CS18 and ESUG 20, Ghent, August 25th - 31st, 2012

This document contains my report of the ESUG conference in Ghent, August 27th - 31st, 2012 (and the Camp Smalltalk during the weekend before it). As there were parallel tracks, I could not attend all talks.

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.

Venue

Ghent is second only to Bruge (amongst Belgian cities that I know) for preserving the beautiful old centre of the renaissance town. The streets that border its canals are very pleasant, with fine architecture in abundance. The Camp Smalltalk site (at yesplan.be) was reached through a street market on Saturday, which certainly gave an element of local flavour.

We had rain over the weekend. Adriaan and Phillippe told me I did not need to seek out a swimming pool for my preferred exercise: "Run in the morning with us and you'll feel like you're swimming", they assured me. In fact, the conference venue's many stairs were sufficient to keep us fit, helped by the hot weather during the first part of the week. By the end, it turned quite windy and cooler.

Summary of Projects and Talks

I give a brief Camp Smalltalk 18 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
- Smalltalk Past and Future

followed by the 10-minute talk track and Other Discussions. Talk slides are available at http://www.slideshare.net/esug/tag/esug2012.

Camp Smalltalk 18

Camp Smalltalk 18 ran for Saturday and Sunday before the conference, and during the conference breaks, afternoons and some evenings of the five conference days. There were 35 people there on Saturday and 50 on Sunday. I worked on SUnit, developing patterns for subclassing TestSuite and TestResult. Others worked on Seaside, Spy and Roassal, Pharo, Seaside and many other projects.

ESUG Activities

Conference Welcome and ESUG Activities Overview, Stephane Ducasse, Johan Brichau

This year, there were 129 attenders (15 students, 10 free entrance) to see 30 talks, 2 workshops and 2 tutorials (on metacello and petit parser). Stephane called for the usual show of hands and we saw there were many new Smalltalkers, and only two people in the audience who also attended the ESUG in Ghent in 1999 (and rather more who were at ESUG in Brussels in 2005).

ESUG sponsors students. There are 15 at ESUG this year, which they feel is the right number for all students to have tasks to do supporting the conference.

ESUG sponsors Smalltalk user groups (Russian, Catalan), the Smalltalks conference, having Smalltalk activities at other conferences (e.g. the OpenSource conference at Paris: Python, Ruby, etc. - and Smalltalk), and Smalltalk articles (three in 2012). If you have an idea that would help Smalltalk and could use sponsorship - or advice or other assistance - email the ESUG board: the worst that can happen is that the board will say no. Ideas on how to promote Smalltalk are always wanted.

Stephane showed a slide of the Summer of code projects (see the 10 minute “Show your projects” talks for details).

They sponsor books, e.g. Andres Valloud’s book. (Stephane is co-author of some of these books but he does not do it for the fame. If fame matterered he’d write in Scala or Javascript. In Smalltalk, the ratio of the investment of time to the fame achieved is much higher.)

ESUG used to offer free seaside hosting. Now it will be done by smallharbour. The seaside hosting infrastructure is now available in smallharbour and you can put it on any server.

They get requests for lectures from Spain, Turkey and Africa (Cote d’Ivoire - contact Pascal Andre if you could give such a lecture). They offer 300 ready-to-use slides.

Innovation Awards, Jordi Delgado
There were 14 awards submissions on which we voted "en votre ame et conscience." At Wednesday's dinner, Jordi (speaking at a high balcony, from which he lowered the awards in a basket) announced the winners of this year's Innovation Awards. First he presented two special diplomas:

- **For promoting Smalltalk** to James Robertson.
- **For decades of dedication to Smalltalk** to Georg Heeg. (As it happened, Georg's family were there to see the honour in which the community holds him, which pleased them and us.)

then the main awards:

- **Object-Centred Debugging** (Jorge Ressia, Oscar Neierstrasz et al) came first.
- **Hazelnut** (Guillermo Pollito, Benjamin van Rysegen) came second.
- **Roassal** (Vanessa Pena, Alexandre Bergel et al) came third.

At http://www.esug.org/Conferences/2012/InnovationTechnologyAwards there is more information.

Conference Wrap-up, Stephane Ducasse and Luc Fabresse
Stephane thanked Johan, the local organiser. Johan can attend next year as a free man. (Niall: "It's a great feeling, Johan!")

Georg then spoke about STIC, the Smalltalk Industry Conference (formerly Smalltalk Solutions - they stole the acronym idea from ESUG).

STIC will be in June 2013. There will be a scientific workshop (please write papers). Fabian was their first student volunteer last year - Georg thanked ESUG for sponsoring him. On 24th April 1983, the first ever commercial Smalltalk system went live. Evelyn van Warden who was involved is now involved in contacting original Smalltalkers. Hopefully Adele Goldberg will be there.

(Stephane: maybe synchronise post-conference publication of the papers to look larger and achieve more impact.)

Next year, Stephane wants a "Show us your business session", like show us your projects. Maybe 10 minutes is too long for this. It could be 7, or limited in other ways: e.g. at a Pharo meeting, each talk had just 3 slides.

Have an Idioms and Patterns session (like Kothen and Brussels Design session but this time offering solutions). Again this would be 10 minutes max to present solutions. Aim is to share our culture at the code level. He liked it that Leandro's talk was not technically specific to his use case but general and pattern-oriented.

Stephane then invited feedback. It was noted that a steep stage, with side stairs, makes it harder to pass the mike between questioners than a level room. The schedule this year was at first only visible on the web as a google layout. It should be more visible, ideally on a printout in the hand-outs and

also on A3 pages posted outside rooms, etc. This was agreed.

Christian felt that one track was better than parallel sessions. I noted that occasionally a session is of specialised interest, or a repeat of a STIC talk, so sure to be uninteresting to some people who would be willing to see a parallel talk if it would not otherwise have room. (Afterwards, not in the main discussion) James Foster expanded on this: people always want a single track when the idea is presented to them in the abstract. However when a concrete case is presented, they would often rather have a choice of talks than not have the parallel talk in the programme at all.

Someone felt that while the 14 competitors for awards were great, with such numbers it was hard to see all - you arrive at a station and the presented was already half-way through demoing to someone else. Video clips are always asked for and frequently late or missing and to demand them would be to exclude some entrants. For every entrant to have a poster, displayed on/by their table, had been done in the past and is recommended.

Applications and Experience Reports

Continuous Integration - a Practical Approach, Maikel Vandorpe and Elke Matthis, MediaGenix

Their product supports the back-office of anyone broadcasting and scheduling content (e.g. a TV station). The product has existed for 20 years. In the 90s they were told to rewrite it in Java, did so, went broke over two years, then restarted the company using the Smalltalk product again. Now they have a team of 30 developers working on their 1.9 million lines of code in 15,000 classes (and that's not counting our VW base). This team grew to 34 during the conference as they completed 4 Smalltalk hires. MediaGenix employs almost 100 people overall.

Suppose a TV company has a cool idea for programmes to broadcast on Halloween. Much has to happen between the idea and the actual broadcast. They must allocate money, choose a distributer(s) and allocate rights to transmit. They must provide media of the programme and add it to a file server. They then must manage a range of tasks - edit, convert format, gut some parts, change credits, whatever – to get it ready to show on their channel. Four weeks before broadcasting, TV guide info is needed. They must assess, based on ratings, when is best in the day to show for the target audience. Lastly, before airing, they double-check all possible issues: right media, no more than X minutes of commercials per hour, etc. Afterwards, they gather statistics, write reports for the government and other companies, etc.

The MediaGenix system supports them doing that. MediaGenix have several customers in North America - Fox being the best known - and are starting to get customers in Australia and Singapore, but most of their customers are in Europe where they started: 35 all told. All their customers want everything as fast as possible.

Elke then took over from Maikel. She started to talk about continuous integration. CI splits release into small pieces, each containing a small

change to integrate and test. A new version is automatically built and tested each time you integrate, so it is much easier to detect which change caused the problem. On average, they have 70 changes each day. The integration process is easier because each one is smaller.

They run unit tests, of course. They also have 3 other kinds of tests. It would take 12 hours to run all tests sequentially so they parallelise.

Consistency checks ensure that code adheres to company guidelines. For example, their product has many windows and the common features are in TopApplication. Their default menu has 5 menu items, and more are always added between Tools and View. `buildMenuBar` enforces this and a consistency check on all subclasses of top application enforces that no-one overrides this in a subclass of TopApplication.

A method is part of a whole chain of what a user does. Acceptance tests imitate what a user does. They have many more acceptance tests than unit tests. They want never to change the functionality except when a user has asked for it (or they intend it). Her example showed the use case of a customer correcting Angelina Jolie to Bruce Willis in a person object assignment. Acceptance tests are implemented as SUnit tests, like others.

Finally they have refactoring tests. A basic button “What is on?” may have a basic function but also a different function provided because a customer asked for it. Thus `methodX`, called by basic button `WhatsOn` is no longer called. Refactoring tests detect that there are no more implementors or senders, or similar such cases.

