses Smalltalk to support advanced scientific research in Homological Algebra in a project that started in 2009 and was revamped this year. Leandro was a mathematician and still has friends who work in pure maths. Leandro told his friends Smalltalk can do anything. One day, the friend came to him with a homological algebraic problem. The paper was 40 pages of homological work that actually calculated stuff (unusual - maths is that art of finding relationships between things and never actually calculating anything). The paper was about deeply nested relationships.

Leandro started his app. The papers structures were complex and Leandro pared him back and back until they reached the integers:

Z = {0, +-1, +-2, +-3, ... }

Next polynomials

Z[x] = {a0 +a1x + a2x^2 + ...}

Building maths objects needs making decisions: is the polynomial merely a data structure or is it a first class object? The latter was the choice as this was still near the start. He needs the ring of all polynomials as a first class object too. In Smalltalk, they have the class Integer but not the set of all integers, likewise Fraction but not the set of all Fractions.

Next came

z[x, y] = { sum of all a_{ij} x^i * y^j }

This paper was about non-commutative algebra. Leandro defined that as a constraint so he could define the base as commutative. He gave as example xy - yx + $2y^3xy$ ... where the first two terms are not guaranteed to cancel each other). Of course, the actual case was

Z[x1, x2, x3, ...]

For a year, Leandro worked with his friend at the weekend (Leandro has a day job). Leandro found that after thinking for an hour he could always work out how to keep going without adding too much complexity.

Any function has a tangent line in one dimension. In 2d you have a surface and the tangent is a plane. And so on. All this can be translated into linear algebra (linear geometry). Next he discussed differential forms. These appear when you translate differential geometry into differential algebra. Because this was non-commutative, it produces non-commutative geometry and "not even mathematicians can imagine that". He showed us the equation

omega = set of all differential forms

Derivative of product is derivative of 1st term and of second term and ... . Next they put these into Tensors. He showed these as cross-products, which Leandro, lacking the font, showed as a red o superposed on a red x. You then have direct sums of these things. Mathematicians understand these by thinking in terms of lots of diagrams.

```
Y^3 -> Y^2 -> Y1 -> Y0
  |      |
  Y  ->  Y
```

where the push-out (Leandro did not call it that, but I happen to know that is the technical term) must commute (i.e. two routes through the diagram from the same start point will lead to the same endpoint), where the arrows (functions) map between objects in the various spaces Y...

An entire row or column in the diagram can be regarded as a single object, with all the arrows to its nodes as a single function. Having thus reduced the diagram of many nodes to two nodes, they get a new diagram made out of lots of these things.

All this can be modelled in Smalltalk in a natural way (easy it is not, but it is natural).

So the mathematicians could reason about these, but could not check the paper. He showed the exponents in the paper - even the exponents were very complex mathematical expressions. By hand, these cannot be checked but in Smalltalk they can be checked. Leandro modelled things and ran tests, and iterated round this, sometimes finding cases where Leandro made a mistake in mapping from the paper and fixing those. One day, on page 30 of the paper, one of the tests failed and it was *not* Leandro's fault. The mathematician thus finally found the mistake. Smalltalk did not write the proof but it let the mathematician write the proof. “Thanks! I'll come back when I've done the paper.”

He came back 6 years later (!) and they began work again. He wanted to start over from the beginning, and so refactored the system, and so hit the place where the paper was wrong. After that they found more mistakes in the paper. By now, Leandro was suggesting to him what minor changes could make the tests pass and these influenced how he went about proving his paper.

Leandro's company does many statistical probability calculations, e.g. just to compute the mean value

mean = x1 + x2 + ... / total.

and there is variance and other stats values to compute. How do you compute these quantities in parallel, given that they all depend on the average and the average depends on the entire sample. There is a theoretical solution to this problem. The paper was full of maths. Leandro wondered if the paper was right; he could see a mistake in a summation. He therefore used his earlier work to reproduce the paper checking its formulae. he opened his system and demoed. The result was a huge polynomial that no human could check was equal to what the paper claimed to proved it was equal to. He showed what the individual objects looked like in his system.

He eventually got his tests passing and so ensured his transcription of the paper was right. Eventually he found he was able to translate mathematical papers into Smalltalk sight-reading.

We should be involved in non-computer-science research. Smalltalk is productive there too. The experience of working with his friend was productive for both of them. There are some fundamental coincidences between maths and smalltalk. Both are axiomatic: objects and arrows, objects and messages. Mathematicians study things they define. Smalltalk is the same; define the objects you will use. Maths is compositional: structures and morphisms v. inheritance and polymorphism. Maths is about formal proofs. Smalltalk is about execution and testing.

Finally, Leandro emphasised that the Smalltalk did not prove anything. The tests found mistakes. The mathematician found the proof.

Q(Niall) valuable work!

Q. How much redesign? Leandro had already modelled mathematical objects years earlier so he already had a basis for this work when he began it. Thus he already knew to model first class objects from the start. So he had many iterations but fewer than you'd think.

SciSmalltalk: Doing Science with Agility, Serge Stinckwich
Serge is assistant-professor at A Paris 6 University and adjunct researcher at IRD (Institut de Recherche pour le Développement) in an international joint research unit called UMMISCO working on computer science modelling of complex systems. The work targets developing countries. Serge stressed that he was a computer scientist, not a mathematician. SciSmalltalk (MIT licence) on https://github/SergeStinkwitch/SciSmalltalk runs on Pharo 4 and 5 (and Squeak 4.5 v1.02). Competing software would be Julia, SciRuby, R, SciPy.org, etc. Didier Bessier did most of the work; his classes handle many numerical methods. They have also integrated Nicholas Cellier's arbitrary floating point arithmetic, and contributions from several others handling complex numbers and quarternions, random number generators, etc. There was a book by Didier Bessier 'Numerical Methods in Smalltalk and Java'; they have reused the Smalltalk (only!) part of it in Object-Oriented Numerical Methods - an Introduction in Smalltalk. (He thanked Didier.)

They had the support of two Google Summers-of-Code. They have 600+ tests for 77 classes and 185 classes without tests. He showed the Roassal visualisation for the system. He demoed calculating pi to high precision. He showed a calculation of a large polynomial where the precision greatly affects the accuracy of the answer: the right answer was -0.8...many digits... but in ordinary Smalltalk floating point you would get a positive number - a very wrong answer. Precision of 200 is needed to get close to the exact answer and nearly that precision is needed to get even close to it. They are working on IntervalArithmetics which will soon by in SciSmalltalk and which addresses this area.

He showed a disease infection model: numbers and rates for immune, recovered, infected and uninfected people. He ran a Lorentz attractor demo and showed its graph.

SciSmalltalk is a poor name: they would like a better one. Stephane is subdividing the whole system into modules so you can load the bit you need. Pierre Chanson (of Object Profile) is trying to reproduce the R system in Roassal. Any other contributions, especially statistical ones, are welcome. He closed on a dramatic view of a computation whose output was a black and yellow galaxy-like shape.

Live Robot Programming, Johan Fabry, Miguel Campusano, Pablo Estefero, Computer Science Labs, Universidad de Chile
He showed the large sheet of paper with some artwork (modern, minimalist) which the robot 'Dora' will follow. People think robots means h/w: wrong!

He saw a video of a research robot fetching beer. (He is now in Chile but was originally from Belgium; this story touch his soul! :-)). Thanks to Chilean government, he bought the robot. Then he told the robot to fetch him beer and the robot didn't. Moral of this story: software is fundamental.

Johan teaches a 2nd-year engineering course. Students must build a robot to follow a drawn line. He showed two groups' solutions, one fast, one slow - first video could be shown twice while second was shown once. With the same h/w and the same time to program, two groups had very different results. Time is money in industry. Time is research in academia. Time is brain-power everywhere. Every problem has innate mental complexity and also the complexity of dealing with the technology you are using.

An example: two weeks ago the very expensive robot stopped working. It took an afternoon to remove the two PCs in the base of the robot (no easy access) and see 'BIOS error' and do F1 and discover the motherboard battery died and replace it. The battery costs < £1 but 2 weeks of no use of an expensive robot cost a lot more.

He wants to spend time on innate complexity, not trivial complexity. If Dora stops, did she stop because program complete or because sensor failed or motor failed or software wrong or ... ? You can spend a lot of time on that. You can spend time just tweaking the program parameters because process is do tweak on other machine, reboot robot, put back to start, upload code and resume.

So Johan wants an immediate connection to the behaviour the robot is executing. Smalltalk is a live programming system but he is now talking about another live programming system. He showed a video from Bret Victor's talk in another conference on code to draw a tree. When he changed a code parameter, the tree immediately redisplayed. WHen he put a magnifying glass over a bit of the tree, the code that drew that bit was scrolled to and highlighted.

Live Programming fundamentals. His language LRB is live, controls robot behaviour (i.e. *not* image analysis algorithms, machine learning or whatever). It uses nested state machines. He started setting up Dora (she is slow to boot) and showed a slide of the coding environment: LHS writes Pharo, RHS shows state machines. He created a simple tick/tock machine and showed code creating state machine, dragging state machine alters state in code, drawing state machine transition arrow appeared in code, etc.

He connected to Dora over wifi to use this IDE to control the robot. He started up the interpreter, gave it Dora's IP address. He coded a very simple line-following state machine: if sensor sees white, turn right till see black;

if sensor sees black, turn left till see white (i.e. the robot bounces along the line). This is not just classical state machine because the robot must actually read the sensor from time to time. Standard smalltalk code is written in brackets within the state machine code. He then gave it a start state. He moved the robot over the line and it transitioned from state white to state black. Next he told it to start the motor when in white, until it is in black. The robot started then stopped but he can see it is in the black state: he can see it stopped because it was meant to stop, not because anything was wrong. (Applause!) He then added motor behaviour when in black so it would follow the line. (James Foster leant him a flashlight so the low-grade sensors could actually see the line despite the low light level in the auditorium.)

Q. It could have stopped for many other reasons; you know the reason because...? It would be a big coincidence that the battery should die at the exact moment the sensor registered black and moved to the state where stop was intentional.

Q. Why no errors? It is a very simple example. When programming robots you have errors all the time. That's why you use nested state machines. You have standard state machines for low-level operations and these state machines capture errors (e.g. low-light state) and present as states to the higher machine to contain the errors.

Q. How much for Dora? 250 euros or so; Dora is just a toy. The beer machine is *much* more expensive.

He showed a video of work of Pablo. It combined a view of the robot with one of the programming environment showing its state machines, gradually getting more complicated. and the robot's behaviour gradually getting more sophisticated.

His example took him 10 minutes to write. If he wrote it as his students who use Java do it, it would be two hours. People write robot software in C, C++ or, if they are lucky, Python. He is going to the other extreme. The immediate connection between the programming and the robot speeds research.

More details at http://pleiad.cl/LRP

Q(Stephane) Can add state dynamically? Yes! So how do you debug Petri nets? Robotic community does not talk about that. Yes, but how do you do it? Lots of “I got to this state and I don't know why...” :-(So we need trace of execution? Yes; Johan is planning to (get students to) do this.

Q(Andres) You know mindstorms? Yes, this is called Lego mindstorms as it comes from there. Purpose? He wants to improve software because people doing robotics don't know how to write software. He also would like ti teach his students this so they did not have to write slowly in poor languages.

Q(Jerry) can this become a chaotic system as states are added. How will your Live Programming paradigm guard against this, e.g. if used in other domains? Good question, answer is he does not know; partial answer is to use live programming to evolve to a 'pretty-well right' state machine that you then reimplement in another paradigm. Another question continued this and the reply was that he can stop his interpreter / freeze robot at any time if things start to look like they are going wrong. He showed how a typo causes an error to be thrown at compile time which means that code is skipped (and logged in LRP errors window). It could be made to stop everything the moment there is an error.

LRP is a domain-specific language that embeds Smalltalk in itself (in a really clever meta way that would be a good subject for another talk).

Compliance of SOAP/XML standards does not release from extensive testing, Holger Guhl, Heeg (Stephan Melzig of Global Foundries could not attend)
(Oscar Neierstrasse fixed the beamer's ability to display Holger's slides.)

Global Foundries was formerly AMD (Advanced Micro Devices - changed name in 2008). Their site in Dresden is large: 3000 machines, 3000 VW images running. (The other site is in Malta in New York State in the U.S.) Large wafers of silicon are converted by robots into chips. (Just like Johan Fabry's robots) the robots, the chemistry and the physics needs to stay on the path. It takes weeks: they reduced production of an Athlon from 90 days to 60 days and they were very proud.

GlobalFoundries has their own software environment. They use VisualWorks, Tkl scripts, C#, Java. The attitude is pragmatic: make it work; keep up the yield rate (what you can get from a wafer). EquipmentInterfaces control all the garage-sized tools that drive the process. Many applications communicate with the tools though the EquipmentInterfaces. Everything is communicated in XML. The baseline is a framework that GlobalFoundries created to manage the machines; not 80 separate applications for the 80 different machine types but instead one application with 80 configurations.

GlobalFoundries ported their Equipment Interface (EI) software from Cincom VisualWorks 7.5 to 7.10.1. VisualWorks 7.10.1 contains major harmonization to SOAP/XML standards, which is good in general. Complying with SOAP/XML standards is valuable but it does not mean you can skip extensive testing. Just deciding to use standard SOAP/XML does not mean you are compliant.

XML is a 'standard'; your web service schema can be used elsewhere. When the project started in 2004, the XML standard was mature but the industry was not clear about that, or how to use it properly. In brief, they had no practice they had no experience and totally relied on the tools - VW tools and Java tools and ... The tools were not mature either so they created bugs. Every externally supplied WebService they received was buggy too! They spent money to verify them and they did not pass. Also the standards evolved; use of WSDL 1,0, WSDL 1.2 and WSDL 2.0 coexisted. You have no choice but to rely on the tools. Some of the tools were good, some were bad and some were ugly (i.e they worked but the learning curve was high). In VW 7.2 through 7.5, our support worked for importing and creating models, and had a good approach for object to XML translation, and the openness of VW meant it was the tool used to fix the bad XML of other tools. However these VW tools were neither perfect nor mature.

In 7.10.1, VW SOAP/XML is very near to the standard; he worked with Tamara Kogan (who was swift and effective) to fix the few bugs still there. However there were many thousands of SUnit tests and thousands of them broke. The old tests were producing 7.5-style VW XML, or the XML from other tools of that era, and that old XML had lots of bugs. SOAP is fire and forget: you do not expect a return code and you do not get one. This was a huge problem area because the far-ends they were addressing were of the kind that had their "own understanding" of XML. These old apps were understood by people who were moved to other departments or left or were fired. Global Foundries wanted that the XML to those far ends not change, which was impossible. They had to produce correct XML and the far end often did the wrong thing silently.