Maikel resumed. The test environment wants to get the results as fast as possible, and to know when a build is broken (i.e. can’t load). A priority 1 bug - cannot transmit on air - must be fixed in 1 day.

They created their own system which runs multiple tests on the same database. Each test assumes the database is clean (no data from other tests). When a test starts running, it logs all its changes, and these are rolled back out of the database at the end of test.

Elke took over again. Roles: there is a test master, an image creator and 4 test coordinators each with 16 test slaves. (They have suitable icons ☺).

A developer releases code changes. A dialog appears ‘Schedule tests?’; they say yes and commit to Store with preference. The test master can see a scheduled test in Store. The test master loads, handles/delegates, then sends results. The test master has queues: to be loaded, loaded now to be tested. The image creator loads for the test master. When all is loaded, the image creator tells the test master, which then puts the created image in its second queue. When at the top of this queue, a test coordinator allocates each test class to one of its available slaves. The slave runs all tests in the class then reports it is done and ready to get another test class.

Maikel then opened a video of a test slave running UI tests. Many windows

opened and had values put into fields and reacted and so on.

The image creator can load up to 4 versions simultaneously, matching the 4 test coordinators available to run each version. Each coordinator has 16 slaves. Users can not run irrelevant tests, and the queue can recognise when a version is superceded by one later in the queue (earlier one is run overnight). Priority bug fix tests go to the front of the queue. There is a time-out of 30 minutes for loading, 75 minutes for coordinating and similarly for slaves (a test with a `self halt` left in it can run forever). Test classes are assigned in a smart order, to avoid situations in which all earlier-started slaves finish while the last slave gets the slowest test.

If a build fails, its developer is told within 45 minutes. If it loads and tests, they get an email about how many tests passed, failed and errored. Priority cases get results within an hour. Others typically wait less than four hours. A UI shows which versions are queued, which being tested, with estimated when-done times.

Maikel then demoed some tools. They save test results to Store and the tools rely on this. He opened a Store browser which shows tests' status (via abbreviations) in version display strings. He opened their SUnit-based tool on the failing tests and reran them to see the causes of failure. He can also locate from Store the version in which the test first failed.

The important goal is to emulate actual users and to meet time targets: Priority bugs must be fixed in a day (test results come within an hour), and others must be fixed quickly (test results come within 4 hours).

Still to do: test every single version, so as always to have a state.

Q(Niall) More acceptance SUnit tests as against developer SUnit tests? They found they were rewriting unit tests a lot because of all the refactoring they did. They found more value in user acceptance tests because that helps them ensure that what they deliver to a customer is what was promised.

Q(Georg) Not testing every version (i.e. jumping over older versions) is dangerous because an error is always where you don't expect it? Chris proposed testing of just changed code using just tests that gave coverage of that code, to allow testing of everything in a possible time.

Q(Christian) Customer has connection to Store? No, we load using the same process as we use when building to give to the customer, so we're testing what customer would get if that step was the last step in a release.

Q. The developer controls the process but surely even if a developer does not choose to test, you should still be testing every version? The system is busy 24x7 so some versions are skipped but most are done during night (or when everyone is at ESUG).

Q(Niall) Database rollback as against creating or having multiple clean databases? Each test slave has its own database. It rolls back between tests,

taking several milliseconds as against 15 minutes to create a fresh database.

Extending a Base Product for Multiple Customers, Denis Defreyne, MediaGenix

Suppose a Customer asks for feature X, but other customers don't want it. If your only option is to add it to the base then either one customer is unhappy or other customers complain of bloat. Customisations are customer-unique: they never share a customisation between two customers; each gets their own.

Customisations are needed because every customer has a different system and/or a different workflow. You must check that customisations do not breach any rules: no showing films you do not pay for, no films above parental ratings, etc. (The various rules are quite different between countries e.g,. between Finland and Belgium).

MediaGenix has some 30 large loyal customers: this strategy does not work for 000s of customers.

Customisations must never accidentally land up in the base product. They are fine-grained, changing specific method(s) or class(es). The main motives are:

- talking to other devices, such as integrating to external systems: playout servers, music systems, scheduling systems.
- catering to a varied market in Australia as against the US as against Europe
- resolving conflicting requirements
- a specific fast fix to keep a customer happy (but beware, do not write everything as a customisation!)

(His talk examples resemble actual customisations he has done for the three customers he mainly handles, differing only in omitting some detail.)

He showed a very simple film editor (the real one is a lot larger). If you want a widget for ID for MTV, you create a new class called a site class ('site' is a synoym for 'customer').

```
Film class>>siteClassForMTV
  ^FilmMTV

newFilm
  Film siteClass new
    title: ...

Site>>siteClassOf: aClass
  "returns appropriate site class, or base if absent"
```

There are customer packages, named for customers: FOX, MTV, etc. PackageB can override a method in PackageA. Thus the MTV package contains the `newFilm` override and the FilmMTV class.

Supermodel means (not Angelina Jolie, quick move on from that slide :-)) but a means whereby they plug in all this. The supermodel of the Film class gets the model-layer stuff for the new ID for MTV widget but must also override in the form class to make the widget appear.

```
Site>>siteClassOf: aClass
  addFormField: #idForMTV
```

Must you call `siteClass` everywhere? yes. However if you forget it you will likely notice (and if you don't notice and your tests pass, you are OK).

Important: avoid long methods. Short methods are easier to override to change the behaviour you want.

You cannot `siteClass` superclasses: it complicates the inheritance tree. In such cases, they move any methods they want to `siteClass` into a delegate, e.g. Product (subclass Music, Film, etc.) can have behaviour moved to a ProductBehaviour delegate, which can then be `siteClass`-ed.

```
ProductBehaviour>>isValidCertFor: aTx
  "can customise parental controls"
```

Lastly, they have convenience methods to make some common cases easier than creating a site class. By calling an `addCommandsToToolBar:` basic method, then `addCommandsToToolBarForMTV:` etc. in

```
SiteForMTV>>buildmenuFor: aToolBar
  self addCommandsToToolBar: aToolbar
  self addCommandsToToolBarForMTV: aToolBar
  ...
```

they can give specific menu items to specific customers.

So their techniques are to use (in this order for preference) site classes, behaviour delegates or convenience methods.

Finally, a Module is a pluggable component, which can be enabled and disabled, where some customers buy some modules. They have modules for ContractManagement, for SecondaryEvents (e.g. the logos that popup), for VideoOnDemand, etc.

If they want a `buildSuperModelWith:` in the Film class, and to extend the super model in a module,

```
self modules do: [:each | each buildSuperModelWith: ..]
... addFieldsTo: form for: self
```

(The null-object pattern ContractAbsentModule does nothing when code like this is called on it.) He demoed enabling the contract module, so causing the contract field to appear via this mechanism.

Modules can be shared between customers whereas site classes are meant to be customer specific. Modules require hooks whereas site classes can use overrides. (N.B. he means overriding a method in a subclass of its defining class, not overriding the code of a method in situ. They only use code overrides where they change the VisualWorks base).

Q. Testing customer-specific code? There is no special procedure, just the process spoken of in their other talk.

Q(Nick) There is much code in the database layer; do you ever customise that? MediaGenix have written an OR mapping framework. Denis does not do much with it.

Q For every site the tests are run? Yes. If he customises MTV then he puts code in the MTV package and it gets scheduled and all 30,000 base product tests are then run plus any specific to that site.

Q(Michael Prasse and Niall) if three customers want the same customisation? The Film class exists in different branches in several site packages. It's a vague, case by case, process to decide when and whether a customisation has become popular enough to be in a module or the base.

Building a Business with Cincom Smalltalk, Arden Thomas and Dirk Verleysen, Cincom Smalltalk

This talk demoed the whole process of building a business-support app, showing enough of each part of the process to make it clear how each was achieved. ObjectStudio is windows-centric. VisualWorks is cross platform. Both run on the same base.

Design and Modelling with the Modelling tool provides some code generation that, via bi-directional development, gives the model layer and, via the mapping tool, the object relational mapping and database creation.

The scenario is that Dirk and Arden are hired to solve a health and fitness club requirement. The club administrators need a software system to track members, dues, club attendance, guest attendance. They want to do promotions to their members. They want to send happy birthday emails. They want to send promotions to those who sign up as guests. They want to keep existing members happy and get new members. They want a web presence, with class schedules online. Classes are limited in size, so the club wants to allow reserving of a spot in a class online or via smart phone.

The club's employees also need features. They must login to software. Data must be backed up. There must be a backup plan if the membership card reader fails, or the whole computer fails.

Arden and Dirk start with a database of current members from an earlier failed attempt to build software.

ObjectStudio supports the usual methods (Rumbaugh, Wirfs-Brock, Coad-Yourdon, etc.) and free-form design. The process is to get the customer's inventory and their 30,000ft view with their terminology, so begin to understand their business. Thus they create the model and walk through with the client. The model must be iterated until it is OK. The client must often see a wrong model in order to discover the right model.

Actors (employee, member, guest, greeter, coordinator, instructor, trainer)

have use cases. An RFD reader will automatically check in members. Dirk and Arden asked, would it help you to also track when they checked out? Clubs usually don't, but when it was suggested, they said yes. As they are monitoring RFID tags, they could put them on the machines to see what is never used maybe it is broken but not noticed).

Example use case: if a member goes on holiday for a month, they can put their membership on hold, pushing out their end-date.

ObjectStudio's generator is white-box so it can easily be changed by customers with specific variant requirements. ObjectStudio has several tools (Use case, Explorer, CRC, ...) to model use cases.

The mapping tool can be used in three typical ways:

- do model, create domain classes, generate matching database tables
- do model, create domain classes, map existing tables to domain classes
- create domain classes from existing tables (unusual)

Our ObjectStudio system will need a client interface in the club, a web interface for members not at the club, and a way of connecting to the RFID interface (they use an ActiveX component).

Dirk started ObjectStudio and opened the modelling tool diagrammer. He created classes Person, Member, Employee, Trainer and CreditCard. He created a hierarchy (Person subclass Member and Employee, subclass Trainer, FrontDesk). He then defined attributes for the classes: names for Persons, expiration time data for CreditCards. He created a relationship between CreditCard and Member.

At this point there are no Smalltalk classes (Dirk showed there was no CreditCard class). He then generated the code, and showed there was code on the classes that maintained relationships. Dirk noticed the Person class wanted an age method, obtained from date of birth. The model now shows it has an age method. (This also works if you change code in the debugger or in workspaces.)

He then created a database and tables in the tool. He picked a class, checked the DB mapping of its instvars in the table, assigned to public schema in Glorp. Then he set primary keys, foreign key constraints. Again, he demoed that there were no tables in the database and then (after the usual demo hiccough - he forgot to map classes to tables and save to a descriptor system) pressed a button and they were created. He created the descriptor system ESUGFitnessDescriptorSystem in package ESUGFitness.

He ran a simple script to create two members, one with two credit cards, then used pgAdmin to show the data had been entered. Only the members were registered but the credit cards were in a foreign key relationship with the members so they were written too.

Next he created a UI widget by drag drop (widgets and their labels offered when he selects items in the model). He tweaked date format and layout of

fields, then generated the form for entering members. He showed the code created on classes to preserve data.

He showed the front-desk window next, with panes listing next exercise class and who is enrolled in it. They check people in and out so can see how many are in the club, for fire regulations and to know when closing up for the night. He created an AidaWeb / Swazoo site and he started it (doIt in ObjectStudio launcher) then opened a browser on it. He registered for a class and then (should have seen that he) was in the front desk UI list as registered for that class.

There is support for migration of data when the tables change.

Q(Christian) How does support for migration of data as the tables change work? You use old descriptor to read and new descriptor system to write.

Q(Leandro) Programmer works in modelling tool window? That is for the programmer to work with the business analyst, or the analyst on their own to do that.

Q. More than one developer? Currently only one model can be open at a time for the bidirectional modelling. They are looking at collaborative modelling. (Andreas) Store is in ObjectStudio, like all VisualWorks features, so one modeller can give their output to 15 developers. Those 15 developers can change code and then the one modeller can load successively the 15 changed developer versions and merge each into the single model. So many developers is OK, just not many concurrent modellers.

Testing Smalltalk AJAC/SJAX Web Applications with Selenium, Carsten Harle, straightec

Carsten is the founder of a company that provides teaching applications. He also consults for Datenzaentrale, who handle taxes (dog, kindergarten, waste, etc.). The application of the talk started as an ObjectStudio fat client and now also has a web client using AJAX and SJAX (Synchronous JAX). 90% even of the GUI code is reused between the fat and AJAX clients. (The latter uses the Prototype library which he may convert to JQuery.)

A web app must support many web browsers: IE from 6 through 8, Firefox 3, 4 and 5, etc. He had to be able to test on all browsers. His test framework had to be fully scriptable from Smalltalk and compatible with SUnit. It must be able to wait for the AJAX call to finish. (Usually you are using special message boxes and wait cursors, and the tester must wait for these.)

There is a large test team and he needs them to be able to write tests without much Smalltalk familiarity. They are in fact using Smalltalk syntax but they don't know it. Selenium has a UI recorder and Carsten wrote a Smalltalk API to drive it, plus custom code for special UI widgets. He put this together with his continuous test and build framework (see his 2006 ESUG talk: on publishing code, automatically tests start running on it and report failures). This API was of form

```
  selenium action: <Locator> {value: parameter}
```

e.g.

```
  selenium click: #buttonMyName ...
```

The problem is that raw selenium is often unreadable. His slide gave a good example of trying to click in the second line of a list. The commands were completely obscure on the slide (and wouldn't work anyway). His API has helper methods:

```
selenium click:
  (selenium listView: #ctvOverview row: 2)
```

became (so that his users did not have to get brackets right)

```
selenium clickListView: #ctvOverview row: 2.
```

Clicking in expandable widgets needs to work regardless of which levels of tree view are open, e.g.

```
selenium
  clickTreeView: #ctvOverview
  itemlabelPlus: #('Water' 'WasteWater').
```

works regardless of which levels of the tree view are open. It opens what it needs just as if a user was going to it. This makes the tests more robust.

Carsten made special accessors for tabs and suchlike widgets for testing. Everyone in the team knows these widgets and can talk in those terms. He can click buttons in a toolbar, click items in a tree view, etc.

Sometimes you get the wrong dialog box during your test. They've never actually had 'Format Hard Disk? appear instead of 'Open Order Book' but the code checks the dialog box has the right title before clicking 'yes'.

He then opened an image and a Firefox and had the tool record all his actions. He pasted the code into an SUnit test and ran it. It opened a new browser (you can have the tests reuse the browser, but it is more reliable to run always in a new browser, so one test's failure does not corrupt the next).

He showed a left-side menu that stays open at a given context after clicking, so the test needs to handle the state of the menu, find what is open and, as if a user, walk to the right menu-open state for the current test.

While some tests are created from recording as he showed above, others can be manually written. Usually, one generates, then transforms by hand. However often a single conceptual step is 5 or 6 clicks so you write it manually to express the intent rather than show the lower level click-by-

click recorded command. Carsten would like the recorder to reason up to that level but that is harder. The intent code is almost plain English and even non-programmers can write it.

AJAX request are asynchronous, so spurious no page load errors can occur in parallel with some delay in the expected return. For example, list views make asynchronous call backs and you must not click in a list view until its contents have been rendered. Just waiting 1 second is mostly slow and occasionally unreliable, so they have Javascript code to know when callbacks will happen and wait for them.

```
self addWaitCondition:
  self class waitConditionPrototypeAjaxRequests.
```

You can have modal dialog boxes which create their own wait cursor instead of the standard ones. Sometimes you can interact with elements in the browser even though there is a modal dialog open. Special Javascript detects presence of such dialogs and “click but dialog box in way so fail”. You can click in a standard message box by giving the element

```
selenium clickMessageBox: <element>
```

or an explicit location. Customers may disable some input fields but the tester may attempt to write to them, so must detect this and raise exception.

You have easy access to Smalltalk objects.

```
self assert:
  self activeController object bookName = ‘Moby Dick’
```

This lets you test much more powerfully by modifying the Smalltalk model-layer object as part of your test. Of course, an actual user cannot access, still less affect, the Smalltalk objects except through the web UI.

He demoed again, showing how the tool ran fast by not using a ‘wait one second’ style approach.

GUI tests are slow compared to other SUnit tests. They have 600 - 700 UI tests and it takes 8 hours to run them. They have a cloned virtual machine - 30 clones of it and they plan to double that - and these clones coordinate the test runs between themselves (see his 2006 talk). He showed the status report: green means passed in your image, blue means passed or failed in another image.

Q(Niall) Available? He will ask his customer if he can make his selenium add-ons open-source. (I also mentioned that Carsten had no demo hiccough - clearly the talk was well-tested too. :-))

Q(Martin)? Carsten explained that serialised JSON handles coms so it is easy to write language bindings.

Q. Visual result checked? No, they check values in widgets (or in Smalltalk code) obtained by operations. They can generate text print-outs of XML and compare to text.

How do I represent model scenarios, Leandro Caniglia
Their users (oil industry) build models using their software. They change things in the model and then want to compare scenarios to analyse different strategies. He showed a simple example where he has a reservoir located in deep water with several wells: appraisal wells and main wells; three locations, main with 10 wells, two auxiliary with 5 wells each. Software appraises cost of these activies against value of selling resulting oil.

Rigs are costly and are options: various wells can be drilled with various rigs. He scheduled some activities, using the two rigs to start drilling exploration wells, then appraisal wells, then building the three facilities. In this schedule, all three are ready at the same time, along with a pipeline. Then you drill the production wells. He then ran a simulation of this plan being done. Things are as expected except for the time which is longer than scheduled. The plan had the same rig drilling two facilities' wells.

He showed the reservoir estomated rate: exploiting the reservoir and then production ramping down by 2028. The graph shows lower and higher lines, the lower being the limit of the planned systems capacity, the higher being what the reservoir could offer for those wells.