Morals

- If you have a key topic, e.g. XML, ensure you obtain and retain experts.
- Standards are good - if they are mature.
- Pay your technical debts.
- Improve your tests. Those thousands of tests that broke revealed that they had a problem - that was vital.
- Improve your integration tests, especially in fire&forget situations.

This story ends happily; the project ended successfully.

Software and Business Modelling Platform DynaCASE, Peter Uhnák, Jan Bliznicenko, Robert Pergl, Faculty of Information Technology of Czech Technical University
Peter is a professor in Prague and head of the Centre for Conceptual Modelling and Implementations research group. Currently, most modelling tools are either specialised for specific notations, i.e. not very extensible, or else versatile but with crude UI and less usable except by domain experts. Last year, two students over two semesters rewrote the OpenCASE tool in Pharo. The assessment was that this equalled two years of development in Eclipse. The reasons for this were the dynamic features of Pharo, the power of the Roassal library (he thanked Alexandre and his team) and the excellent community.

Jan then demoed, opening the tool's Spec UI. It has the usual panes with graphical models and lists. The basis is to define DynaCASE editors on model layer objects, the editors being paired with Views that present them. He opened a state machine, added connections, events. In the style of Roassal, he mixed Smalltalk writing with views of the Finite State Machine

Simulator to show that the platform does not care what you do with a view; it is the integrator that decides what to do next. (His slide drew a comparison with Eclipse, unflattering to the latter :-)). Next he showed Business Object Relational Modelling. Participants (customer, vendor, whatever) are represented by state machines. He added a participant 'Student' and defined its state machine. He reviewed the semantic validation and reporting (using Pillar) of these BORM models. Find it on http://dynacase.github.io.

They will be hosting ESUG in Prague in 2016.

Write everything only once. Smalltalk in government, Clement Noe
Clement decided (on his own) to rewrite a mainframe government application in Smalltalk. His guiding principle was to write everything only once. He feels that what he learned could be used by other government divisions, since this resulting application has more functionality, a fraction of the original code volume and is easy to maintain.

The RJV application takes data on hours worked and salaries paid as input. The Smalltalk application has two hierarchies, one for the basic data entries, another for navigation and totalling and verification. He does verification by using delegation squared: exceptions are sent to the RJVExceptionFinder. His CarrierTotal collects totals over various container hierarchies. An ErrorFreeCarrierTotal can total excluding unverified data. He addressed performance issues by caching and resetting within the carrier subtotals.

As always in government, new cases, new exceptions are being added. He used the CommandPattern for access control (undo/redo is its more usual use). He has done some work to use Glorp and ObjectExtender in this.

He was replacing a 20 years-of-effort COBOL implementation (which its builders never really finished). The Smalltalk app took two years. A Java rival was started in 2008 and is still unfinished. Much of the code he made in RJV can be reused in other government applications. He invited people to bid for more government work and reuse his code.

Point of Sale, Dario Romano Trussardi
This was an extra talk, or rather, showing of a video of the point of sale system. Using GemStone GLASS, Seaside and Pharo, he built an impressive system for shops and/or online shopping. It shows graphics of items on sale, handles purchase, stock managements, payment, receipts, etc.

Q(Chris Thorgrimsson) how many people took how long to do this? Just him, taking not so long with GemStone GLASS, Seaside and Pharo as it would have needed with any other toolset. How many clients? 2 clients (in Italy). Only usable in Italy or can it be multi-language? Yes, he used the multi-language stuff.

Q(Niall) The video is great; allow just a little more time for reading the large white slogans you have on some of the video (only another second or two - it's just too fast to read the slogan while taking in the visuals at the moment)? He agreed.

VMs and Development Environments

Cincom Smalltalk Roadmap 2015, Arden Thomas, Cincom,
Arden Thomas is the Product Manager for Cincom Smalltalk. He reviewed the Cincom talks, and the Cincom Smalltalk products. OST and VW use the same Cincom Smalltalk Foundation. The current releases are OST 8.6 / 8.6.1, VW 8.0, 8.0.1. Next month, we will release OST 8.7 and VW 8.1. By end year we will have 8.7.1 and 8.1.1 ready and next summer will be 8.2 and 8.8

There were major changes in the foundation in 8.0. We've been digesting those changes in 8.0, 8.0.1 and 8.1. The “millennium changes” are things that VW built before the millennium and now have changed: Text2 and SourceCode2. Text is used everywhere in the product and so the new widget, using the new UI, affects everything. Our new SourceCodeEditor absorbs or replaces many features that grew as goodies over the years but were needed in the core: autocomplete, editor themes, warnings as annotations (so they will not become embedded in code), pluggable APIs so you can build domain specific languages, and more. This is part of our “sharpening the saw”.

8.1 was called the stability release. The changes in 8.0 were great and we consciously decided to be cautious around them. The GUI flicker issues are addressed - we have double buffering. The old browsers are now deprecated.

AppeX is our new web framework, using SiouX and Xtreams. A core Javascript library on the browser connects seamlessly to Smalltalk on the server. AppeX supports Javascript themes, restful API, etc. Avi's “if I were doing Seaside again today, what would I do” was AppeX motivation. AppeX has Seaside integration, and a lot of examples to let you get started.

SiouX has RequestFilters. It supports Linux PAM. Glorp has improvements. Etc. So much for the foundation; now the products.

ObjectStudio runs on Windows and aims to be “the thinking machine for people in business”. A key change is the next generation UI. Foundation, database and related UI (e.g. gesture support) and performance enhancements have been done. The GUI testing tools have the happy aspect of being able to test any Windows UI, not just ours.

VisualWorks likewise has a key change to its GUI. Look&Feel2 uses skinning - Smalltalk skins on native objects; it aims to give the best of both worlds. No longer must you wait till an emulated L&F is written in Smalltalk to make you up-to-date with a rapidly-evolving deployment platform. We surveyed the Linux installations that customers used, choosing a newer base, but not one so new it excluded customers.

What's Next: there will be more millennium changes. Completing the next generation UI in OST, making Widgets2 in VW, especially the DataSet needs improvement - everyone uses it and everyone has customisations of it. The UIPainter2 will be a major item.

Q(Christian) what platforms? Still being decided.

Arden then moved over to general Smalltalk aspects. He looked at MatriX / Polycephaly. MapReduce comes from functional programming: map does something to each element in a list and reduce puts the results in a new collection. Smalltalk does this easily, of course. His slide showed code solving a problem (how many times a word occurs in a list of documents) in linear fashion and then in the MapReduce pattern. MapReduce problems are easily passed to MatriX as multi-image problems. These and other ideas were still to help you explain to others why Smalltalk is still the smart choice. To convince management, you may also need consultant/analyst papers. Capers Jones (Namcock Analytics) wrote a paper (published 13th July 2013) showing Smalltalk's exceptional productivity. Every year Stack Overflow does a survey of developers. Arden has an action item to alert the community (circa February 2016, probably) to make sure Smalltalk is noticed in that survey.

Amazon has offered a new font “Bookerly” said to be optimised for readability. If that is true, it's a good candidate for Smalltalk code editing. Monospaced fonts are sometimes popular with code editors, but of course they can have poor resolution, so harder on the eyes, and the uniform width makes it harder to see typos. Smalltalk used a proportional font from the beginning - and who cares about lining things up as tabs are sufficient. He showed slides of various fonts that have been popular in code editors. Verdana, an award-winning font, is the Cincom Smalltalk default. Bookerly is what Arden now uses as it feels like putting on a pair of glasses to move from another font to Bookerly.

Q(Stephane) licences for these fonts? Need to examine that.

The input family of fonts tries to be proportional while addressing the reasons some people prefer mono fonts. SanFrancisco is the ElCapitan and iWatch fonts. Arial and Century Schoolbook are other common fonts.

Norm Green, GemTalk, GemStone/64 update
On May1st, it became two years since they left VMWare and became GemTalk. They have had six product releases in that time, no staff losses, and financials are OK.

On July 8th, 3.2.7 was released; they expect to release 3.3 in September.

In 3.3 they have faster VM performance; they optimised native code generation on Intel64 so that faster integer math (25% better average). A faster Transaction log means faster restore from hot backup.

A customer asked for Smalltalk access to Unix system log; it's done. There is a new immediate class: SmallFraction. Class GsFile has native utf8 support and some key methods are primitives so faster. They support LZ4 data compression - 10x faster than gzip but only 80% of compression. This is now used in the remote VM to page server, remote GCI, Stone sending updates to remote standby server, etc. You can now encrypt Gem to Page Server - set GEM_PGSRV_USE_SSL. When a transaction aborts you can have a thousand old views on it; their disposal is now more aggressive, so faster. The statmonitor can have wildcards in its options for filenames, etc. The pageaudit is multi-threaded (20% faster).

3.3 has an object canonicalisation project. There can be many unique instances of e.g. the same string value, the same data. A GemTalk/customer partnership developed this framework.

An LDAP authentication used to start with an anonymous bind but some corporate environments would not allow that so now it can be done otherwise.

3000 sessions could have 3000 page server processes, so they have multiply-threaded the page server. His diagram showed how 3.2 had a 1:1 relation whereas in 3.3 one page server has many threads, thus there is a saving which for a few hundred is small but for a few thousand can mean gigabytes of memory saved. They now pre-warm caches by pulling all pages to mid-cache in a single round trip over e.g. a slow LAN, not take them one-by-one. A customer asked to be able to clear shared page cache; they're not sure why but the method now exists. The GC garbage collector's settings can now be changed straightforwardly from Smalltalk methods, so is easier than it was.

The GbjTest app on Android has been carried forward. It's not a product and they don't know if it is practical for anything. It's cool. :-) Bill Ericson works on this; by all means download it, try it, email him if you have input.

The graphing tool VSD 5.0 (written in TCL/TK) needed a lot of upgrade as it used things that were obsolete. (A customer was using it a lot.) You can now copy a point to clipboard.

GemBuilder for Smalltalk: they support VW 7.10.1 and earlier, and are now working on VW 8.0. Other dialects are likewise being upgraded to latest. GBS 9 will be out by end your and do VW8. GBS 10 will integrate the two code lines for VA and VW.

For 3.4? More multi-thread mid-cache server. Improve commit performance - fewer memcpy calls. Your suggestions are very welcome. For example, they are thinking of offering more login options. Postgres supports GSSAPI + Kerberos, SSL Certificate, RADIUS, SSPI. Which of these should Gemstone support?

Licenses: perpetual (buy once, own forever), subscription for 1 year, Value Added Reseller. In the GemStone community Edition (was 'the web edition) disk data size is no longer unlimited, otherwise as before.

Q(Christian) How's business? Good; kept all old customers and have some new.

Q. Pharo support? Dale Henrichs does that and is not here so he must ask him. No plans to support Pharo or GemBuilder formally as we need critical mass of customers who will pay.

Q(Christian) Free edition has GBS? No. We could discuss small-scale paid-for use of GBS with free edition.

Martin McClure, GemTalk, Concurrence in Smalltalk - beyond Threads

Martin's first slide read “Martin is getting the projector to work with his laptop.” :-) He then showed a diagram of a unicorn at a birthday party - or at least that was one interpretation of the icon. :-) Today is Martin's birthday. 30 years ago he attended a Smalltalk user group meeting at which noone present had ever used Smalltalk. When Apple released Apple Smalltalk ($50 for the diskettes), he ordered it and organised a meeting - for a date when it had not yet arrived. At the next meeting Martin was the expert - he'd used it and they had not.

Sequential Model of computing: one thing after another. Threads: each thread is sequential, all threads share the same memory space. (Called processes in Smalltalk; that terminology is now out-of-date. Now, process usually means something with its own separate memory space.) Threads have advantages (do more than one thing at once) but though you program sequentially you execute randomly; almost any thread's statement can execute before or after any other thread's statement. Thus you get 'race conditions'. In Smalltalk, you use semaphores and locks to control this, but if you overuse locks you risk getting deadlocks. So just the right number of semaphores is the goal and that means needing to understand the entire system, unlike sequential where you can often just look at the local statements before and after the one you are studying. It is also hard to test - 'execute randomly' means there's a huge state space.

Martin showed a talk given at the usunet conference in 1995 titled why threads are a bad idea. Ten years ago the paper “The problem with threads” said the same thing. “... threads discard understandability, predictability and determinism. ... Rather than pruning non-determinism when not needed, we should introduce it judiciously only when needed.”

Martin asked who knew the programming language E (some did but have never programmed in it). E has been around for 15 years and some old projects tried to bring E ideas to Smalltalk but none are active today - so Martin is attempting a restart.

Mark Miller's doctoral these was on the E language. His desire was to solve the composition problem: programs from separate teams being combined. Objects are great - when you only have one thread and your objects (and machines) are not malicious. (E is therefore mainly about security, but Martin's talk will focus on its concurrency aspects.)

E divides the worlds into Vats. Every object belongs to a Vat. A Vat has its own object space and a single thread of execution. Everything in a VAT looks like s Smalltalk program. There are two kinds of send: immediate and eventual. Eventual messages are queued for later execution while the sending program continues. Thus a VAT has a stack and heap, as usual, and a queue of not-yet-delivered messages. A turn is when a message is taken off the queue, delivered, gets its return value and that is returned to sender.

An object reference can be assigned to a variable and can be used to send a message to that object. Martin defined an object reference as the second meaning, i.e. as a channel through which you can send a message, since assigning to variable is merely saving a reference for later. E has two kinds of reference: near (only for use in that Vat) and eventual (can be used from any VAT). Near references can be sent near (standard Smalltalk) or eventual (send then continue) messages whereas eventual references can only be sent eventual messages. He showed an example system of two Vats.

Near references in the sending Vat become eventual references, except for data objects: immutables, containers of only immutables that are themselves immutable (e.g. a date holding integers), copy-reference objects.

So what does an eventual send return? It sends a promise: an object that is an eventual reference to the result. Every p[romise has a resolver for that promise. The promise stays in the sending Vat, the resolver is created in the receiving Vat and the two relate to each other. You can send messages to it before or after it is resolved; before resolution, messages are queued, to be sent as soon as it is resolved.

So how does this affect latency?

```
  workQueue removeFirst process = 'done'.
```

If the `workQueue` is a forwarder, you send it `removeFirst` (returns forwarder) then process (another forwarder) whereas with E, you have an eventual reference to a `workQueue` and send it `removeFirst`. It immediately replies a promise and process is queued on that promise; just one 'forwarder'. This is very useful because machines get faster but the speed of light continues to obey Einstein's law, not Moore's law.

Suppose a promise resolves to a broken reference. It signals the exception indicating broken reference in the local stack of the sending Vat.