He wants to explore the effects of giving the plan another rig, or expanding the capacity, without losing the original model. He made changes and showed them being logged (spreadsheet-like display). Smalltalk is an example of classes and method being executed to get results: classes and methods are specifications of objects that come to life when the messages are sent, leading to specific results. In their system, the model spec, the simulation of it, and the results obtained, parallel this.

Cloning a model spec is one way to track scenarios. It loses the connection between the original and the copy so you've lost synchronisation. As the base model is altered, only the alternative scenario changes should differ between it and a scenario based on it.

Smalltalk has a logging system where you can (re)execute every change. Using that as inspiration, they model the changes they make to a spec and make them (re)executable. Class UserCommand knows the receiver and selector of the change, and the user who made it, when they did, etc. It has subclasses ResourceChange/Creation/Deletion/Renaming/Duplication.

He opened a tool, created a scenario with another rig and showed the changes list. Every pane in the UI has a model object. They wrap the model object. The wrapper registers the message as a change and then sends that message to the model. The change set transforms simple objects using their change representation, and has ways to handle more complex objects.

Running the scenario shows that adding another rig alone loses money while expanding the capacity alone makes money. However doing both gives a better result than either on its own! This is a good example of how modelling scenarios gives value.

A single wrapper class is enough: it has wrappee and changeset as its only two instvars. All methods on the wrapper do the same thing with different arguments. These methods are generated when first called, just to replace DNU with a faster mechanism (asking the developer if in development state of system).

Lastly, their scenario modelling system can consider uncertainties. They can run a monte carlo simulation in their system. That run shows the add-rig strategy has a probability of being good in itself.

Q(David) Your example combines two scenarios; done automatically? Done easily.

Q(Carsten) Why do you automatically created wrapper methods (all mixed into one class)? It records the API of the application and you can sell that. All methods in the API can be seen for metrics, documentation, etc. Many applications use this protocol that is in one class.

Refactoring Support for Smalltalk Using Static Type Inference, Martin Unterholzner, Lifeware

Martin works at Lifeware, a Swiss company that provides lifetime support to insurance products. 35 people, almost all developers, now work at Lifeware. The company has existed since the 90s and has more than doubled in size over the past 5 or so years.

Smalltalk (and even more Lifeware) has many tools to support refactoring: making changes that preserve behaviour. He showed a trivial program in both Java and Smalltalk. Classes Bird and Cat both understand `move` and we rename to `fly`. This is an example of Java's strong type system. (Ignoring CASTs) we just rename `move()` to `fly()` in Java. In Smalltalk, a static method rename would rename `move` to `fly` for both Cat and Bird.

In his example, the polymorphism of move is not actually used. Absent a Java interface, we cannot use these methods polymorphically. In Smalltalk, we can use them polymorphically so the default refactorings assume you do in fact use them polymorphically.

We can enhance our refactorings if we use knowledge about actual polymorphism. To get this information, we use type inference and symbolic evaluation. You walk the parse tree, recognising literals, asignments, message sends and etc. He stepped through his simple program, showing the steps of type evaluation.

Q(Georg) you introspect `new`? Yes, Martin works down to primitives whose type effects are known.

This works in theory for renaming. He could extend it to other refactorings. He randomly chose methods in Lifeware's image and type-inferenced successfully for 25 - 40% of methods (the percentage differs between base code and for-customer code).

Finally, he demoed on his example program. The RB changes browser opened showing that `move` was renamed to `fly` was renamed in Bird and its callers, but not in Cat and its callers.

Q(Stephane) You have looked at RoelTyper (in the Cincom Open Repository)? (Niall) Also at RBDynamicMethodRenaming (in VW and the Cincom Open Repository)? (Alexandre) Use dynamic profiling with unit tests? There was an offline discussion – Martin was interested.

Q(David Chisnall) perhaps you could reach 80-90% with a very little hinting; have you looked at where hints are needed? Hints tend to be needed just where hinting is really hard to do.

Q(Stephane) Simplify by reducing scope? Yes, easy to do by hand-programming in VW. (It should work the same in all dialects to restrict the scope by restricting the base BrowserEnvironment of the refactoring).

He will make it visible and would be glad if others wanted to work on it too.

VMs and Development Environments

iOs: Smalltalk and ObjectiveC, Tansel Ersavas

To learn Smalltalk, other programmers must unlearn what they have learned. To learn ObjectiveC, Smalltalkers must relearn some of that.

Tansel asked if anyone was using ObjectiveC? - yes, several hands. Then he asked if anyone was loving using it? - no hands. For Smalltalkers, it's a step down, for others it is a step up. ObjectiveC separates interface and implementation. There are many UI-related macros.

He began demoing with examples of some Smalltalk in ObjectiveC such as [self.navigationController pushViewController:videoStarter animated:NO] and [UIView setAnimationDelay:wait:]. Then he started merging C and Smalltalk and commenting on some ObjectiveC quirks.

He showed an example of defining an audio device interface, colour-coded (purple for macros). IBOutlet and IBAction will be provided by the user of the interface. The - sign shows instance methods (class methods start with a +). Then he showed an implementation of that interface: synthesizing creates getters and setters for the button widgets.

ObjectiveC provides much of the same meta-protocol as Smalltalk, e.g.

```
+(BOOL) instancesRespondToSelector:aSelector
-(BOOL) isKindOfClass: aClass
```

and so on, but it does not have `become:`. ObjectiveC naming conventions are verbose and otherwise Smalltalk-like. A good ObjectiveC tutorial is at Cocoa Dev Central.

On an iPad, he showed a few days work to get a walk-around dungeon area. The ability to get this done in a few days is the reason he likes working in iOs.

Dowloading XCode from the apple site is free but needs registration and OS-X (not native Mac hardware, just OS; on Windows and Linux, you can use VMWare to run OSX). iOs uses XCode. Do not expect the Smalltalk IDE; XCode is a different environment. It is getting better daily.

He opened the many-windowed IDE. There's an output pane (Transcript-like) and a quick help pane. The interface builder is part of it, so you create your UI icons / widgets and connect them to behaviour. You must have paid for registration to run the resulting app on your phone. You will need custom artwork to provide the essential quality of look. (Or be a graphic designer and read the Apple UI guidelines.) Apple, and the web generally, provide many images.

The retina display is new. Apps should cater for it, or be scaled in and look poor. For retina, all artwork must be in twice the resolution targetted (just append @2x to those files).

Your iOs platforms are iPhone, iPad touch, iPad, or universal (runs on all). You choose the orientation: portrait, landscape or make it work in both. There are many apps but they can be categorised: 1/2 page apps, tab apps, etc. The iOs demo shows a level balance that is a good example of a simple one page app. The periodic table list is a good example of a tabbed app. Navigation-based apps create a tree-like structure, usually with a navigation bar at top. He showed his baby example.

OpenGL games can be less compliant to Apple guidelines - they are whatever they are.

If XCode is too fiddly for you, you can use tools. Many claim to be cross-platform but you need to check this on the web to see what their users say. HTML5 operates quite well in the Apple world. You can't make an Apple app just with this, but it will look like an app - and can be bundled in an app if you need to put it in the Apple store. There are also tools for 2D and 3D game creation. These tools let you build stuff impractical with just XCode.

He consulted with a company to scan a museum in Istanbul and make it a walk-through app. He could not have done this in XCode.

There is a web course on app development from the university of Stanford.

Q(Christian) Any second thoughts about working for the dark emperor? He is riding the wave, not working with them, zen-fashion.

Q. Ruby? He's worked in it for 3 years. Tansel dislikes the syntax but is otherwise OK with it. Many people don't give the mental investment to go to a new environment that will be more productive. The Alan Kay experiment in Japan, introducing Smalltalk to children will make it hard for C and Ruby to unlearn that. But too many people get corrupted in school and university before they have the chance to learn Smalltalk.

Q(Noury) using the web to embed into iPad to avoid the apple store and

XCode? Tansel dislikes the web, but has started to use javascript and HTML5. Alan Kay's 1997 OOPSLA talk is as relevant today as then.

Cincom Smalltalk, Arden Thomas
What do Cincom Smalltalk and the Tour de France have in common? Well one thing they don't is that Cincom Smalltalkers use no drug except Smalltalk. Smalltalk is addictive. The debugger - studing the code live, not doing an autopsy on a corpse - is one of Smalltalk's many addictive features.

Arden's talk was not a Tour de France but a Tour de Force around Cincom Smalltalk, using some Tour de France features as analogies. The race has riders from all over the world and so does Smalltalk. Each team has a manager, usually someone who has been in the race before or who has done bicycling before. Arden is the product manager. He's been developing in Smalltalk since the late 80s. Suzanne is the programme director for Smalltalk - her history starts with digitalk way back when.

The 'peleton' is the big group of riders who stay together in the race. If you are behind another rider you use less energy (air resistance). Each team often has a GD rider who is the one whom the team are helping to win. The domestique rider sacrifices themself to help the team - e.g. by getting in front and breaking the air (i.e. the air resistance). At the end of each day they hand out coloured jerseys: white (best young rider), green (best sprinter), polka dot (best hill climber), yellow (the current leader in the overall win points table).

The current latest releases are ObjectStudio 8.4 (8.4.1 will be released this autumn) and VisualWorks 7.9 (7.9.1 will be released this autumn). Using his racing analogy, Arden began awarding jerseys for past, present and upcoming features in these products.

The international community category: VW has always been cross-platform, and has introduced CLDR internationalisation and unicode VM. VisualWorks also keeps up-to-date with Seaside. Xtreams has been open-sourced for the community (and external projects have used it, e.g. Colin Putney's). Xtreams was rewritten four times to be clear and effective. In future, there will be better Font rendering, kerning, unicode 6.1, and rendering non-Latin. (Integrating Pango would have solved some things but the new approach does 80% of what Pango would do and it is all in Smalltalk.)

Best new feature is WSDL 2.0, the SOAP 1.2 revamp, and latest is external encryption. External encryption offers more choices, sometimes a gain in speed, and lets you adhere to using a specific library in cases where that is demanded. For 7.10, we anticipate IPv6 and a new TextEditor. (Their old TextEditor has survived from the old Xerox Parc days. This testifies to its being well designed but it is time for a re-creation. The new editor is enabling some good things.)

A sprint means an experiment. VisualWorks has Delays (he means the class

Delay - not that it's slow :-)). These worked fine in some cases, not in others, and now they've been reworked to function in all cases. The current sprint is Skins which we've now got in. Next release, the sprint will be the TextEditor.

Climbing awards are given for making progress in difficult areas. Making 64bit in Solaris and Linux equi-capable with our 32bit product was certainly that. We want to make Store faster and also to offer better configuration management. We will develop these new features by using them internally and then offer them to customers. We aim to make other improvements to Store. The font framework is being revamped.

The Domestique award: Polycephally is a framework that uses multi-core processors. People started using it and got benefts e.g. in running test suites. Polycephally II is now out; this lets you use remote computers, not just multi-cores - i.e. grid computing. There have also been database driver improvements - thankless but important work. The same could be said of the work that made Windows 64, Com 64, Oracle 64 and ODBC 64 part of the main release. GC improvements have been done - including much old C code retired to make a cleaner and leaner virtual machine. Future domestique awards will go to code completion, code highlighting and this needs better text editor parser work.

The overall winning feature was Polycephally / Polycephally II, is encryption in the latest release, and will be TextEditor in 7.10.

New features: the Project Launcher helps brand new users, and helps deal with restrictions in recent Windows operating systems on where images can live, as against where the installed product can live. If you could use the Project Launcher for more things, suggest it to Arden (or just do it and show us the code). Cairo graphics are now there. The latest best new feature is Windows 64bit. You can have 30Gig of memory in your image for a fast huge model. (Arden noted that Windows 7 64 is the first Windows 64 bit OS that is really practical.)

ObjectStudio lets business people and developers work together. It is the only Smalltalk with the Windows 7 logo (so it gets the international community award). It now runs on the same VM and underlying code base as VisualWorks. The mapping tool now uses the shared Glorp platform. In future, we look to use shared COM.

ObjectStudio's best new feature was the revamped model and mapping tools, is the mapping tool on Glorp, and in future will be the UI work (see Andreas' talk). Its best sprint was getting Vista certification and will be getting beyond Windows 7 logo to have Windows 7 certification.

In future, we will do R&D work. We want to keep our customers happy and to grow the community. We will use Store config management ourselves to know what is best. We will do (yet) more testing.

Our release process has changed. We do semi-automated weekly builds

running a huge battery of tests. Arden, before returning to Cincom, worked for a customer: a hedge fund. That customer had hundreds of applications and to move to a new release was a lot of work. Would a longer release be better? We tried it but no, it slowed the freeze and release. So we returned to annual releases. Customers asked for a true maintenance release, not just a dot release that is a smaller new release. Now our dot releases are true maintenance releases. Support and engineering now work very closely on these dot releases.

Sportsmanship is really important in the Tour de France. Cincom will support ESUG and STIC, we have open-sourced Xtreams, we will directly and indirectly (through partners and customers) employ Smalltalkers. We want Smalltalk to succeed.

Pharo, Stephane Ducasse
Pharo's aim is to let you become creative in Smalltalk, including making money with Smalltalk. The Pharo consortium's purpose is not to make money for itself. Even 5 years ago, Stephane would never have believed that INRIA would sponsor Pharo. That Pharo is used was key to letting them break in. He listed success stories such as Cmsbox.

In Squeak, Stephane would ask about something and they would reply "Yes, we can do that" and Stephane would reply "but I can't". Pharo must be easy to use.

Pharo 1.4 is the maintenance release. Pharo 2.0 is the new major release.

In 2.0 alpha, you can see the new Nautilus browser, the new Filesystem (from Colin Putney, better than 70s-style FileDirectory in which 'rename file' could take 20 minutes!). There is a new package implementation (and manifest) and a new system announcement framework. 2.0 will soon use metacello for its integration. The new core is 1.36Mb (or, in hazelnuts, 120Kb).

"How many use Smalllint? You do it every accept? No, you do it every month." Stephane wants it to be (fast enough and accurate enough to be) done on every accept.

Q(Jan Vrany) I use Smallint on every accept but it's useless because there are so many false positives? When they start using it on every accept they will see and fix or remove these false positive rules. They will use the meta-information.

There is a great deal of community work. Sven's Zinc work was shown in a 10-minute talk. There is also the serialiser of Martino et al, the key mapper of Guillermo Polito, Ring (Veronica Uquillas-Gomez) which handles classes in image or on disk or other locations, and many more. The Athens Graphics 2.0 will indirect the vector Graphics canvas so that Cairo or BitBlt or other backends can be used without difficulty (it is being used by a guy in Lugano to make a zoomable UI.)

"Who wants Opal 2.0?" Lots of hands "You see, Marcus, they all want Opal."

Pharo 3.0 will use Tanker as its binary code loader. Then code can be loaded fast. He wants to be able to load chosen packages (e.g. Seaside) from just a DVD with certification, and tests being run.

The process is that you save the configuration in your local repository. When you want to share it with the world, then you share it to the Metacello repository. (That repository should be able to go down without affecting your local work.)

The FFI situation in Pharo and Squeak is a mess. We want a single FFI for all cases, loadable without needing expertise. NativeBoost will give this.

Bernard van Rysegem is working on the browser.

The VM - every time they think Pharo is cool, they think about the VM and are brought down to earth again. Their first step was to ensure that everyone can compile the VM. In the past, a skilled sorceror from a high mountain came and did it and went away again. Now the VM is compiled every day. The code is in Git so that anyone who wants to fix VM stuff can do it and put in Git, not have a hard time presenting it for review.

How can we sustain Pharo? How do we structure the community? Should we rely on people's free time? In 2009, Stephane started talking to INRIA lawyers. The plan is to announce the consortium in October. INRIA will do any hiring. (French admin is not something you want to do for hiring one person. If Stephane submits a request for reimbursement of a 6 euro train ticket, it costs the French state 150 euros checking that request, so economies of scale are essential.)

People and companies can become consortium members, which means getting privileged access to the core development team, ability to influence the priorities of the next development, and some engineering support time. Pharo itself will remain free.

You can also be a sponsor (grades 'normal' and 'diamond'). Every member is a sponsor but you can also *just* be a sponsor: you get the logo and some nice feelings.

For companies, gold membership costs 4000 euros and gives 4 days of engineering time. Silver membership costs 2000 and gives 2 days. Bronze costs 1000 and gives 1 day. "So we are much more expensive than Cincom, but that's the price of freedom, no?"

Individuals pay 40 euros for membership or 99 euros for premium membership. Every member gets their own blog.

Pharo by Example was translated to French, Spanish (all dialects :-)) and Japanese. The German translation is mostly done. 'Deep into Pharo' is a

new book they hope to release by December.

Q(Georg) changes to Pharo: how to handle them in books? Stephane is a great admirer of Smalltalk by Example but he now wants to avoid tutorials. In 'Deep into Pharo', he will focus on the new things and videos. What James is doing shows the way: anyone can make a video. (Stefan Eggermont offered to advise anyone wanting to make a screencast during the sprint.)

Every single contribution is valuable; reporting that a bug is no longer there, or does not reproduce, is important. Two years ago, Stephane did not know Zinc. Now Zinc has replaced a lot of old stuff. Much work from Sven, of course, but there are different levels of participation.

He showed the list of Pharo sprints: some in 2009, 7 in 2010, 3 in 2011, 3 so far (counting this one) in 2012.

Q. When is the Seaside one-click image being updated to 1.4? When someone does it - by all means volunteer to do it.

VA Smalltalk, John O'Keefe
John's black eye is because he and Seth met ten Java programmers in a dark alley the other night. Two Smalltalk programmers can deal with ten Java programmers easily. (Or it could be to do with slipping on the sidewalk recently :-)). John has been working on VA for the last 20 years.