E has a when-catch expression to handle broken promises. This can implement multi-way joins so e.g. you only proceed if all eventual sends resolved for a given case. You can guarantee order of delivery of messages.

Martin then asked: is this a good idea? The advantages are no shared state so no race conditions. There are no locks so no deadlocks. The model is straightforward. This enables distribution (which threads do not by themselves) and it maps better to multi-core. (Smalltalk VMs are shy of exploiting multi-core because the likelihood of interleaving is raised and, even more, garbage collection becomes a much more interesting problem - as Java found.)

There are drawbacks.

- You can have deadlocks where promise a needs promise b which needs promise a so execution can still halt. Anecdotal report in Mark's thesis was 60 programmers in E in several years had only two documented locks. Also datalock bugs are easy to find.
- Recursion can be an issue.

Martin thinks it is worth trying. So how easily can we add it to Smalltalk? Much of it is easy. You need a syntax for eventual sends. (E's semantics were Smalltalk-like but its syntax was Java-like.) You'd want better support for the data objects. Smalltalk has a small set of immutable objects plus a toolkit to make an object immutable but the syntax could be better. For data objects, should equal and == be the same; is a symbol with the same characters in two different Vats the identically-same object or not?

For E info, look at erights.org.

Q(Stephane) What would be the architecture of forming the Vat? Do we have a Vat that is Smalltalk (so with multiple threads) or take e.g. Pharo and disable the threading and add the communication. That means one Vat per operating system thread which is scalable up to a point and not further. There are two existing implementations, one Java, one CommonLisp, that use threads in the VM, one thread per Vat, and that way communication between Vats is faster.

Q(Andres) Should each vat's message queue be a first-class object in that Vat and should messages have priorities? E does not use priorities; it is strictly FIFO. they were able to build a large robust application without needing that. (Graham McLeod suggested you could assign priorities to a Vat as a whole as it is either a thread or a process and they have priorities, can be cancelled, etc.)

Q. Why send `eventual` to object in same vat? Firstly, you may not know it is. Secondly, `eventual` send can let your program continue and if you get an error you handle it the same way for all cases.

Q. Order of execution problem; the Croquet guys had sequential numbers? There is a guaranteed ordering of eventual message delivery. It is not a fully causal ordering (that is not a solved problem) but some problems are solved well by the order that is guaranteed. Croquet were keeping state synced by sending state messages, which did need sequential ordering.

VA Smalltalk Product Update and Roadmap, John O'Keefe, Instantiations

Instantiations is doing a lot of community outreach; conference sponsors, new technology. The VA Smalltalk forum is aggregated in http://forum.world.st. (The old forum content - prior to May2011 - remains available read-only.) Their website has several podcasts (many by James; alas, his untimely death means they are not being created as rapidly now).

The next release 8.6.2 is due in Q4 2015. This will be the maintenance release for the 8.6 stream. It improves class/pool/lookup and iteration; this affects people with 50,000 plus classes in your image. The OSObject symbolic field access has been restructured to be more efficient (also useful in 32/64 bit support). Zip and Gzip can be used for in-image data in collections, streams, etc. Reading PNG and JPEG is robust to the many non-standard-conformant JPEG files there are out there (i.e. ones with the meta-data somewhere in the file rather than where the standard indicates).

An abt.cnf file contains chunk-format Smalltalk code executed early in image start-up sequence to compute initialization parameters. For example, you can configure the image to open with the debugger breakpoints off - useful if you save an image with a breakpoint somewhere that stops the image opening. These code chunks are sorted into epochs, so code that expects that image startup has passed certain points can be put in a chunk that will be run after that point has been passed.

Scintilla can display files with embedded NUL characters (previously, it thought NUL meant end-of-string). They also have multi-lingual workspace support.

Refactoring: if you remove instvars when editing code but leave references, you have problems; the class definition and the method definitions live in separate compile units. There is now a checker that warns when you do this.

Their source-code compression functions were being hindered because different platforms and locales have different views of e.g. which code points are letters (windows has a much larger set than Unix). They now use their own definitions of what is an alphanumeric character. (Running a new 8.6.2 image on an older VM is warned about since that could be dangerous.)

V8.6.2 has Seaside 3.2.0 and Grease 1.1.14. They now show that this is ConfigurationOfGrease-JohanBrichau.304 and ConfigurationOfSeaside-JohanBrichau.242 to show cross-over points for anyone using VASmalltalk and Pharo. V8.6.2 will support OpenSSL 1.0.x and SST HTTPS uses TLS. GLORP has been updated and GLORP can use SQLite. The last release used InstallShield on Windows and now they use standard installers on Solaris, AIX and Linux. An AIX install used to take over an hour and now takes 5 minutes. On Unix they will support headless installs (by customer request: do not require a Unix machine that has UI enabled). The installer does validation earlier so any problems in installing to a given location are now handled better. Environments, developed in 8.6, knew nothing about prior versions but that is now fixed; you can even launch your old pre-Instantiations VisualAge images (sometimes :-) - that feature is not formally supported).

Documentation for the ENVY/QA feature has been added to their online library. V8.6.2 will run on Windows 10, Fedora 21 and Ubuntu 15.04 as well as the current list.

MultiLingual support. Their aim was to keep it all in Smalltalk but also provide good support for editing all the files - config files, scripts, web languages and other computing languages. So the design of the architecture that powers Scintilla and etc. was made pluggable. They therefore exploited this pluggability for their multilingual support. They also made workspaces read only what they needed, not the entire file before anything was shown. Workspaces dynamically adjust the algorithms for code completion, brace completion, etc., based on the language (the 'file type') the workspace is reading. He demoed (on the 64 bit) VM), showing how options can be set by users, and new options added for new types. He opened a file of C and showed how it highlighted keywords in comments in C, etc.

Q(Joachim) Did integrate inflate/deflate with HTTP? No (not yet).

Q. Image startup script - can this cause a security problem e.g. for licence checks? They only work at development, not in runtime, so your customers cannot bypass your licences.

Q. Virtual compiler for instvar references - how does it work in practice? It will give you a popup list of references and you must fix *if* you have virtual compiler switched on (you do not have to). John admitted this feature “took me a bit of getting used to”.

Dino2 - the Amazing Evolution of the VA Smalltalk Virtual Machine, John O'Keefe, Instantiations
Their existing VM is very stable and has had very few bugs over the last 15 years. It was developed using a proprietary (Smalltalk) VM modelling language. It was done 20+ years ago when gigabyte machines and processors were unknown, so it's optimised for smaller hardware. It had no C compiler dependencies because they were less effective back then. Thus Smalltalk was written to describe an operation and that generated portable assembler, which generated machine code that you then debugged. However this low-level modelling language obscures the algorithms. The portable assembler it generates does not take advantage of new machine architectures. John then gave us “the eye test that nobody will pass” - a slide of the proprietary VM definition.

They have moved the VM to target producing C. The old VM code/spec was inherited by Instantiations from IBM who inherited it from OTI. Noone from the team who built it is still around and there is a minimal set of test cases. These were problems going to Dino2 but the good side was that the C code that the compiler produces is much better than the old assembler/machine code. The design of the existing VM was “I can jump anytime anywhere” whereas that does not work in C, so they needed to manage that. Having got the existing VM mapped to C, they worked incrementally (the rule was to have a working VM every day) replacing

Smalltalk-proprietary code with C modules. The 64-bit cleanup changes were made as they went. They can use 32bit ICs with the 64bit VM, so conversion could be deferred. An aim was to ensure users could make no/minimal/deferrable changes when moving from 32 to 64 bit. Their OSStructures used to use fixed offsets but are now elastic - they saw a negligible, almost imperceptible performance impact of doing that. This handled 64 bit pointers but gave additional benefits; it allowed custom packing, let them embed structures in other structures, handle anonymous and union, etc. Since they are calculating the offsets, they must ensure they have the information that calculation needs so they have a dependency analyser - ensure you don't have a circular dependency of structure embedded inside itself.

With this, creating the 64 but VM was just a recompile. Their interpreter benchmarks were within 80% before tuning. Writing in C instantly revealed that some of the algorithms were poor; they got a times-six improvement in one primitive by fixing that. (The C lets them recognise the algorithm intent - and so see when it is badly implemented.)

John started the demo video. The video explained that VASmalltalk uses native widgets to render and the layout can be radically different between 32 and 64 bit native widgets. So it's not just a VM change. A key benefit is the ability to use 64-bit shared libraries, e.g. the 64bit scintilla library. He started a lego game, then saved a 32-bit image, then reopened it from the command line. The splash screen appeared and then we waited for a while as it converted the image. (This is a whole image - you can also load ICs into 64 bit image and they will be converted.) After a minute or so, it opened and the game continued from where it was saved. He allocated 20Gb of ByteArrays, inspected, then dropped reference (GCable) and showed the effect on the memory the image was using on the platform.

They still have to tune performance and provide a JIT. The must provide a 64bit packager. Their GC is fast on average images; it will need work to be fast on images ten times that size and more. On Windows, the Alpha will be offered to selected customers in 1Q2016. An open Beta will be available in 2Q2016. Unix will come later.

Q(Graham McLeod) you have customers with very big images; they assist in testing? That is what the alpha will do.

Q(Niall) generating C let you recognise the algorithms? Yes - we could see it was e.g. doing a sequential iteration over the disk and when we changed it to use C shift functions performance improved 6-fold. In the Smalltalk autocode that could not have been seen because it was too involved in the low-level minutiae.

Q(Leandro) other than GC, which parts of the system need performance tuning needed? Two areas, one simple - they run all VM builds in debug mode so no C optimisation is being done - and the other already mentioned - find poor algorithms and fix them.

Pharo, Stephane Ducasse, INRIA
He showed the members of the Pharo consortium.

Pharo's goal is to be a system - and part of an ecosystem - in which to invent the future. An ecosystem means teachers; he showed a long list of universities where Pharo figures in the curriculum. YouTube has 1540 Pharo-related videos. See http://pharo.org.blogs and http://pharoweekly.wordpress. Thorsten's blog is good - http://astares.blogspot.co.uk. Pharo 4.0 had more than 76 contributors. Stephane thanked them all. He listed the research groups that make any use of Pharo.

Success stories are welcome, both technical successes and business successes. (Sometimes work is technically great even though the contract was cancelled for external reasons.) More books on Pharo are being written: “Numerical Methods in Pharo”, “Enterprise Pharo: a Web Perspective”. They will update “Pharo by Example”, and there will be an(other) Pharo on web book.

They are preparing Pharo immersion courses: 5 weeks of 1 hour video per week in English or French or maybe Spanish. INRIA has invested 40k euros in these videos. The aim is to offer these in April 2016. They are also doing a startup kit (motivated by the Cameroun visit - there was no internet there so they needed to be self-contained).

“Pharo is (y)our vehicle.” (Try modifying a core class and hoping to get your fix included! :-)) Pharo is not a black box. He gave the example of Sven who is continuously upgrading his Pharo server getting a better version tested by the Pharo people. If they had 1% of the resources of Java, Pharo would be amazing but they have the resources they have. Taking just 1 hour a week (e.g.every Friday afternoon) to fix something can make a difference. (Stephane fixed a widget last Friday.)

How to strengthen Pharo? Two years ago they created the Pharo consortium. INRIA manages it (for free). Many companies, institutions and user groups can contribute to paying an engineer and getting privileged access to that engineer. Access means they can influence the development path and they get engineering support time (limited amount of). Training and job posts are also offered. A full engineer for Pharo needs 50k per annum so needs 50 bronze members or 25 silver members or 12 gold and a silver. That paid for Estaban. There is also the Pharo users association. 99 euros premium or 50 euros normal members. That has funded useful work.

Next question: how to provide professional support for Pharo users. “I have this bug; help!” “We are 4 years on from that version; help yourself!” (they are not that mean in fact - they do help, but ...) “I want Pharo on Solaris.” “You will pay for that?” So a Pharo Pro service will be offered; Estaban will lead that.

Their goal is a major release every Spring, so in beta after Christmas. Significant bugs found in the dev version are backported to the latest stable version and that is a summer release. Pharo 4.0 was released in spring this year and Pharo 5.0 alpha is ongoing. Moose was migrated in two afternoons by two people; it is not hard to upgrade.

In latest Pharo lots of stuff got fixed, there is a cool Dark Theme, there are new tools. GTInspector lets you define what panes are shown when inspecting instances of a given class. He showed inspecting a Point class and being able to select its methods and see their byte code and etc. (Markus - not a UI guy - did that because he liked the tool.) There is GTSpotter - a search tool to find stuff in the image. Stephane started a video by “The Glamorous Team” titled 'Objects deserve to be Visual': a very quick run through looking at all sorts of displays on various objects.

Markus will talk later about instvars as first class objects ('slots') without loss of performance (likewise for class variables and globals). The active value pattern (from LISP - setting value prompts reaction). With these you can code relationships. You can also apply patterns, e.g. there are ~ 40 different implementations of how to cache values written here and there; a first class variable can be assigned one or other kind of cache behaviour. Avoiding loss of performance: Pharo has an open compiler so byte codes are used to achieve this.

They are working on better database support. The Garage project will unify API for DB drivers and Glorp will be brought up to date. They will improve reflection, e.g. stop and open debugger when you get to this state. Epicea will let you replay refactorings (PhD of Martin) and support code review. They will complete GIT integration. Spur is a new garbage collector. It doubles performance of a normal application (Spur itself is 10 times faster to do the GC). In Pharo they use Ephemerons, fast become: and pinned objects (no bouncing of them between old and new space). Spur has a new way to encode information in the structure of an object. Spur is in stabilisation phase in Pharo (not so advanced in Squeak). Spur handles recursive cyclic dependencies better. (They had hoped to have it released for ESUG but ESUG was unusually early this year.)

As well as Spur, they are working on OSWindow, Bloc, Brick and more. Spur runs in 32 bit but is designed by Eliot to run in 64 bit, so after releasing the 32 but they will work on 64. Sista is a runtime optimiser. It detects hotspots. A fourth level of optimisation can spot when a given stack run is worth generating assembly code (already in V8 and some Java VMs). When you save the image you save the optimisation (not the assembly but the knowledge this stack is 'interesting'). Sista aims to be ready mid-2016.

Another video showed events and gestures in Pharo. Block is a complete rewrite of the lowest part of Morphic. He demoed a list of a thousand varied-size elements in Spotter reimplemented on Block.

There are lots of external projects. Stephane showed a few. A smoothed particle hydrodynamic system displayed airflow over wings. Another showed a 3-d element in water on top of a screen display (as if the screen was the floor of a swimming pool) - not calling a library, all in Pharo.

Woden 3D is a three-D game platform in Pharo.