VASmalltalk 8.5.1 was released in March 2012, VASmalltalk 8.5.2 will be out in September. This talk is about things added in 8.5.2.

Seth has done work on code completion - see his talk. 8.5.2 will add it to inspectors and that will complete the work.

They're adding a Monticello importer: import mcz files, handle pool classes to pool dictionaries with _PRAGMA_ methods generated, map packages to applications, categories to subapplications (based on rules), generate loaded/removing methods. They have beta supprt of the zip/unzip: it works for Monticello's needs but needs testing beyond that.

DateAndTime class now offers TimeZone support, using the Olsen database. Conversion between timezones work. Time arithmetic spanning the summer/winter change works.

They had many logging frameworks and no common API. There is now a common logging framework, log4s, with ideas from log4j. This is .ini-file configurable, so behaviour can be changed from one run to another. They use block parameters to ensure performance stays good when not logging.

Their preference setting framework was improved in 8.5 and more for 8.5.2. Settings were in a workspace, or well hidden in some cases. Now validSettings holds arrays with non-homogeneous content, read from a file, and (almost) all their code now uses it.

External functions can be supplied by external .dll or .so files. These names can change over time, or between Windows and Linux. A method held the logical-to-physical mapping. Now an .ini file does, easier to change.

They use native platform widgets. In 852, they offer Windows Rebar control. (Rebar is the Toolbar that has a grab handle, can be resized, etc.) They also have better progress bars: smooth (not blocks, marqee-style). Date widgets are provided.

851 has Seaside 3.0.6+ and Grease 1.0.6+. 852 will make no changes (not enough to be worth changing); they'll do another step for the next release.

Documentation can be read from their website or installed locally and in either case is read in a browser. Now, Google search can search their documentation. (Google rules do not go through Javascript; almost all their content sat behind some javascript.) James Robertson has been creating podcasts for them and they are about to put up a page of 80 - 100 podcasts at http://www.instantiations.com/resources/st4u_videos.html.

They improved Envy performance in 8.5.1. In 8.5.2 they have improved performance of browsers to remote Envy repositories (the chatty protocol is now somewhat better). Glorp is upgraded to 0.4.190, and will move forward in each release till it is current. System>>getProcessId was added. Some Seaside components were added. WebServices have some cleanup code. They support Windows 8 (not metro - no suprise) and Ubuntu 12.

60+ bugs were fixed in 851, 50+ in 852. Releases will continue bi-annually, usually aimed at the STIC (just before) and ESUG (just after) conferences.

Tell John what VASmalltalk needs. He may not do it, but if you don't tell him then he for sure will not do it. He wants to do full unicode support (only supported in some places at the moment). He wants to keep Seaside up to date and provide continuation support (they do have a plan for that). He wants SST Servlet multipart support, Web Services tools and debug, and a more validating XML parser. They will finish zip support. They want to upgrade security support, moving to full OpenSSL 1.0 wrappers.

In 2011 John showed a prototype GTK support that will let them offer more widgets. They are still working on it. Fabian worked on it at the Hasso-Plattner institute. Change browser and merge tools could be improved.

Seth is their new VM lead developer: he'll look at GC, 64bit, etc.

JNIPort is on VAStGoodies. Every talk, John says that the next release will use the standard Windows installer. :-) It is now in beta and after some feedback it may finally happen! Unix installs will move to DEB/RDM (will support headless installs).

Educational licenses are free for teaching. Standard licences and evaluation licenses are available. As appropriate, email info or sales or support or john_okeefe, at instantiations.com. Open Source projects can commit on

http://www.instantiations.com/company/open-source.html.

Code Completion Seth Berman, VASmalltalk
Human ability to recall names is limited - especially for some of the weirder methods. The aim is to make suggestions be smart.

This feature arrived in VASmalltalk 8.5 (last August) driven by a comunnity discussion. Users in the community started both the requirements and the implementation. Seth started at Instantiations 1.5 years ago. He took this task and ran with it.

This feature showed larger overall changes in 8.5.1 than in 8.5: there were many new features and exposing of configuration options to users. They wanted to show an excellent standard experience, i.e. what people were used to in terms of VisualStudio, Eclipse, etc, then support configuration. Pharo users like ‘tab’ completion but much of the world is accustomed to ‘enter’, so ‘enter’ is the default setting and ‘tab’ is an available option.

By the time of VASmalltalk 8.5.2, the code completion utility was mature. 8.5.2 added code completion in inspectors, better configuration, theme and match highlighting engine. It could match CamelCase acronyms, not just prefixes. It had live filtering support using hotkeys - change the defaults just for this current match as the match progresses. It had block argument detection - get the blocks in an `ifTrue:ifFalse:` and similar. It supported drag-drop and new completion types.

Then he demoed. CTL + SPACE brought up an iconicised list of classes and methods, with (Sub)Application subclasses sorted to the end of the class list. You can turn CodeAssist on or off, or tweak it. You have code competion choices: 40 options sorted in various categories (e.g. the ‘basic’ category contains choices with default settings that they imply are standard for all IDEs in the market: does CTl + SPACE bring up code completion, whether to append or replace the existing filter when you type, whether to autocomplete by ‘enter’ or ‘tab’ or what, whether the CamelCase-match policy is very lenient or stricter, etc. Tooltips appear to tell you what an option means

Other, less common-to-all-IDEs options are the colours of the background and foreground. The colour of what you have matched is highlighted distinctly from rest of what you’ve matched, so you can type ‘ord’ and see ‘Ord’ highlighted differently in a highlighted ‘OrderedCollection’ that you have matched. This can be done by different colours, different fonts or whatever the user likes best to show.

The prefix match can be case sensitive or not. The CamelCase policy can be very strict (as it is in NetBeans or Eclipse), i.e. you *must* tell the code completer where the camel case letters are so every string must have R and W and S to match ReadWriteStream. If you have chosen to the highlight distint match regions option then the R W and S will be highlighted within ReadWriteStream.

Seth had seen a different algorithm in IntelliJ. They do it with Java's regular expression engine but VASmalltalk does it faster with their specific regex engine.

He set the CamelCase option to very lenient. Now ReadWriteStream is found by 'rws'. He apply the same CamelCase-matching algorithm to methods: `add:after:index:` can have the after-colon letters be matched as if caps, so 'aai' finds the method. For `OrderedCollection new` it does a class hierarchy sort and just shows you what `OrderedCollection` can understand. Hot keys can switch between alphabetic order and class' methods, then superclass' methods. All the hotkeys are around your right hand,. There are hotkeys to drop Object from the list, get rid of private, etc. default set in options and the hotkeys just let you escape from your defaults for a particular search.

Advanced options. You can open the popup without selecting the first element; this is for people who accidentally open it and hit enter or tab before they realise. You can make directional keys close the popup.

GemStone/S Update, Monty Williams, GemStone
Monty was one of the founders of GemStone back in 1982.

GemStoneS/32 is still used a lot in the financial sector (showed slide of huge trading floor). The 6.6.2 release, was made in February of this year, backporting some stuff from 3.0: better GC, backup and restore from NFS filesystems. End-of-life for the 32 bit stream is planned for October 2012. This means there will be no more major releases and no new product sales. Support for existing customers will go on until end-2015; no maintenance contracts will run longer than that.

GemStoneS/64 version 2.4.5.1 is the latest 2.4 release and a 2.4.5.2 is being worked on, but all major work is now in the 3.0 stream (showed slide of shipping activity, "These guys use 3.0").

In 3.0, a method is converted to native bytecode the first time you call it. The foreign function interface was presented at a past ESUG.

3.1 was released in July, and 3.1.0.1 was released two days ago. It has mid-level shared caches between the stone cache and the remote Shared Page Caches. These prevent redlining your network when 500 machines cluster to one. He showed the standard star config of local machines with local SPCs: you look in the remote SPC, then in the SPC on the stone central machine, and if what you seek is not there then you look on disk. The mid-level caches introduce another layer of hierarchy: groups of remote machines are clustered to the mid-level SPC machine which, if it too does not have the page, then asks the stone.

```
System mideLevelCacheConnect: hostName.
```

External password validation was asked for by many customers and is now there. LDAP can be used, e.g. to manage the situation when GemStone ID is ‘norm’ and LDAP ID is ‘norm.green’.

```
(AllUsers userWithId: ‘norm’)
  enableLDAPAuthenticationWithAlias: ‘norm.green’
```

In 3.1, many operations that were done in a single thread are now multi-threaded with native OS threads. How aggressively this is done is controlled by 2 parameters: the number of threads you may use and the CPU % across-all-cores threshold (the thread will sleep if it is exceeded).

ProfMonitor can now sample down to 1 microsecond. Monticello support is now built in - no need to install it. The old GemStone error handling is deprecated and ANSI is the norm. More selectors are inlined: `ifNil:`, `ifNil:ifNotNil:`, etc. You can now add instance variables on the fly, and without requiring instance migration.

There are some Smalltalk syntax changes: for example, the array literal syntax changed from #{1, 2, 3} to {1 . 2 . 3}.

Segment has been renamed GsObjectSecurityPolicy because the old name so routinely confused users. They have wanted to change it for years and have finally done so.

LargeInteger replaces LargePositiveInteger and LargeNegativeInteger. ScaledDecimal is renamed FixedPoint in 3.0 and its literal notation is now ANSI compliant.

There is a new class/metaclass hierarchy, inspired by Maglev, where the class Module appears in the Behavior, Class, MetaClass hierarchy.

They’ve made performance improvements. The old Gem / Stone locking and serialising of request response has been replaced by an atomic bit, so communication is no longer serialised. Session priority values have been added to ensure this loss of sequentiality does not mess up running important stuff before garbage collection, etc.

In 2.x all hash table row access were handled in a spin lock. Thus if three Gems wanted page 6, the second took twice as long and the third three times as long. Now all three can lookup in parallel.

Tranlogs are written to disk so work can be recovered, and can be a bottleneck - transaction not complete till written to disk. They replaced POSIX (buggy and slow) with their own code. This improved index auditing and commit times.

(Monty’s next “where we are used” slide was an offshore oil platform - a well in the North Sea - as they trade all the London crude.)

GemStone 3.1 has hot-standby databases. They used to have people copying tranlogs every 15 minutes and restarting quickly in case of

problem on main, but now the tranlog is replayed in real time.

It also has Locale-specific collation. They used the open-source International Component for Unicode libraries (written by IBM). It also has secure remote procedure calls for passwords.

Backup and restore is one of the things that have been multi-threaded. The performance improvement is an order of magnitude. Symbols are now garbage collected. Under low freespace conditions you can defer reclaim.

3.1 has IPv6 support, and IPv4-mapped IPv6 (and IPv4, of course :-)). You need to be aware whether your linux system is configured to support IPv6.

The 3.2 release will have thread-safe GC, optimised to run on VMWare virtualised machines.

MagLev status: MagLev has seduced some Rubyists into Smalltalk. VMWare focussed them elsewhere for a year but now MagLev is being done on 3.x (Monty is just finishing it now) and the Hasso-Plattner institute are partnering with them to do various improvements. If you just want Smalltalk, the MagLev VM will run Smalltalk.

Monty's last slide was of the first sale of GemStone to Stanley Su, showing Monty and Alan Otis (another founder of GemStone). Monty has been doing work like this for 42 years and next year, his birthday present is to travel a bit and do stuff. He's worked at GemStone for 31 years and has known everyone. VMWare is an excellent, well-run company, and his only regret is that for 31 years he's known everyone in the smallish GemStone company whereas you can't know everyone in a 13,000-strong company. He will likely come back to create a start-up company of 100 people.

Then Martin showed a video from the March STIC Biloxi conference, done as a lightening talk at the Bau Rivage casino on the gulf of Mexico. The talk was about Subnormal Floating Point Numbers: a six-bit floating point number with a sign bit and a fractional bit.

More and more complex equations express how the bits become the number. He restricted to positive floats which was simpler. He zoomed into number from 0 to 1 and got stuck at 1/8. Space them by 1/16ths.

Then he examined 64bit floating point numbers. Look at 1 divided by some huge number. He looked for zero on this scale, doubling distance each time. The pictures went out of the hotel and out of town and after a while we left earth. We passed the moon, passed venus, passed mercury, we passed the sun, reached mars on the far side and then saturn and then uranus and then neptune. Well outside the solar system, 8 times past the orbit of neptune, we found zero. So the gap between zero and a small number is in a sense huge, and remember: mind the gap!

Q. You presented a lot; how many people did that? There are 14 of us in the team counting Norm, who spends as little time as possible being a manager

and as much as he can being a programmer.

Smalltalk and Java Interoperability, Claus Gittinger, Jan Kurs, Jan Vrany, Marcel Hlopko
Why bother to interoperate? They can reuse Java libraries: many are bad but some are good. They can use them from Smalltalk even if they have to endure writing a little Java. Being able to offer seamless integration is a selling point. For example, Jan Kurs did some XML in Smalltalk, then was asked to do XSLT transforms. A Java program for SAXON already exists. This is an example of the general case: a solution may already exist in Java.

How to interoperate?

- You can serialize between Java and Smalltalk; the language boundary is a bottleneck.
- Another approach is to have a Smalltalk VM that emulates Java at the image level; problem: slow.
- A third way is to integrate Java support into the VM. That is what they did in StX:java. Of course, Java and Smalltalk differ.

Selectors in Java have types, not in Smalltalk.

```
JAVA java lang system out printLn: 2012.
JAVA java lang system out printLn: 'Hello'.
```

They create special dynamic proxy methods on demand. Their system finds the best type-match from the Java methods to serve the Smalltalk method: in this case printLn-*proxy* finds the appropriate Java printLn(..).

They have simple transformations to map String to Java.lang.String and likewise for Dictionary. These cause performance, identity and state synchronisation issues; they have ideas of how to improve.

How many monitors are entered if you evaluate 3+4 in Groovy (a simple Java runner)? [Niall-boaster-Ross: I'm the one who guessed right - 2000+] It is possible to emulate monitors with Semaphores; there's just a lot of it.

Exceptions in Java are not proceedable, and the `finally` handler is a only a syntactic entity. Suppose method `smalltalk3` with ensure block calls `smalltalk2` with ensure block, which calls `javaMethod` with `finally` handler, which calls `smalltalk1` with ensure block, which raises an exception. Smalltalk wants to put all handlers on the stack but `finally` is not an object you can put on the stack. We must execute `finally` when we meet it but now we've lost an ensure block - who will take responsibility for that.

They solve this by putting the exception in a wrapper, then put the program counter to the start of the `finally`, thus execute the `finally`, then see whether the finally took the exception and resume walking up the method chain.

Q. To do this, I must reimplement the Java VM? Yes; they reimplemented in C. Yes, that means you lose specific VM work of Oracle, etc.

Martin pointed out that you can make a Java VM modified to use a Smalltalk VM's object space; GemStone did that a long time ago. Jan Vrany remarked that VisualAge for Java translated Java bytecode into Smalltalk bytecode. The two bytecode sets differ so much that runtime performance was poor, though it was usable for development.

Q(David Chisnall) Java is a patent minefield. Have you licensed stuff? No, the non-free they do not get or use.

StX:libjava, Jan Vrany
They have a Java VM built into their Smalltalk environment. Their approach differs from all prior attempts that he knows of, such as JavaConnect that runs JVM in parallel and communicates, or the approaches that translate Java bytecode into Smalltalk bytecode. Their VM can run ST bytecode and Java bytecode and can JIT java code.

He opened a workspace with Java code:

```
hello := new stx.libjava.examples.HelloWorld();
for {int i = 0; ...}
  ...
```

He opened a browser and showed the source code of the HelloWorld Java class. Next he opened a Smalltalk workspace and demoed running Java from Smalltalk.

```
JAVA stx libjava examples HelloWorld main:
#('Smalltalk');

JAVA stx libjava examples HelloWorld new.
0 to: 9 do: ...
```

Q(Martin) marshalling? No. The Java objects are in the same object space, can be inspected with Smalltalk inspectors, etc. Passing a Java object to Smalltalk code has no more cost than passing a Smalltalk object to Smalltalk code.

He opened an example XML app - a list of CDs with xml info. SAXON is a program to handle XML but is far to large to port or reimplement. He showed the Smalltalk code that drives SAXON - a transliteration of driving the (wordy) Java API. Yesterday, they explained how to catch Java exceptions in Smalltalk. Today, he demoed the transform working, showing the CD data, and then corrupted the filename to see exception handling working. The debugger opened normally.

Nowadays, the web is the future (say many people, not Jan, but let's look at the web anyway). He showed running a web browser from their local machine running Tomcat; you just load it and run. He showed us the system properties, then changed the Java vendor to 'ESUG Fans' by Smalltalk invocation, then looked again at the web page and there was the change. He showed a Java servlet that provides a Smalltalk workspace: enter Smalltalk and evaluate it and the Java code on the server evaluates the Smalltalk code. He then send bad code to bring up a debugger showing a stack of mingled Smalltalk and Java code.

Current status: he can run most of the non-UI Java classes. You can browse and debug the code but you cannot accept Java code in a browser yet. This is not production-ready yet but is progressing. There are more than 1000 native methods and you really need to implement them all to run all Java code: they do not have the resources to do them all, especially for the UI ones that are tricky, undocumented and sometimes ridiculous in their behaviour. The performance is not dire but not yet good enough. Logging (ubiquitous in Java) also slows things. They can run the Eclipse tools without UI: parser, refactoring, etc. A real incremental environment for Java - i.e. accept in browser - will take longer.

Get this from https://swing.fot.cvut.cz/projects/stx-libjava. Blame Jan, not eXcept, if you find problems.