Reflectivity: Behavioural Reflection in Pharo, Marcus Denker, INRIA
Markus presented Reflectivity, a framework for behavioural reflection. Reflectivity was implemented some years ago as a research prototype but now has been re-implemented more useably in Pharo 5 in a practically usable way. Everything is an object? Well, code is not; it is just a String. However it can be represented in an Abstract Syntax Tree. The refactoring Browser's AST is widely used so his work is based on that. It has transformations, visitors and annotations (i.e. properties). Pharo reimplemented the compiler to use the refactoring browser AST. CompiledMethod>>ast returns it now. The inspector offers views in this object; the code view, a special tree view that highlights the nodes. Selecting a node on the LHS highlights the text in the RHS. Markus added the ability to highlight the scopes of where you are - he selected in an optimised block and showed its scope.

For any block, you can ask the AST node generating the block. This does not just decompile the block; it actually returns you the correct node, i.e.

```
  [1 + 2] sourceNode == thisContext method ast
blockNodes first
```

returns true. They added an ast cache so that the same query gets the identical node. (Q. cost in memory? They expected soon to move to last-recently-used but have never yet met a problem so have not done so yet.) The Opal compiler uses the RB AST and its visitors. They have an intermediate representation between the finally-visited RB AST and the byte code builder. They are no longer sure they need this. All visitor stages are pluggable so the semantic analyser or code generator stages can be changed. This compiler options are now very easy and robust. The byte code and even the IR is too complicated. The AST nodes can be given annotations. Basis (which he calls “The Evil Twin”) can create a reflective method paired with a compiled method. If a reflective method gets a compile instruction, it knows it cannot compile and instead gets a message send. These paired reflective methods are created on demand, and exchanged with the compiled method by invalidating. He did

```
  Morph methods do: #createTwin
```

As Morph is running, some will soon go back to being compiled methods; he showed that Morphs 800 methods were mostly now reflective but some had been run and so been changed back to compiled methods. So the process is to annotate the AST, create the twin if needed, and apply the analyser to it.

A MetaLink can be set on any AST node and it invalidates - result is like a method wrapper called before node is run (and in being run switched back to original compiled method). Markus confessed he took a couple of months to realise he could use a block, not a selector and receiver in MetaLink instvars. The options are before, after and instead (no around as yet - is it needed). The condition can be a boolean expression or block. The arguments takes symbols and seeks their bound values. You can use this for method coverage and many other things.

Pharo 5 will remove the old compiler and replace it.

Q(Niall) like method wrappers? Yes.

Q. Performance overhead? Just what you yourself demand. If you do 5 method sends for every method, you run slowly.

Pharo Evolution, Estaban Lorenzano
Estaban's job is to say no when cool but unready stuff is offered to him. Sometimes he must say no to Stephane when Stephane has invented the future but the present is not yet prepared for it, or drag Markus back from his reflected universe. Estaban reviewed the old history of Pharo, reaching Pharo 3 (“the first one where I started to feel a little bit proud of it”). The difference between Pharo 4 and Pharo 1 is huge, and has all been done by incremental continuous change.

The key question is why is Pharo (or Smalltalk in general) better than “Hyped language <x>”? We are good in dynamic contexts. We still dominate the “live environment” experience. We provide the best developing experience in the world. And much more can be said. But there is also a down side. Some of Pharo's technology was very cool in 1995 - for example, BltBlt blocks modern visualisation tools. The editor is not brilliant for editing small methods. The UI is mouse-based. And for CVS, Monticello is not good enough.

Nevertheless, Pharo has a good IDE. For doing what? Web is one - Seaside. Desktop applications - how should we publish them? In the cloud, mobile. Etc. The bottom line is Pharo has a good web experience but not yet a good desktop experience. Mobile versus the cloud is like quantum mechanics versus relativity. The desktop area is being addressed with Athens (vectorial graphics) and Bloc/Brick (replacing Morphic but with same paradigm), and OSWindow (finally, Pharo has multiple windowing systems), etc. He showed the improved look of apps in Pharo when using these.

For the IDE, they are adding the GTools, the Rubric editor, the Git tools and others: they want to integrate PetitParser and so recover Helvetia, get a clean version of Magritte, and add Pillar.

For the image, they are creating configurations for everything using metacello. They will finally be able to add a breakpoint without typing self halt. The image will become an object in its own right.

The VM will have Spur soon in 32 bits and in 64 bits later, The Sista adaptive recompiler should give performance gains. The ARM Spur is upcoming. A threaded VM exists; that will avoid e.g. freezing the VM when you have a long-running external call. The image polls at the moment (costs 5% CPU just to have image open); they plan to switch to an event-driven VM.

FFI is fundamental because they cannot do everything. At the moment, they have the original FFI (poor), Alien (deprecated), NativeBoost (assembly in image so needs a lot of effort to maintain: the change to Spur impacts it; moving to 64bit will likewise impact it). They will therefore drop NativeBoost and move to next-generation FFI.

Q(Christian) namespaces? If namespace merely means MyNS.MyClass instead of MyNSMyClass, the value is slight. If it means getting same-name classes from two development lines and not clashing, that is valuable. They are working towards environment-aware calls but it takes time. The importance is low from a business perspective.

Q. Mobile v. Cloud? They have microVMs from Android and other devices, but Pharo is not prepared for mobile UI-wise. The ability to embed Pharo in microprocessors makes more sense.

Q. Rubric? Make it work with the new text model, and make it look better (change back-end to something Athens-compatible).

Q. Magritte is workable but not well documented, and in parallel to the native classes; any thought about unifying Magritte by moving it into the Pharo core? No, because they want to keep the kernel small. Their plan is to make Magritte slot-aware. They are also going over the tutorial to improve documentation.

Hardware-Assisted Liveness on Reconfigurable Silicon, Boris Shingarov, Labware
Most of his life, Boris has written Smalltalk VMS. This talk is about hardware issues we must be aware of to have a Smalltalk for the next century. The basic problem is the end of Moore's law. How far do we want to apply the ideas of open-source? Do we get a technological base from someone (Microsoft?). Do we get chips from Intel or AMD and assume they are just there? If you are outside the target audience, you need knowledge of the system, i.e. the source code. Knowledge is a driver of reliability. Security is an another issue.

These are all software ideas but all these issues also apply to the silicon upon which you run your software. Boris therefore expects the silicon to become more open. The classical picture of VLSI design is an ISA becomes a design in an HDL and that is synthesized into a NetList from a technology library which becomes the fabrication of an ASIC. An FPGA is a random-access memory device. The core is a truth table: any combination of inputs with one bit of output, built up to an output through a flip-flop (so state depends on current input and what happened in the past).

Below 65 nanometres, Moore's law ceases to apply; more transistors just means they will all melt. Thus the boundary between hardware and software (the definition of ISA) has to be rethought; something not done since the 1960s. The line of separation can no longer stay in the same place because for example you need software control of your frequencies and voltages. Application-specific instruction sets, not just writing to an existing hardware chip, also moves the boundary. A key issue is how fast you can explore the vast space of ISA designs. You cannot spend 5 years writing an ISA for which some other team spends another 5 years writing a gcc backend. Things you can specify: composition (what registers and etc.) and the instruction set. You need a means to derive your toolchain and to synthesise a simulator.

He then looked at a very soft modification of an open-source micro-processor. This is not targetting an immediate problem; it is play, to increase understanding. You can buy a binary piece of intellectual property: precompiled netlist binaries. Instead, you can get one of the synthesizable cores. LEON (fault-tolerant SPARCv8) is used on the space station; these are used for real. Boris chose XILINX. His focus is debugging race conditions due to access to the same ports. When this happens from different processors, there is a bus, or cross-bars (much better performance). If his software has a race option, the synthesising could occur in various ways. If the synthesize is done in software, it can see not only what is being accessed but in what context. His device has an RS232 port that can be used to see what is being done and what state is resulting. The challenge not easy to overcome is if you make the whole chip out of blocks, the XILINX takes this as pieces of binary blobs. So even though we synthesised the SPARCcode ourselves, the XILINX knows nothing - you cannot open it and inspect inside.

Conclusion: since 1960s, hardware and software designs have ossified. Post-micro-processor, post-Moore's-Law, there is a new era and Smalltalk is very useful in that new world, because this ossification has not affected Smalltalk nearly as much as C or Java. This opportunity to re-learn and re-question is fun.

Q(Leandro). Assumes Moore's law invalid but technology stays the same (e.g. quantum computing could change this)? Yes, but surely we will not have the same old gcc compiler on a quantum computer.

Frameworks and Utilities

Windows Systems Tools in Smalltalk, Andreas Hiltner, Cincom

Andreas developed tools to help people test and diagnose bugs in next generation Windows GUI and OST. (GUI talk tomorrow, system tools today.) They already had a tool but it was not written in Smalltalk and so gave no access to internals.

Class PowerCapabilities manages whether there's a battery and what the battery can do, the settings for sleep, hibernate etc. It gets status notifications from windows (battery down 10%, laptop lid has just been closed, etc.). He can override system power facilities, e.g. do NOT shut down because my GUI tests are running so 'sleep after user does nothing for 10 mins' is to be ignored. He can also e.g. gracefully shut down database connections when someone shuts the lid while image connected to DB, and re-establish connections when lid open again, or whatever. He showed his battery widget.

He has a GUID generator tool to e.g. generate a UI for a COM component. (This is just a proof of concept.)

Windows Spy shows all registered windows - not just windows of its own Smalltalk process but also the windows desktop and everything else. So you can test non-Smalltalk tools; he demo'ed testing the windows calculator (the numbers on the windows calculator button are bitmaps, not text, something he discovered while preparing this demo). The tool can also get info about the process the window belongs to. He showed the tool with its list of all windows. There is a Windows Finder to locate the window you want. If e.g. your window will not stay at the top right, you can look at it and its process settings in this tool.

Process Explorer: Stephane in his talk mentioned an image that had had an uptime of 240 days; you could see that in ProcessExplorer - and whether any process is hogging the CPU, what GDI objects it is creating; is something greedy for resources and not releasing them? You can also see the user credentials, what permissions the process has, etc. He showed a list; some rows were in red because he did not have enough permissions to get their detailed values. (The system administrator running the same tool would see these rows normally.) The show can also list the loaded modules actually attached to the process. Anyone who ever worked in support knows that dlls can be anywhere and it can be hard to track which are loaded by the process. This tool's ability to show what is loaded and by what PATH has found and fixed several problems where the customer's PATH was wrong and they were loading an unexpected dll. Andreas ran a video. He found a process, and by menu item invoked a google search on it (Andreas hates it when an unknown process is running on his system).

The GDI log was really important. It helps you find resource leaks, which often guides a code review that can fix the problem. He demo'ed in a video on the calculator. He set the timer to 1 second (Dirk usually uses 3 seconds.) It showed graphs of GDI resources. As he used the calculator, the resources used moved up (the regions and fonts stayed the same, but the bit maps went up a lot - and then declined a little as there is garbage collection in windows, just with a lag). So you can test anything that runs on Windows, not just an ObjectStudio application.

Q(Christian) can you see single resources - see if a bitmap and font is holding onto things? Yes; you can identify the handle. You cannot close the handle of another process but you can clone the handle and then explore it and understand it.

Windows system programmers have these tools but not in one single package. You have to use three separate tools, mentally mapping info for one to another, to reproduce this kind of thing.

Q(Jerry) socket resources? That is the next step; ask Arden to let Andreas do it.

Q(Christian) can have in VW? Just load it, and its ObjectStudio prereqs, into VW. (This is Windows-only anyway.)

Smalltalk-driven GUI Testing, Andreas Hiltner, Cincom
Andreas talked about some of this yesterday. ObjectStudio has a new GUI so he needed to test it. He can record and replay events. He has a class OSWindow (much like Pharo's, same purpose). He wanted his tests in SUnit, not to be using an external test tool.

His first idea was to record all the mouse and keyboard events with their timings, and replay them later. Problem - some applications save state between one run and another (e.g. the Windows calculator does that). What you see is what you get (good) but you are using absolute coordinates, the whole recording has to be redone if anything changes, however small. He showed a video of his first approach to testing the calculator. For a small sequence he recorded 350 events. So he wanted to script: that needed access to Windows, and to mouse and keyboard events. You can retrieve a window with a given title, wait for a window with a title, get the window at a given location (maybe it always appears there). Then you can get the window menu, send it an event. (The operating system does not know about VisualWorks menus.) He can retrieve window title bar info.

He has to script keyboard input: string, one char, shortcut (not at all like sending a string), keys (LeftArrow, PageUp, ...). You may want to take a screenshot with or without a window frame. Dirk found tests could fail just though stopping momentarily in a run. The eventually worked out that he uses a different screen background in which the picture changes every 30 minutes AND he used partial transparency for titles - so the title bar could look different after a pause. So Andreas arranged to capture screenshots of windows without titles to compare. Screenshots can be saved as Bitmap, PNG, or JPEG (usually he just uses bitmap). Comparing a screenshot to a bitmap in a file needed the ability to highlight the difference, which could sometimes be hard to spot.

He talked us though a script that waited for a window title

```
wind := OSWindow
  waitForWindowTitle: 'Calculator'
  duration: 5 seconds.
wind isNil ifTrue: [^self].
```

If he gets the window he then sends it events - delete (to clear state) then a string (actually a number, of course, e.g. '1234') and then a return. Then he captures the image of what it now looks like and compares them to the bitmap in a file (saving the current to a temp file if there's a difference, as well as failing the test).

A line like

```
wind sendShortcut: 'ALT + 3'.
```

sends one key, not five.

```
wind
  clickAt: rect origin + (35 @ 200);
```

```
    sendString: '12345'.
```

sends 5 chars after clicking a button. He showed his video of running the test; set up was quick and running the test was faster (and could be run much faster but demo needed it slow enough to see, plus if testing hypothetical human user then better to use a possible speed). Lastly, he mentioned the power control.

Q(Christian) test VisualWorks? Yes. You must know the keystrokes and menu; get window location, get mouse location, compute difference, therefore call menu item

Q(Chris Thorgrimson) they use autohotkey and the tests keep failing when the developer moves the button. How to handle that? You ask the window for its buttons then ask location, of button, thus compute how to invoke that button wherever it gets moved.

Q. Is this in ObjectStudio or VisualWorks? You must have ObjectStudio loaded to use this. Ask Arden re use in VisualWorks.

Q(Claus) suppose one window - no separate window for button? In VW, ask in Smalltalk for subunits locations that calculate offset. On e.g. Skype, just know offset.

Cincom Smalltalk Security Update, Jerry Kott, Cincom
Our security frameworks received some significant upgrades in recent years, enabling the products to support all of the current versions of SSL/TLS. Jerry showed proper TLS configuration and discussed TLS cipher suite support, including work on Elliptic Curve Cryptography.

Security is becoming more and more important. Security can be overwhelming. Good security is invisible. Who uses security in their email? (Very few hands raised.) Jerry will give the big picture view then go into details. (The big picture helps him, not just us.) Security uses modular arithmetic, hashing, secure randomness, encryption and key exchange.

Modular arithmetic means operations on LargePositiveInteger: Multiplicative inverse (not simply a division), exponentiation and the usual + - *. Smalltalk shines at these. Look at the extensions in the SecurityBase parcel.

```
  16rFFF...FF
    raisedTo: 16rEEE...EE
    modulo: 16r01234......CDEF
```

would run you out of memory if these operations were not optimised. We have both all-Smalltalk implementations and external library calls. The all-Smalltalk implementations include MD5, SHA, SHA-1, SHA256; find these in the MD5 and SHA parcels. The libcrypto (BCrypt.dll on Windows) external libraries can be called. LibCryptoHash and BCryptHash classes do SHA-384 and SHA-512 (these are all in the SHA 2 family) and they are in parcel Xtream-Crypto (because efficient streaming is important). Use Xtreams.Hash, not deprecated Security.Hash.

```
hash := Xtreams.Hash algorith: 'sha1'.
[hash updateFrom: 'Message in the bottle!'.
digest := hash finish] ensure: [hash release].
^digest
```

There are also Xtreams hashing streams, e.g. DigestWriteStream.and DigestReadStream. You just keep writing or reading to/from the stream till you are done.

Secure randomness is much more complicated than the ordinary pseudo-random generators. Modern *nix provide randomness from operating bits of the system itself - see /dev/urandom. On Windows, there is a similar API. Our class CryptGenRandom accesses these.

Encryption is either

- symmetric (same key for encrypt and decrypt) such as AES DES ARC4 and Blowfish; the problem here is how to share the key, so they are used for a given message after a key has been shared by another method

or

- asymmetric (public and private keys) encryption such as RSA, DSA, (and ECDSA, ECDH new in VW8.1) in the RSA, DSA, DH and ECC parcels.

ECC stands for Elliptic Curve Cryptography. It offers more security for a given key size than RSA and so can instead be used for faster (smaller key) same level of security. The external library support for now only has libcrypto. His smalltalk implementation let him understand the protocol, but given that we can call external libraries, it is debatable whether we will support the all-Smalltalk solution.

```
alice := PrivateKey algorithm: #ECDSA size: 256.
message := 'a signed message' asByteArray.
signatiure := alice sign: message.
bob := alice asPublicKey.
bob verify: signature of: message.
```

This is in the ECC parcel.

DH (DiffieHellman) allows computation of a shared key from the private and public keys (and another parameter). It has been in VW as a Smalltalk-native solution since 7.1 and could be done via external library since 7.9. Elliptic Curve DH uses ECC to achieve the same goal. In VW 8.1 it is supported in TLSv1 and beyond in OpenSSL 1.0 and beyond. ECDH support in Windows should be in 8.2.

```
sPrivate := PrivateKey algorithm: #ECDH size: 256.
sPublic := sPrivate asPublicKey.
cPrivate := PrivateKey
  algorithm: #ECDH
  elements: sPrivate elements.
cPublic := sPrivate asPublicKey.
sShared := sPrivate derive: cPublic.
cShared := cPrivate derive: sPublic.
```

So SSL has been in VW since 5i, SSLv2 was never supported. SSLv3 is officially broken since 2014 (the POODLE attack). so in VW8.1 it is supported but raises a resumable exception TLSInsecureProtocolWarning. We aim to deprecate it in 8.2 and remove it thereafter. TLS is the successor to SSL (name changed because spec body changed ownership of specs). TLSv1 through v1.2 are supported. We support the renegotiation_info extension in VW8.1. we are adding more and more cypher suites and certificates. The SIOUX HTTPS config protocol is straightforward.

Q. Support secure LDAP? Maybe in 8.2.

Q. HTTP2? In 8.2 (8.1.1 will have some steps towards it).

The Roassal Visualization Engine, Alexandre Bergel, ObjectProfile (and Univ of Chile)
He opened the Roassal playground.

```
b:= RTMondrian new.
b nodes: (1 to: 10).
b open.
```

He showed this (10 squares in a line) then 1000 squares, then executed

```
b layout grid.
```

to show them in a square of squares, not a line.

```
b nodes: Collection withAllSubclases.
b edges connectFrom: #superclass.
b layout tree.
```

The tree layout of the hierarchy showed on the RHS pane.

```
b shape rectangle
  width: #numberOfVariables;
  height: #numberOfMethods.
```

He showed that, then changed to

```
b shape rectangle
  width: [:cls | numberOfVariables * 5];
```

and so on.

```
shape
  ...
  linearFillColour: ...
```

writing code to make colour vary with number of lines of code, then, more finely controlled,

```
b
  nodes: Collection withAllSubclasses
  forEach: [:cls | b nodes: cls methods. ...
```

and got a popUp - click on node, see method name and source.

```
b shape rectangle
  if: [:cm | cm linesOfCode < 5]
    color: Color green;
  if: [:cm | cm linesOfCode between: 5 and: 15]
    color: Color orange;
  if: [:cm | cm linesOfCode > 15]
    color: Color red;
```

and can sort

```
b nodes: (cls methods sortedAs: #numberOfLinesOfCode).
```

(He got a demo bug at this point, corrected a typo, then it worked.) Then

```
b edges connectToAll: #dependentMethods
```

and then forced layout according to a magnet analogy - so Pharo progress bar took a while to show layout of all elements. There are various layouts e.g.

```
b layout cluster.
b view restoreCamera.
```

and you can do

```
b normalizer
  normalizeSize: #numberOfMethods
  min: 5 max: 80 using #sqrt.
```

or you can use colour

```
b normalizer
  normalizeColor: #numberOfMethods
    using: { Color black. Color green. Color red }
    using: #sqrt;
  normalizeSize: #numberOfMethods
    min: 5 max: 80 using #sqrt.
```

The cluster graph acquired colour and size changes as he did the above. The following is new in Roassal2:

```
b shape
  bezierLineFollowing: #superclass;
  color: (Color blue alpha: 0.3)
  ...
```

Then he looked at graphs.

```
b := RTGrapher new.
```

```
ds := RTDataSet new.
ds points: {2@3. 5@4 . 8@9}.
ds ..
```

or

```
ds dotShape: circle color: (Color red alpha: 0.3).
ds points: Collection withAllSubclasses.
ds x: #numberOfMethods.
ds y: #numberOfLinesOfCode.
b add: ds.
```

Another example of a graph:

```
b := RTGrapher new.
ds := RTDataSet new.
ds points: {1@ 1. 2@2 . 3.2@3.2}.
ds x: #x.
ds y: #y.
b add: ds.
```

This showed a graph properly scaled which is in fact a harder problem than you would think. He showed other graphs where the display was sensibly scaled in relation to the data.

He opened the examples browser. He showed the animations - move mouse to create/drop images, drag images. He showed acceleration

```
  VITimer new cyclelength: ... seconds ...
```

He showed the right-left-up-down cone animation. He showed a country display, selecting x axis and y axis to be population, average city size, etc. - these choices displayed those data plotted on x and y. He showed non-graph pictures with axes, and boxplots. He showed a calendar, which looked nice and was defined with some subtlety - layouts of dates by weeks and months with Sunday dates in red, etc.). He showed the colour palettes in various layouts, themselves laid out using Roassal.

He then showed a long example where you could propagate highlight state between items. Then a graph where a button added each data point in sequence as the user pressed it.

The 'experimental' example category has some far-out examples of what they can do. One was an elaborate multi-line graph, with different colours under each line, able to be morphed slowly into another graph. He showed a slider that made the lines in a cluster compress or extend. The 'explora' is also experimental. Example began with one class, click on it 'explodes' to its subclasses and so on.

He then showed a graph exported to HTML and viewed in a browser - *with* the original click behaviour (in this case showing value of the point where mouse was) in the HTML. He scrolled down a long list of example categories (stopped briefly at Legend - can have picture legend appear only when you move mouse over it).

He then opened the Hapao test coverage tool. It is integrated in their Jenkins build tool and displays in Roassal of course.

They are writing a book "Agile Visualizations". Find their site on the web and see the existing chapters (more being written).

He then showed Roassal on VisualWorks (the video on YouTube). He showed it running on GemStone: Pierre will demo using it to visualise objects in a GemStone DB tomorrow. He showed INRIA Chile showing Roassal on a huge screen. He showed some AstroCloud data visualisations.

For 2016 they want to (go on having fun with visualisations) and write the book, find more work, etc. Sponsors: Lam Research, INRIA, individual donations, and the Chile Research Agency.

He had no time for questions as the talk took all the time.

Test Automation, Claus Gittinger, eXept
Claus showed a graphical representation of a bunch of blocks that perform actions with inputs and outputs that communicate, so a Petri Net. He showed the debugger tools; see trace and data. In testing, you simulate the outside world. The customer does not write Smalltalk code, but can formulate processes to explain their system. These processes can be expressed in these blocks, and then a new requirement can be expressed as a new block.

He created an action that produced a filename and added reading the contents. He added rotating the image to a constant angle, and then another to show the image, then nested them. He then showed running the block, ending with Stephane's face rotating round the screen. Next he added queues to apply actions in parallel to a succession of inputs. He next looked at what the action was defined in; eXcept can use Smalltalk or Javascript (some customers are more comfortable in the latter).

You make a test system out of this by creating blocks that do what the tests need. They have blocks to connect to databases, blocks to drive a Java GUI, etc. He opened the multi-pane window in which he could access and drive these. He created an action to click in a window and viewed it in tabular and petri-net/graphical view. You can input selenium recordings to these objects. It guesses the x-path for the components (usually, guesses a longer path than needed) when a recording is done. He showed this for a set text cursor position, get text, assert value is what was expected.

Test programming is still programming so you need to be able to debug your tests. He showed setting a breakpoint at a location in the petri net. The log of what happened, with every input and output, is also useful for that.

They have many plugins for remote control of iPhone, Android, etc. A plugin can also be to a database (your test of a web server might see webserver getting data from cache, never going o DB). He ran a test, saw a failure, and opened the report tool. It can generate a report in various formats, including Jenkins form.

Q(Niall) why would customers prefer Javascript to Smalltalk if not programmers? Airbus is a customer. Bosh (car control) are a customer. They must connect to their systems and so they use whatever is the natural language to connect to that system, so often they use Javascript. As well, they may task specifically their programmers to write that and the programmers do not know Smalltalk.

From Legacy Database to Domain Layer Using a New Cincom VisualWorks Tool, Niall Ross
(My own talk is written up at much greater length than the others.)

At last ESUG, I demoed a whisky-selling application – and brought a bottle of single malt whisky to motivate it. This year's demo is not a whisky website. But never let it be said that Scots are mean; I decided nevertheless to bring a bottle of Clan Ross single malt (a miniature; never let it be said Scots are spendthrift). Foolishly, I mentioned this to Andreas, who then told me he was bringing a bottle of German whisky to compare, so I then felt obliged to bring a full size bottle or two - to compete; never let it be said that Scots are not vain! (We offered drams of my Ardbeg and Talisker, and Andreas' Elbe after the talk in exchange for VW/OST feedback.)

We're upgrading our ODBC support from the 2.0 protocol to the 3.0 protocol (I demoed). Parcel ODBC3EXDI is in preview in 8.1; it will be integrated with the main ODBCEXDI parcel in the next release.

In VW 8, we provide drivers for Postgres 3.0 protocol. The driver in 7.10.1 and earlier used an older, increasingly-deprecated API, and was a goodie – technically, not supported by us. Its EXDI was a façade over another driver, and took some shortcuts that rendered some aspects of EXDI's API inoperative. Above all, it had a rare but hard-to-fix bug whenever multiple sessions ran on the same connection. (I demoed while talking.) Our new driver supports the latest API, and can connect over a socket as before, or (if Postgres' libpq library is in your PATH) using Postgres' C API. The new parcel is PostgreSQL3EXDI and PostgresSocketCnnection is the new socket connection class (use PostgresLibpqConnection. We encourage you to move to these. Pre-requing the old parcel name loads a compatibility parcel in which the old connection class name uses the new protocol – by default, that is. (The 7.10.1 driver still works in 8 and is available for download from Cincom's website. If you download it to your 'contributed', folder, it will take precedence over the compatibility parcel.)

The old socket driver never bound; it pretended to in its façade but the wrote the bound values back into the SQL stream. The C API can bind. (Our new socket driver can also bind, but until we improve the efficiency of its binary mapping, unbound will remain GLORP's default for it, and our recommendation for performance. Be aware of this if you force binding in EXDI.) Unthreaded, the C API blocks the image – not usually an issue but you might, for example, want to run a huge "set up your site" background query in the socket API while running small queries that service users in the C API. We also have a threaded C API, and may offer it in preview in 8.1.1, but we suspect the combination of socket and unthreaded-C will be preferred, so will not productise threaded-C unless customers show interest.

A more likely subject for feedback is array binding. A single unbound socket session can send multiple statements in a round trip, and get back multiple result sets. The C API cannot do this. It could however do what many another database's C API does: send a single statement with an Array of many rows' bound values. However, only users of Postgres who have paid EnterpriseDB for that feature can use this; Postgres is free, but Array binding in Postgres is not free. So we will wait until / unless we hear from customers who are in that situation.

And now a word about that rare bug in the old driver. It is our boast that our new driver survives inspection – literally: it always survives opening an inspector on objects mapped by GLORP to Postgres, even in multi-threaded scenarios (that is of course the norm for inspection, especially if done while debugging). The issue is that when two sessions are executed on the same connection, that connection's socket serves up their results in the order of execution. The user can ask for the results in any order and at any rate: all at once; next by next, ... A minor impact was that the Postgres 2.0 driver avoided some problems by getting all the results for a query when it was first presented to the EXDI layer. Thus a user executing next in a loop in their code *thought* they were getting their results one by one from the DB cursor but in fact the whole result set had already been read in the lower driver layer, potentially hurting performance. More seriously, functional disaster could occur if the two sessions were not running in the same thread. Bizarre mismatch between data expected and data actually read could occur if, e.g. you selected a (glorp-proxied) instVar in an inspector (which forks – and which terminates a slow reply). Any application forking a thread on an existing connection had the same risk.

In the non-threaded C API, the fix was free; an unthreaded Libpq connection has session-specific internal tracking, so gets its the result of each session back to that session. (But in the threaded C API case, the problem reappears. That's one reason why our threaded version – PostgresLibpqThapiEXDI – will only be offered in preview, unless there is strong interest.) In our sockets API, the most recently executed session can read the socket at its leisure, next call by next call. On execution, it forces any uncompleted prior "most recently executed session" on the same connection to buffer itself (this is done thread-safely). Thus we avoid the bizarre functional errors, and we only get the performance hit of the old Postgres driver on the rare occasions when it is unavoidable.

So what is the effect of these new database drivers on GLORP – and what is new in GLORP. First, a quick reminder of what GLORP is. GLORP lets you address relational database models as if they were made of Smalltalk objects. It hides the fact that Object-Oriented systems and relational databases have very different idea of how to model data. GLORP is the brainchild of Alan Knight, formerly head of our engineering team and now trying to make DART benefit from Smalltalk's insights. Before he came to Cincom, Alan worked on TopLink. That taught him a great deal about object-relational mapping. As the saying goes, first make it run, then make it right: GLORP was Alan getting TopLink right. GLORP has no ActiveRecord-like style restrictions, and, more importantly, GLORP remains flexible all through the lifecycle. It's not one of those frameworks that are high speed in a greenfields situation until you reach the end of their linear development process, then suddenly abandon you in untamed jungle. GLORP, the product of a 'make it right' phase, deals better with the real world. GLORP is designed to map efficiently to the external database interfaces of the various Smalltalk dialects, exploiting such performance aids as group writing (of statements), binding (of data) and array binding, when these are available. GLORP maps Smalltalk to the database but it also maps the database to Smalltalk. Rollback a transaction in the database and GLORP rolls back the state of those image objects that the transaction would otherwise have committed.

Now a reminder of how GLORP works. GLORP models the domain classes you want to persist, the tables in which you want to persist them, and the mapping between the two. GLORP s system of descriptors, mappings and joins let it solve this third and hardest modelling problem. (I showed my diagram of all this - from last year's ESUG talk, slightly revised - and used it as context for the following remarks about specifics. A Smalltalk image that can talk to a relational database already has a database interface layer, and that layer must already model the database' transactions GLORP's UnitOfWork handles the issue that GLORP must keep the image aligned with the database: if a transaction aborts, the persisted image instances must revert too. (The pattern is described in Martin Fowler's book 'Analysis Patterns'.)

Another area is handled by the DatabasePlatform hierarchy. In VisualWorks' EXDI layer, the job of ExternalDatabaseConnection hierarchy is to communicate SQL text and bound data between an image and a specific relational database management server. It handles the wire protocol. The job of GLORP's DatabasePlatform hierarchy is to map generic SQL and data into the specific SQL styles and data formatting quirks expected by a specific relational database management server. As Alan Knight once remarked, "SQL is a standard: that means it differs everywhere." In our foremost Glorp-using application – Store – we have up to now (by dint of occasional cheating) managed to pretend that the resulting mapping between ExternalDatabaseConnection and DatabasePlatform subclasses is one:one or at worst one:many. As the list of old and new protocols we support grows, we foresee it will become many:many, and that may mean our login UI must evolve. Some customers may have the same issues, so we're studying this for them as well as for ourselves. (The alternative of trying to figure it out after connecting – i.e. as part of initial connection, would in some ways be neater but it raises both latency questions and some "opening instructions inside" concerns.) I briefly showed a couple of possible UIs; later, I solicited feedback on our whisky stall.

Whether you code a mapping by hand, or let meta-GLORP generate the code for you, or use a mix of both, you only need enough information to disambiguate the system. GLORP deduces the rest of the fields and link tables that connect the ones you've mapped to classes.

Consider the simple case of database tables DRIVERS and VEHICLES, with a many-many relationship table showing which drivers have driven which vehicles, and which vehicles have been driven by which drivers. I showed a slide of everything we could tell GLORP about mapping to classes Driver and Vehicle, then looked at its elements.

```
classModelForDriver: aClassModel
  …
  aClassModel
    newAttributeNamed: #vehicles
    collectionOf: Vehicle.
```

If there is only a single relation in the schema between Driver and Vehicle then we can simply tell Driver's class model that it has a relationship to many vehicles: the manyToMany mapping in Driver's descriptor model will be deduced. Alternatively, we can tell Driver's class model nothing, but tell its descriptor model that there's a many-to-many mapping to Vehicle.

```
descriptorForDriver: aDescriptor
  …
  aDescriptor manyToManyMapping
    attributeName: #vehicles;
    referenceClass: Vehicle.
```

That will have the same effect; no more is needed. If Driver's table had a direct relationship – oneToMany to one or more main vehicles, say - as well as the more general manyToMany relationship to all vehicles driven, we could disambiguate for GLORP by telling it to look for a link table for this relationship.

```
    usesLinkTable: true;
```

If that was not sufficient (if there were several routes between the tables), we could explicitly tell GLORP the join from Driver to the link table, expecting GLORP could deduce the rest from that.

```
    join: (Join
              from: (driverTable fieldNamed: 'ID')
              to: (driverVehicleTable
                         fieldNamed: 'DRIVER_ID'));
```

If that was not sufficient, we could tell GLORP the link field(s) to use when seeking the onward route from the link table to the relationship's target,

```
    linkFields:
      (Array with:
        (driverVehicleTable fieldNamed: 'VEHICLE_ID'));
```

or simply explicitly tell it that relationship as what GLORP calls the reverse join.

```
reverseJoin:
  (Join
     from: (driverVehicleTable
              fieldNamed: 'VEHICLE_ID')
     to: (vehicleTable fieldNamed: 'ID')).
```

(As it goes from link table to target, not target to link table, 'reverseJoin' is the reverse of what the join from the target would be - not the perfect name, perhaps, but that's what GLORP calls it.) And of course, we can tell GLORP all this information, not just some minimum amount.

To summarise, how much to tell GLORP is a question of what kind of refactoring robustness you expect to need. Not mentioning the table fields at all is more robust to their changing, but less robust to the adding of new fields and constraints that could make minimalist definitions ambiguous. Decide based on how you expect the database schema to evolve - or on simpler criteria like how much typing you feel like doing.

My goal is to expand GLORP to its natural boundaries. The where: blocks of GLORP are so like Smalltalk that it's confusing when "would work in Smalltalk" does not work in GLORP. I aim to remove all unreasonable cases. Here are some cases that are solved in 8.1.

GLORP used to raise an error if you sent `in:` to an object mapped to a composite primary key (if you sent = or <> it just fell over in the DB). It's now OK to use these everywhere. In fact, it is more than OK; GLORP writes efficient SQL for databases that support it (on the others, it rewrites the query into ANDed non-composites, which works OK).

It is now safe to `alsoFetch:` across toMany relationships, plus you can `limit:` it. For example, you can ask for just one customer, and alsoFetch that customer's accounts to make it a single trip to the DB for performance, without worrying that 'just one customer' will become 'just one customer and just one account'. Furthermore, the accounts will be alsoFetch:ed in order if the mapping specified an order; in the old days it was 'just one randomly chosen account'.

You can nest iterators without fearing crashes. Dictionaries can have keys that are not in the scope of the mapping's owner-linkTable-value chain but just related to the value (see last year's talk). Rollback has more options and now never attempts to rollback literal Doubles (Doubles have indexed instvars – who knew!) and suchlike literals. An object with all key instvars mapped as relations, none directly, can be refreshed. Cache searches are not blocked when a filter field is not a primary key. GLORP numbers binding parameters in queries now, so it no longer repeats the same bound value used in multiple locations in the SQL string. We removed obstacles to parachuting an object into the registered objects list and cache. (N.B. doing such things is for GLORP power users only.) A self-returning Login>>asGlorpLogin improves polymorphism for mingling GLORP with other login approaches, e.g. while migrating.

Dirk and I have presented at past ESUGs how elaborate modelling tasks can be done in ObjectStudio's tools, and the models exported to VisualWorks for further programmatic development if desired. The new GlorpAtlasUI utility in VisualWorks is a vastly simpler tool, but useful for when you just want to get a quick first cut of a legacy database schema, or to study a fragment of a schema, or in a situation where the power of the ObjectStudio tools is not yet needed, or you cannot conveniently use them (or simply cannot conveniently be on Windows). I demoed using the tool to generate a very simple descriptor system (that Jerry then made the basis of a web app in his AppeX talk after lunch).

Q(Joachim) if you delete an object from a collections, does the collection needs re-proxying on commit? (We discussed offline.)

Q(Madhu) do you ensure `alsoFetch` checks the cache? Yes, we've done work on that.

Single Page Web Application Development with SiouX and AppeX, Jerry Kott, Cincom
SiouX is an HTTPS server based on Xtreams. It is very configurable and integrates with Seaside. AppeX is a web application framework; it uses SiouX (and Xtreams). The core Javascript library on the client connects to the server. The JS code editor is integrated into the VW IDE and what you see is what you get. It is OO so has a real class hierarchy and the meta behaviour that we are used to. ECMA script 6 has just been finalised; we had already implemented support for it.

AppeX is not a Smalltalk browser but one of the examples indicates how you could do that using Amber. It is not a multi-page web development tool but we do have examples for that. There are RESTful API demos, scaffolding (lets you build web pages from the server/schema). The jsonStream service type lets you serialise JSON objects directly to socket to stream and etc.

He noted which SiouX-* parcels did what, then demoed. He opened the server monitor. Each server has one or more responders - a responder is (the gateway to) an application. He opened a multi-lingual hello world. His slide showed the code to create and start a server. He did so and added a listener and then a responder. Responders are flexible but not very sophisticated. AppeX provides sophistication.

AppeX-Scaffolding is a mini-framework for creating web applications. He selected the Driver descriptor system I had generated, and its database, and then set his desired options and generated the Smalltalk and Javascript. He tried to start the application, got a blank browser, found all browsers were blank so restarted his image. Restarting his image cleared it. Deleting is supported but is not generated by default as the user must specify the semantics. He changed the themes and demoed this. A field that is required but left empty is notified - that nature of the notification is configurable (colouring title red is what he demoed). Events likewise will raise warnings when required. (He got an error, switched on debugging and saw the stack browser complaining that Time does not understand hour, so realised he needed to load a parcel of extension methods.)

Q(Chris Thorgrimson) You support web sockets on the server side? Yes.

Q(Chris Thorgrimson) you can link to external javascript on the client side? Yes; you can link to them; Christian Haider's code is an example of doing this.

SQL Queries on Smalltalk Objects, James Foster, GemStone
Traditional Smalltalks must fit their images into virtual memory and the object space if visible to a single virtual machine. GemStone/S is a multi-user object space whose size is limited only by the available disk space (terabytes or more) and supports thousands of concurrent sessions. Security is built-in. Everything is done in Smalltalk. Persistence is as in other Smalltalks - if reachable from a root, an object persists. The database view is isolated with repeatable reads, and the system does cache management for you.

James has been working on an SQL & ODBC interface. It is skunkworks but the code is available. It can be played with - and if there is a business justification it can be used for real. For example, many people use CrystalReports and that uses SQL over ODBC to extract data for the report from a database. There are some 50 DLL functions that one must support for ODBC. SQLConfigurationEvaluator is his main class. James opened a Pharo 4 image, loaded his packages and ran the tests. He then opened a GemStone 3.3, logged in and installed his packages from SmalltalkHub, then went to the tests and ran them. He then went to his local (Windows) machine's data settings and created an ODBC data source to reach his Gemstone. He then went to his C environment and looked at the SQLExecDirect: and other dll methods. He connected and looked at the SQL messages being sent and received.

He obtained the BNF grammar from the SQL-92 standard. When GemStone received the SQL, his PetitParser parser parses the grammar and generates Smalltalk. He ran a query and looked at the Smalltalk code the query generated.

Q(Niall) use under Glorp? Not yet; we will work together. Circular process ST -> Glorp -> SQL -> Parser -> Smalltalk.

Q. Performance? No checks run yet. James thought it was probably not too fast. His motive is more about making it possible and if eventually users start saying make it perform better then he wants to reply, “Sure; just let me write it in Smalltalk”.

DBXTalk and Garage for RDBMS, Esteban Lorenzano
Esteban wrote DBXTalk as his first project when joining the Pharo community). He has since written Voyager and also become the Pharo fireman. Pharo's strategy for RDBMS connectivity is to use Glorp, Database Drivers and other tools as people desire and provide in other projects. GLORP is the ORM for Smalltalk so no need to do another. It's

mature, modern and tested. “Glorp is amazing” (a certain Niall Ross said that at a past ESUG) but Pharo's GLORP port is less amazing. It is old, noone knows what it was branched from - maybe Glorp 4 circa 2006. It is compatible with PostgresV2, NBSqlite3 and OpenDBX. A newer port will be done.

There are several native drivers for non-commercial databases for Pharo: PosgresV2, PostgresV3, Sqlite3, MySQL. They are good but their APIs are incompatible so it is hard to plug them into Glorp. They use FFI so their calls freeze the VM - usually unnoticed but some queries take a long time. And of course, FFI is yet another layer. Documentation is absent or hard to locate.

Esteban saw all the above and began work on a project: Garage. The aim is a common API (e.g. JDBC) that Glorp can use. They have drivers for Sqlite3, PostgresV2 and MySQL - and OpenDBX. There is a Jenkins service to test the work. It is all on 32 bits at the moment.

For next year, they will do the new Glorp port. They will have a threaded VM - FFI calls no longer freeze image. They will improve drivers, document, etc.

Q. Seems better to refurbish Glorp than do new? No, he wants to go to the latest Glorp.

Querier – simple relational database access, Michal Balda
Querier is a library for querying relational databases in Pharo. It is inspired by NotORM. Today, accessing relational databases in Pharo is only doable by either writing SQL in full by hand or else use Glorp. Querier falls between these two: it directly exposes tables and rows and only needs to be told the primary key, the foreign keys and the table name. For example, data from a table 'song' can be read by

```
db song do: [:row | Transript show: row title; cr].
```

generates

```
SELECT * FROM song
```

It pretends a table is just a collection of rows and a row is a collection of values. A fake object is passed to the block and it gets executed and creates the query.

```
(db song sorted: [:a :b | a length < b length])
  allButFirst: 20
```

or

```
db song at: 123
```

will select the row in table song whose primary key is 123, or you can write the block ... row id = 123]. You can use collect: and select:.

```
db song select:
  [:row | row album name = 'Unknown Album'].
```

gives

```
SELECT * FROM song LEFTJOIN album
ON song.album_id = album.id ...
```

If you do

```
db song do: [:row | transcript show: row album name]
```

it generates two queries, one for all albums and then one for all songs using the result of the query for all albums. His update will select rows before updating them unless it is a mass update (which can have conditions). Writing is via

```
| row |
row := db dong row new.
row ... set values.
```

Table querying uses a collection-like interface, queries are built transparently and executed lazily. Simple effective access to related tables is one of the goals of Querier.

Q. Garage? Garage sends the queries. Querier is about constructing them.

Q(Niall) Look at Glorp's MessageArchiver; it uses the same fake-block technique and has code to solve some of the side-issues that arise.

The Glamorous Toolkit for Pharo, Andrei Chis, University of Bern, Mould your development environment. Your system is too special to leave it in the hands of generic tools. The Glamorous Toolkit (GT) brings a new generation of easily customisable development tools. He showed a hierarchic diagram (circles within circles) from Roassal showing classes in the image and that there were 168 extensions to the GTInspector. Most are simple; some, for complex domain objects, are complex extensions. GTSpotter has 101 extensions - maybe it's easier to find things than to show them.

He opened a playground with some code; putting some objects in a dictionary which they then inspect. It starts with a key-col value-col and strings for the entries. Select a row and get a 3rd RHS pane showing the more detailed view of the value, with printString view below. Tabs along the top of the third pane let you select various views for the various types. He selected an RTView and saw its typical Roassal presentation. All Roassal classes have examples so select a Roassal class and RHS pane has tab where you can see all those examples minimised.

Select slot and see what methods use slot (standard RB view). Markus loves the views of a CompiledMethod (see his talk). He selected a png file and saw its picture in a tab. He selected a zip archive and saw a list of contents. A tab can expand right, TrailBlazer-style, to another pane with tabs. He walked from a script to its pictures and classes and so on.

Q Is the script file now changed? I have not saved the script. (He tried to but got walkback, later got another hiccough: he realised he had added extra keys to dict, then tabbed back to earlier view of it and saved.)

He looked at a performance profile, then at Roassal graphs of statistics on the profile.

How to create extensions. Inspect the GTInspector class. Select an extension method.

Alex took over and presented the search tool GTSpotter in the same way. The search tool's LHS pane lists SystemBrowser, Playground, TestRunner, Spotter, MonticelloBrowser and other areas where one could search. He found four packages with GTSpotter in the name, selected one and opened a browser on it. He found all classes with UITheme in the name. He found a method, thence all implementors and all senders. He searched a directory structure, dived in and searched only for zip files and searched inside them. He found an XML file and looked at the object the XML defined.

Q(Christian) it regards the schema of the XML? Yes.

You can tab back (explicitly using the list of tabs added at the top as you search step by step.

Type regular expression to get the regular expression framework. He found extensions (and showed there are 104 - three have been added since his colleague looked).

The spotter is not about search algorithms but about adding additional functional searches.

He then showed a playground based on snippets where some snippets are text, some are code and some are XML (he showed extracting values from the XML) using Pillar.

The Glamorous Toolkit for Pharo - hands-on Tutorial
This was a well-designed tutorial that I found fun to do.

Event Touch / Gestures for Pharo, Merwan Ouddane
Merwan has implemented gestures in Pharo for touch devices. In Pharo, the OSWindow is a platform-specific window manager. He has been working in this. The SDL library allows low-level access to Windows events. Finger Trackers capture the events relevant to him. Classes OSTouchScrollEvent and similar capture these events in Pharo. He used it in 3d modelling with Woden. The code will be on CI soon.

PetitParser, Jan Kurs
(I missed this and only quote the abstract.) PetitParser is PEG-based parser combinator framework utilizing scannerless parsing and packrat parsing. PetitParser makes it easy to define parsers with Smalltalk code and to dynamically reuse, compose, transform and extend grammars. Recently, PetitParser was extended in two areas:

- parsing context sensitive and indentation sensitive grammars; and

- parsing imprecisely or incompletely specified grammars (island parsing).

His tutorial showed how to develop an island parser of an indentation sensitive language using the PetitParser framework.

Bringing the Concept of Quality into Pharo, Yuriy Tymchuk and Michele Lanza
Run his inspector on the packages you want to inspect. The result is a code city. A city block is a package, a building is a class and the blocks in the building are its methods. Filtering shows and hides aspects of interest.

Quality Assistant helps you distinguish between good and bad code cities. It consists of the Nautilus, the Inspector and the Spotter. In Nautilus there is a pane at the foot of the method text warning of stuff. (Like our annotations in VisualWorks.) You can mark theirs as false-positive for that method or for all methods of class. When he activated this, people starting complaining that some of the rules were bad. There is therefore a button which sends a message "this rule is bad" with the message (and dialog to confirm). QA is just the UI part of existing tools. "In VW recently they have this amazing way of inlining the warnings." A new tool Renraku will do something similar.

Q(Christian) Warnings yes but in your own code and when you want; able to turn off for not-own-code and for not-now.

Q. False positives? No real ideas but he wants to start by being informed of useless warnings. (Niall) suggest button generates workspace of method code, the rule and a place for the user to type why the warning was useless in this case, plus an email to which the text can be sent.

Q. Criticism is negative; can we do it more positively? General response was that these warnings are good, subject to Christian's caveats.

Towards Well Planned Code Cities, Natalia Tymchuk
A code city is a good way of visualising software. Natalia's plan is to add a distance scale. Real cities are like Barcelona - rectangular grid - or like Paris - areas of no rectangular grid. In Lviv, it was easier to go around the hill in the city and the city layout shows that. She uses Voronyj Diagrams (areas and a central dot in each area). She imposed a Voronyj diagram on Santiago and show that one area was of all points closer to the train station (one of the dots) than to other centres. She has implemented better algorithms: these diagrams were taking 4 minutes for 250 nodes and can now handle an order-of-magnitude more nodes. She had also investigated multi-dimensional scaling: she showed a result for SUnit that suggested it was not proving a useful technique. She compared a force-based technique: on two versions of SUnit, this had a cell for a 1-method class that was bigger than a more central cell for a class with 60 methods.

As well as exploiting distance and position, she would like to assign different features to differing software entities. Do we see force resulting from springs between nodes or from forces on the (connected) nodes.

Nodes should have a repulsion force (to make the diagram occupy the available space) and a repulsion radius within which the force acts. Edges have an assigned length. Traditional distance ignores whether the nodes are of different sizes - it just puts the same length within and without each node. A padded (shape-aware) distance would allow for each node's size. There is a time to allow the diagram to relax into a desired state and time can be slowed as that is approached. She showed examples RTFBLayout.

Q. Results? She showed a case of closeness of connections. Work continues.

Pillar: One Format to Rule them All, Cyril Ferlicot-Delbecque and Damien Cassou
Pillar is a markup syntax and associated tools to write and generate documentation and books. Pillar's features include:

- simple markup syntax with references, tables, pictures, captions...;
- export to HTML, LaTeX, Markdown and Pillar itself;
- customisation of the export through a dedicated STON configuration file;
- support of templates using the Mustache templating engine;
- syntax-highlighting of generated code blocks;
- configurable numbering of section titles and figures;

Ecstatic is a static website generation, built by Stephane Ducasse and Guillermo Polito, that uses Pillar; examples are http://guillep.github.io/ecstatic/ and http://guillep.github.io/DBXTalk/. There is a documentation renderer by Kasper Osterbye.

The speaker showed an example of writing a Pillar script, created a document first.pillar. Using wiki-like syntax, he wrote some text (headers, lists, tables). He executed a command line

```
./pillar -export ...
```

and (had the usual demo hiccough so did not) generated the HTML page. You can create your own doc templates (in files). You can create visitors for e.g. exporting your document in other formats. They have general tests for the visitor pattern; you can adapt these easily for your exporter (just overwrite some things). He also showed adding an annotation.

Bloc: a Modern Core for Highly Dynamic Graphics, Stephane Ducasse
Bloc is the new core for Pharo graphics. Bloc is a complete rewrite from scratch of core classes and logic of Morphic. Bloc has been developed by Alain Plantec for a couple of years; it is the result of several iterations and based on the experience accumulated by Alain while designing Miro and Miro2. Bloc is the basis for Brick the new widget set developed by the University of Bern.

A Bloc has a view (a morph) and behaviour. The morph can change dynamically. Stephane demoed creating blocs, dragging them around, etc., scaling and rotating of inner elements with the overall bloc, etc. Some blocs dynamically wrote large text explaining what they were doing in their inspector panes. There are three hierarchies, the model-layer one that defines behaviour, the one that defines the bloc's morph and the one that defines the view. Slides (later in talk) showed the Bloc's internal structure. You can attach cursors to morphs within morphs, and handle multiple cursors. You can add event listeners. Slides showed code examples. he then showed the Brick architecture that sits on Bloc (and other things). He demoed the variable-element-size list example.

Slots, Markus Denker
He discussed adding slots to class Point.

```
(Point slotNamed: #x) read: 3@4.
(Point slotNamed: #x) write: 7@8 to: (3@4).
```

What's the use of this? With the MetaLink system in Pharo's AST, you can inspect

```
Point assignmentNodes
```

and see and modify them but with slots you can directly annotate the instVars

```
(Point slotNamed: #x) link: myMetaLink
```

and among other things this supports breakpoints. The Opal compiler uses the AST which is the Refactoring Browser's parse output. By default, it provides reflective read and write to slots. Subclasses override their generics to arrange that standard bytecode is no slower than before.

Slots and Globals are now instances of those classes so we can modify behaviour. The current class template specified instvar names only; we need a new class template to let slots be provided.

```
  Object subclass: Point
    slots: { #x #y }"can be more than symbol if slot
non-standard"
    classVariables: {  }
    ...
```

In the next demo he had the usual demo bug - or he suspected it could be an actual bug in this in-development system. They have simple Slot, WeakSlot, Property slot (they are looking at array slots) and Boolean slot. The system still has bugs related to interaction (for example, dynamically removing a slot and adding it again further up the hierarchy needs debugging). In Pharo 4 they added Monticello support, i.e. you can save slots. Pharo 5 removes the old compiler AST and so Slots + Reflectivity works out of the box. However they will not make anything depend on Slots till more work and understanding has been achieved. In Pharo 6, they expect to have a useful library of Slots. Already there is a prototype of Bloc/Morphic using slots.

Thoughts: can we model bit patterns and bind them to named virtual slots? Can we model Array-like layouts better?

Q. (Graham McLeod) where can we find the best documentation? (Discussion: search for slot links on Pharo website.)

Genially) migration issue over time (some large systems using Gemstone have addressed this)? Uses reflective protocol to read from old object to write to new object (if possible).

International Smalltalk Workshop

The papers were

- A meta model supporting both hardware and smalltalk-based execution of FPGA circuits
- Code Transformation by Direct Transformation of ASTs
- DeltaImpact: Assessing semantic merge conflicts with Dependency Analysis
- Software metrics to predict the health of a project? - An assessment in a major IT company
- A First Analysis of String APIs: the Case of Pharo

Show us your project(s) in 10 minutes maximum

As last year, there were about as many in the Tuesday session as in the Thursday session; people now come to the conference aware of these sessions and thinking about what they can offer.

Smalltalk: alive and kicking

Stephane Ducasse gave a 5 minute talk on some smalltalk success stories. He showed slides of stuff powered by smalltalk. A container ship (contents being routed by Smalltalk). Chip manufacturing. JPM, LAM, AMD (GlobalFoundries), then some Pharo ones. Last slide of this set was a picture of a present. He is here to give you a gift - Smalltalk.

Then he used class Boolean to explain dispatch - if students don't get then he knows they don't understand dispatch. [NIALL: I suspect an advantage of teaching it this way is that when they do get it then they will be OK with ifTrue:/ifFalse:.]

Diego Lont, Merge Tool

He worked in VASmalltalk which uses Envy, a built-in versioning system whose basic code-grouping unit is the application (a number of defined classes with methods, and extended methods of classes defined in other applications). Applications live in Configuration Maps. Classes, Applications and Config Maps can have open, editable 'editions' and closed (frozen) versions. He showed two config maps containing different versions of the same app. He opened his tool and using drop-downs selected merging one map into the other. It showed him a list of the conflicts. He loaded them and (saw warning “You are creating a new edition”, and OKed it and) it worked.

Yuriy Tymchuk, Pharo and GitHub
It's not that clear how to use Git stuff. You create a class `BaselineOf<YourAppName>` and put it in a component of the same name and give it a method `baseline: aSpec`. Then you need the script

```
  Metacello new
    baseline: #projectname;
    repository:
'github://username/projectname/[:branch| ... |path';
```

You can then release code or load it. You can express baseline dependencies. Versions have to have three parts but that still means version 2113 is ambiguous, so call it e.g. version21_1_3. Use `#stable` and `#development` and map versions to one or other of these. He showed a graphic of his publishes of a project he worked on.

Christian Haider, Visualisation with AppeX
Christian moved to a new city (Lubeck) and engaged with local politics. Lubeck has a huge debt. It is hard to find out information about it. It gave Christian an excuse to do a new visualisation - the ones that existed did not help him understand it. So he tried VisualWorks 8 with AppeX. The city budget document is 900+ pages of figures divided into 137 'products': traffic, police, etc. (it's an EU standard!). His goal is to show revenue and expenses against the many sub-categories, balancing the data against each other. Christian managed to get the data in a machine-readable form. He then made visualisations: for example, a boxes within boxes hierarchy showing what is big what is small., with popups and highlighting to other displays. He showed getting information extracted from these, e.g. internal admin is 11% of the budget and central admin is 17%. He looked at IT management; a display had the income on its left and on its right the larger outgoing box. He looked at a few items for which there is a surplus. His page expands to show a time series of how it developed: a darker column for the current year of the plan, and the years before and after (ideally, that is; the bureaucrats who provide this information can be six years behind in collating it, apparently).

Christian went down the list expanding line items into years-before, current-year, and projected-year-after for those expenses. He managed to compress a lot of information into a small space and made it accessible to people. He now wants to divide by Lubeck's population so people can see how much their state spent per person e.g. on a theatre they never go to, and so on. He also has an Admin page for the site.

AppeX is cool!. He developed with three browsers open for Smalltalk code, Javascript and CSS (the CSS lives in the local directory). The process is very clean (he found two minor bugs and reported them to Jerry). You can see the site at schatzkarte-sh.de. The code is in the schatzkartenproject package in the Cincom OR.

Niall Ross, Jan Vrany, Subclassing TestSuite and TestResult
My talk was to make people aware of SUnit Open-Source project work on subclassing TestSuite and TestResult. In VisualWorks 8.1, I loaded RBSUnitShowResult, which prereqs SUnitResourcePatterns and also adds

a settings page to assign the TestSuite and TestResult default subclasses that any test runner will use. I showed that the current choices were

Test suite class: PluggableSuite

Test result class: KeepLatestResult

and showed how, with these choices, my example tests always passed, and marked up the RefactoringBrowser with their outcome (via icons against methods and classes). I then chose

Test suite class: RandomSuite

in the settings pane and showed that the tests were being run randomly (because a particular slow test, that had previously caused the 'tests remaining' integer always to pause at position 2 from the end, was now making the pause occur at a random number). It was also the case that a test now sometimes failed (because I had rigged that test to fail if a given other test were run before it in this contrived example; because the tests unintentionally interfered with each other in the real-world example I was mimicing). I ticked the 'Inspect Failed Randomised Test Order' setting and reran and (had the usual demo hiccough: I'd forgotten to hit 'Apply' so when the run next failed it did not display an inspector, so I hit 'Apply' and reran and) after a while a run failed and I had an inspector on the failed test suite in the order run (a PluggableSuite, cloned with the failing order, not a RandomSuite, so it could be rerun in the fixed failing order). I invoked ‘Print It’ on

```
self run
```

in the test suite’s inspector, showing that running the tests in this order failed every time, i.e. this was indeed an order-dependent effect. I cleared results and reran to show that this running in the inspector caused markup of the RB, just as if I had run in the RB.

Summary: SUnitResourcePatterns contains example subclasses of TestSuite and TestResult. (It also has other stuff – including subclasses of TestResource, as the package’s name suggests, and a test skip pattern.) RBSUnitShowResult is an add-on that marks up SUnit test methods and classes with the result of their last run. It captures test results however run (in VW, the runner is usually the RBSUnitExtensions utility) and displays them in the RB. Because the results are weakly keyed by CompiledMethod, editing or unloading flushes the result (in the test case and also in any inheriting subclasses). RBSUnitShowResult also adds a settings page from which you can configure which test result subclass and/or test suite subclass to use. (The UI aspects of these implementations are specific to VisualWorks and ObjectStudio.) The technique of subclassing TestCase and TestResult, and these specific examples of it, should be generally useful.

After the talk, Jan Vrany and I added another resource subclass, to deal with the case where a resource has several states (e.g. compiler settings) and you want to run tests sequentially through all states, only one applying for each run.

Visualisation with Roassal in GemStone
The speaker showed applying the usual Roassal visualisations to classes in GemStone. He has written various tools: a user browser; a pointer detective; a class hierarchy browser.

Milton Manami, Roassal and HTML
He ran a video in which Roassal graphs were exported to web and displayed in the browser, complete with their user interactions (zooming, dragging, etc.).

Noury Bourakadi, Robots with Pharo, http://car.mines-douai.fr
(They are looking for a post-doc for Autumn and an assistant professor - a permanent position - next year.) They want to use robots in a fire to locate people who need to be rescued. He showed a video of robots in his lab (not on fire :-)) collaborating to explore the area without overlapping. Then he showed a simulation of robots exploring a maze.

Max Mattone, Safe Updates in Pharo
He updated a running web app to remove an instvar and showed that a request running when he made the change had a different outcome (and could have failed). He showed a deferred update that waits till no instances of classes affected by the update are on the stack, and noted that was not a complete solution - for example, no such time might occur in a reasonable window.

Jan Kurs and Jan Vrany, PetitCompiler
People have found PetitParser too slow. Jan Kurs, helped by advice from Jan Vrany, has compiled PetitParser. He avoids needless object creation by analysing the grammar and optimising. he compared the term rules in the original grammar and the inlined sequence in the generated grammar. The technique works for this grammar and for the Java grammar and should work for any PetitParser grammar.

Q(Andres) did you consider just making an object, not a compiled grammar? It seemed easier for them to have something they could save.

Q(Stephane) have you any feedback to people on how to write faster grammars? If people write LL(1) grammar, that is better.

CORMAS
CORMAS is a multi-agent simulator for ecological simulations. He started with a model of plants that grow and are eaten. Foragers burn energy to move from plant to plant and eat either 50% or 99% of the plants, and can reproduce themselves. He ported the software from VisualWorks 7.6 to Pharo and from HotDraw to Roassal. They improved it a bit (they now have 20 unit tests; they had none).

Jan Kurs, PetitMarkdown
He used PetitParser to parse text into HTML (which he displayed in a browser).

Laura Risane, MetaBorg, a framework for Board Games
(Talk was presented by Stephane Ducasse) Games where the board is a tiled space and pieces move from tile to tile on it are PackMan, Tetris, etc. MetaBorg lets you implement such games. He demoed.

Frank Warlouzet
Formatting has 35 parameter rules by default. His tool can tell which methods do not conform and possibly reformat them.

Thomas Eliot
He demoed Griotte code review.

An interface to OpenVX from Pharo
VX is a standard for accelerated computer vision. It specifies a set of optimised functions that hardware vendors will implement on various platforms. Vendors include AMD and loads of others. VX users express their work as a graph of nodes and kernels. The speaker converted a colour image to greyscale, allocated resources in Smalltalk code end converted. He then sharpened the image using more Smalltalk code. PharoVX is a prototype (expect bugs). He is a C++ programmer but found Smalltalk a great language for programming computer vision. He achieved in a week what he could not have done in a year in C-like languages.

Enterprise Pharo: A Web Perspective
The book is not yet published but available on the web. It has chapters on the many Pharo frameworks used: Fuel, Voyager, etc. See http://books.pharo.org.

Damien Cassou, Marco Valente, Another Approach to Initialization
Which do we prefer of

- initialize when creating
- lazy initialize when instVar needed
- class-side initialization

He proposed use of ini-modules where a basic direct initialization of the instvars is called via pragma by methods providing values from which the direct values could be computed.

Q(Niall) Have you reviewed Bobby Woolf's old Smalltalk Report paper on instvar classification? Not yet. (I emailed it to Damien after the meeting.)

Noury Bourakadi, More Robots
Robots will rule the world - so let's us rule them through Smalltalk! He demoed videos of what his students have done in their projects. We saw robots (almost) writing on paper, taking a gift from one person to another, following a tracked object, etc.

Norbert Hartl, SS7
You use Signalling System 7 when you make a phone call. He showed the telecoms stack for SS7. He has implemented an ASN.1 grammar parser to parse the definition. ASN.1 has 384 parsing rules; they implemented all of them (except xml and constraints). You have session handling: a state machine for the session and an RPC (ASN.1). The Home Location Register holds mobile customers and allows them to login to GSM networks. The authentication server holds a secret key for a customer and etc.

They've been working on this for 4 years. They have installed a Pharo3-with-Garage-database-driver application for storing customer data in Iceland. The work continues.

Boris Shingarov, The Mathematics of Computing
Create something wholly unaware of what machine it is running on, that gets its idea of what registers it has, etc., from a formal definition. A semantic operator acts on an instruction to produce an output. The task is to invert this. You can solve this by ad-hoc hackery or by finding a mathematical abstraction that describes all the cases. A paper by Knuth and Bendix looks at this. He deduces some properties of the system. He would have liked to prove that for a finite group, division must be unique but is not necessary - but he could not. A later paper showed you need to avoid symmetry in your transforms (otherwise you could loop making no progress). Summary: the goal of computer programming (as of scientific discovery) is not to proliferate stuff for every possible case but to filter the large plurality of cases down to a small and understandable core theory. Maxwell's equations describe the immense variety of electromagnetic phenomena in four lines. Similarly, a good program covers a very large number of cases while containing only very few lines of code.

GC and Cuis, Andres Valloud.
If you want to find out why something was not GCed, an inspector can show references. Another version of Cuis can say “show all pointers”. He showed how deeply nested such searches could be. He wrote a find root method that was bottom up and faster. He then ran a 2 minute process that used an outer array and an inner array of inner plus outer that looks for things that can be reached from inner but not outer.

Other Discussions

Leandro announced the Buenos Aires 2015 FAST conference November.

Conclusions

Judging by the levels in the bottles by the time the conference ended, people enjoyed Andreas’ and my whiskies as much as we enjoyed the conference. Someone quipped that, “Next year, there will have to be an Alcoholics Anonymous stand.” :-)

UK Smalltalk User Group

This notes the talk and post-talk discussion of the January UK STUG.

Zombie-Kernel, Davide Della Casa
His work is similar to LivelyKernel and to Cuis.js. He is talking about a third such project, Morphic.js, done by Jens Moenig (worked with Dan Ingalls and others). The idea is less specifically about Smalltalk than about a good dynamic language that works on Javascript. The speaker took this project, which was like Scratch, as his start point: it looks like a toy language but it is not a toy language. It is not nearly as complex as LivelyKernel, and unlike LK is not on the bleeding edge all the time. It achieves its goal of simplicity. He therefore ported it to CoffeeScript (looks like Python, is a dynamic reflective language, works in browser).

He opened a web browser displaying some instances. He dragged 'Hello, World' about on the screen and inspected the object. The same menus give you some layout. (Taken from Cuis, because Cuis is very terse, unlike Squeak). He edited it (had usual demo hiccough) and updated it on the screen to 'Hello, beautiful world'.

“Last time I ran the tests they did not work but this time they should.” A test can be clever in telling where a menu item appeared, where it was clicked, knowing which menu item you are exercising or similar. He ran and things appeared and disappeared on screen as the test artefacts were exercised. He showed an image diff - two pictures side-by-side with red indicating where the one actually shown by the test differed in appearance from the one the test expected.

There is no image in the Smalltalk sense yet but a first step to achieving that is to have a good duplication mechanism. He showed duplicating objects on screen and moving the separate items on screen. He demoed attaching the slider to other things that moved and demoed; it moved well but he stressed this was not physics (i.e. in the ElastoLab sense or similar) but just ability to script behaviour. He also showed the test natively: a folder of files with the mouse events, screenshot png's, etc.

He created a morph with control, then moved it to overlap the morph he wanted it to control (necessary at the moment) and then made it the controller of the overlapped object (after which it could be moved away from overlapping it again). Keyboard shortcuts let him say that a selected object is to be saved as a png for a test compare, etc. He can also save bounding boxes and the whole screen.

All this testing stuff is new, not in Morphic.js.

Q(Richard) are the tests just experimental or have they been useful to you to find regressions. Detailed answer suggested it was mostly experimental (e.g. to find what kind of test was robust, what was brittle) and for persistence and for programming by example.

Q(Tim) is this built on Squeak.js? No, it is a different implementation.

Q(Bruce) all in the DOM? No. More in raster and pixel-flipping, avoids having two layers to worry about - the DOM and the actual appearance.

Q(Jason) source of widgets? Most come from Morphic.js.

Q(Richard) Morphic.js is CoffeeScript? No it is Javascript and fantastic but written by a guy who is uninterested in being a Javascript expert so it is 10,000 lines of Javascript, etc. he had to grapple with this to shrink it to his much smaller and simpler CoffeeScript. He can now port widget by widget.

It would be nice if the Morphic.js painted morphs rather than direct rasters. Getting Morphic and the DOM working together means being expert in both.

As soon as he gets persistence working he can start writing documentation about the system in the system.

Q(Bruce) what will you persist to? At the moment, Javascript. He demoed creating a simple test: drawing a filled circle and seeing the test repeats a gets the circle. He saved the test and showed the .js file.

Q(Bruce) why did you start? He saw demos online of people playing with Morphic and was motivated. LivelyKernel was great but too brittle - do anything and you find you've broken it. The LivelyKernel people have lots of things figured out about morphic and controllers but the code is atrocious in his opinion.

Q(Richard) Dan Ingalls is at SAP, i.e. working with Morphic.js? Yes. The guy Dan was working with was an attorney (much laughter - Tim thought he had in fact got involved in some legal situation to do with it at some point :-)). Speaker did not know why Dan did not use the DOM but went to the raster - perhaps he was inventing a new language and thought he'd do that part too.

He then showed a custom language that he and one other had made that showed shapes in the browser where simple scripting could make complex shapes and rotate them and so on: colourful and impressive. That system is very DOM-based. He hoped he could make the two merge together in time. He would also like to make use of tablets (he owns 3 tablets but can find nothing meaningful to do with them at the moment).

Q(Bruce) what next? He'd like LivelyKernel-like transformations, plus persistence. Then he'd like to get the LivelyKernel demos working: the clock that goes backwards, the star, etc. He was inspired by Cuis. (Alan Kay praises Cuis.)

Squeak.js ported the whole virtual machine to Javascript (a bit slow - can run eToys bot that is very slow). Amber is another route. There are many translators to Javascript these days.

If you search for Zombie Kernel on the web you will find him.

There was speculation that LivelyKernel was complex because it is trying to deal with DOM's oddities as well as the canvas, whereas this system just went for the canvas. What was saved? He thought not writing an editor saved quite a bit of code. Also the tax of doing both slows easy experimentation work that keeps things simple

Davide is a product manager in his day job and is not using this there yet.

UKSTUG General Discussion
Suzanne discussed Cincom Smalltalk with Bruce and Magnus (who are JPM developers) and the rest. The Cincom Smalltalk Release VW8.0/OST8.6 was focussed on the UI. The VW8.1/OST8.7 release is focussed on stability (but Suzanne is also being supportive of "walk-away projects"). This prompted some CST and ST thoughts from the group.

Bruce Badger has worked on a Python system for the last two years. Python has gaps that Smalltalk could play into. The step they have not taken in Python is image-based development. The cycle is: start up, spin up everything, find problem, shut all down, edit source code. As against that, they boast of their minimal image. CST has parcelling technology so we could do that. Suzanne said do tell Arden (or Suzanne) things like that: shared phrases and shared concepts that work with people from other computing languages are useful to know, as is "how we do things as opposed to how you do things" info. Lots of languages add syntax to deal with the environment whereas in Smalltalk the language / syntax is sparse and these things can be addressed by the environment: GnuSmalltalk, St/X are toward the Python style, VisualWorks could be using a minimal 'start and parcel in' strategy but is not at the moment, and of course something like Gemstone is very different. (Actually, nothing is like Gemstone.)

Bruce mentioned Pandas, whose underlying libraries are C: Python does not get reified in a debugger (no Eliot-style: "compiler writing is the art of cheating without getting caught" is done). Pandas is a reimplementation of the Python library on top of C to do some things that have to be done fast. It's just macros on C and Smalltalk could plug into it too perhaps.

Tim Mackinnon works at Pottermore which does JKRowling's fan site, with 6 developers. The site was in .Net and C# and was very visual but difficult for anyone to use other than a mega-fan who wanted to collect ingredients and brew potions and cast spells and so finally get to the content. Tim is teaching the team to use cards to sort out tasks and how things can be improved - or maybe entirely rewritten; that option is not off the table.

Tim briefly demoed the new GTspotter search/inspector in Pharo. It uses a searchlight-style and etc. approach to give you autocomplete on structure, not just one words, so you end up being able to inspect e.g. files on disk. Pharo have embraced this new tool. The view shifted sideways (like a one-pane Blazer, if anyone recalls that tool) to dive into particular structure categories that the start pane had found. He showed it finding a file. He showed a workspace in which the executing code was shown with structure: executing printIt on selected 4 + 5 showed 7 but in a more structurally-aware display than just linear text.

Written by Niall Ross (nfr@bigwig.net).

* End of Document *