Q(Carsten) Saw similar talk from Claus years ago - was the project halted or did Java VM changes delay it? Claus did raw first step version for 1.0 but in 2010 he saw Java changes had broken it circa 1.1. They've rewritten reflection, and changed the class loader and lots of stuff. Now it runs in 1.6. Fixing needed much rewrite of code in the virtual machine, but the basic architecture is unchanged. Logging and finally blocks were the hard part. Claus did a cool job so they had a great starting point.

Q(Leandro) Why does your achitecture have a JIT and two interpreters, one Smalltalk, one Java? Many VMs have both: some code is rarely called or complex or otherwise not worth while to JIT. An interpreter is easy so they started there to make it work. Then they wrote the JIT; they could not have written a JIT at first. Things like finally blocks are hard to JIT - or rather, you'd have to switch back to unjitted. Thus the JIT complication is reduced.

Q(Leando) How does an object knows it is Smalltalk or Java ? From its class, noting that the top Java class is a subclass of Behavior.

Q(Christian) Oracle owns Java. Does that affect you? Jan has not seen a black car in front of his door yet. Oracle owns a patent on how to make logging fast, so they used another approach they found in an IBM paper. (IBM probably wrote the paper for the same reason - to avoid the patent.)

Q(Martin) A JVM must pass a set of tests to be called JVM; are there any technical, as opposed to resource, obstacles to your achieving that? Not that he knows, but he was unable to get the technology compatibility kit that does those tests. Deep in Java, there are methods saying thread wake if this method is called or if some other thing happens, or randomly.

Amber, Nicholas Petton
He showed the Wat video with some Ruby oddities and many Javascript oddities, including that

```
array == ! array;
",,," = new Array(4);
[[3]] == 3;
```

all return true, and ending with the 'NaNaNa...NaNa batman' return from

an expression. So obviously he'd rather work in Smalltalk, but he also wants to work in the web browser. Thus he created Amber.

Amber is written in itself, compiles to Javscript, is hosted on GitHub and has followers. He showed the Seaside counter example. He opened the Amber IDE and showed the `renderOn:` Seaside code (he was sending `counter asString` as he's already inside the browser). He also just updates the counter contents when the button is pressed. The class browser has the usual basic functions: search for implementors/senders, class, references, and so on, with shortcuts to inspect. doIt, printIt, etc. More functions will be added. As well as being a cool Smalltalk environment, it improves the workflow when writing client-side applications.

Next he showed Javascript. His slide had code to change the background and he just clicked the doIt button to implement it. His next slide was to move a widget and (after the usual demo hiccough - he reloaded his code) it moved, rotating the Amber logo. He showed a demo written entirely in Smalltalk that used the Canvas API to show patterns of rectangles that appeared and vanished in the slide. Amber uses proxies to talk to Javascript objects so if you know the API you can address it using only Smalltalk.

It wuld be too much work to bind everything between Smalltalk and Javascript, so a String in Amber is a String in Javascript and for many other things where it makes sense to map one-to-one.

He also lets Javascript call methods on Smalltalk objects, using `_yourself` for `yourself` and similar. He sent the document message to the window object to demo this, then inspected the window object, which was a javascript object but you get the Smalltalk inspector showing it. Thus you can interact with Javascript for free.

Javascript engines are not just in the browser; they are also in phones and everywhere. It runs on Node.js, which is free and growing fast. It also runs on Gnome shell and KDE - Amber runs on his kindle. The files you need in bin are 'amber' (does fully functional REPL) and 'amberc' (compiles Pharo almost without exception to Javascript).

He showed the toy rabbit (Karrotz) example with a Javascript API so drivable from Amber. He made the rabbit say "Hello ESUG 2012", wiggle its ears, etc.

The debugger in 0.9.1 (latest version) does not yet have stepping (will have it soon). He demoed reaching a halt, inspecting objects, removing the halt and executing code from inspector.

Q. How easy is it to get at SVG objects and to trap events on those objects? Anything you can see in Javascript you can see in Amber. A callback is a block closure in Amber and can be done; it is fully transparent.

Q. Calling Javascript methods with multiple or variable arguments? They tried passing an array but that was ugly so they reused what GnuSmalltalk

does for Java inlining: you add extra keywords when you want extra arguments. If `css:` is the method then you can write `css:color:` or just `css:value:` because what the second argument keyword is does not matter, only that a second keyword is there.

Q(Martin) non-local return but not exceptions? Yes he has non-local return and exceptions but he cannot resume an exception if his compiler cannot inline it, which is so in various cases.

Q(Martin) what is your intent in Amber (obviously, it will appeal to a Smalltalker that does not want to learn Javascript)? You have to know Javscript APIs. It is much easier and better tooled in Smalltalk. He sees few people coming from the Javascript world but quite a few from Ruby or from Dart and similar (most in the Amber newsgroup are not Smalltalkers).

Q(Georg) Reflection on Javascript classes? He has no plans to improve Javascript. Smalltalk reflection is important to him.

Q(Georg) Dart? He likes Dart's compiler to Javascript and he learned a lot from looking at its compiler. He sees the language as a failure: hyped but too close to what we've seen. It seems to have nothing to offer. He doesn't hate it but he doesn't like it that much.

Q. Can you use libraries like Zink? Yes, load library, call it from Smalltalk. (Can be loaded incrementally from server? Yes.)

This presentation of Amber was made in Amber.

Frameworks and Tools

Advanced Visualisations to Tame Wild Program Execution, Vanessa Pena, Alexandre Bergel, Juan Pablo Sandoval, Pablo Estefo

They've been working for three years on how to solve hard performance problems.

Kai is an execution profiler, unlike traditional profilers which focussed on the execution stack, which did not help answering "is there a slow method being called too often?" Kai offers a visual comparison of number of executions and time of method. He showed a typical performance-annotated stack of a standard tool, and then a Kai representation.

The height is the time spent, the width is the number of calls and the color is the number of objects on which the method was called. In another graph, colour shows whether a side effect has occured, i.e. the state of the receiver changed. This can help in deciding whether caching would help.

Vanessa started (in VisualWorks - there is also a Pharo port and an Amber one). She chose what to profile (expressed in Smalltalk code: templates and specific choices are offered) and got her visualisation. The left of the view showed a textual legend of the data. She looked at various methods that took a long time or were executed often or were applied to many objects.

Q(Stephane) Call graph? Obtained by popup menu. However there are often many, many classes so the per-class view is a good place to start.

Popups let her get incoming invocations, outgoing invocations or just see the code. You can also jump instantly to the method or class in an RB. You can also get the standard stack-oriented view at the method. She also went from a class to the overall graph.

Next, Alexandre presented a small scenario applying Kai to the Mondrian utility. The method `bounds` appeared very expensive. After looking where it was called, it appeared local caching would work and indeed it did. This gave a 43% speed up. He also looked at some of the thick yellow methods.

Q(Christian) Same profile? They execute twice: once to get standard performance analysis info, then again for e.g. side effects data. If they tried to collect both kinds of informatin at the same time, the overhead of the second run's collection would corrupt the first run's time measurements.

Hapao does test coverage. Traditional coverage tools have a binary view. But the question you often want to answer is, "Which method should I test next to increase coverage most?" He showed a diagram. The lines inside classes are `self` calls. The height is the complexity (the amount of control structure, got from anaysing the AST and applying the McCabe metric). The other visual measures (width, colour) are as above.

Vanessa opened the tool, selected items and asked for coverage. Again, the left side of the pane showed text on the coverage values and a legend on what the visualised measures meant. The rest shows unreached methods in red and test methods (i.e. roots of the coverage run) in green. Height of a method shows its complexity. The right-click menu lets you look at classes or methods, or popup with further views in that context. For example, she can see which tests reached that method (or that class, showing which specific methods). Next Alexandre showed the large scale coverage graphs of Moose in two versions.

Roassal, following on from Mondrian, is an interactive visualisation engine. A short Smalltalk script lets you define what you view, assigning code and performance metrics to shape, size and colour of your views.

In the Pharo version, they demoed typing code in the Roassal Easel Transcript and getting immediate visual results in the display pane. All the OrderedCollection subclasses were displayed, at first as a simple line of nameless boxes, and then he began displaying the hierarchy relationship, making width be the number of instVars and height the amount of code.

Q(Stephane) Roassal improvements over Mondrian? The cleaner central core makes it easier to extend. He will show examples you cannot do in Mondrian. He mapped classes to coloured shapes and showed zooming in and out, and dragging. Roassal is also faster. Alexandre's goal is to be able to display 10 million nodes two years from now.

He showed visualising the NY fire department phone call incidents data. Height is time spent to react to call, width is time spent on the call. Colour represents whether there was an actual fire and how serious it was.

Roassal can display with Cairo on both VisualWorks and Pharo (remember to load the Cairo parcel, not prerequed, in VW), with thanks to Chris for VisualWorks and Igor for Pharo. They have ported Roassal to Amber and will demo that this afternoon.

They are working on multi-diminsional profiling: showing the changes between two runs. He showed an example: something in Pharo 2.0 was slower because in the XML parsing the `atEnd` method was slower. Pablo is working on seeing how to refactor tests to improve coverage: he showed a browser that displayed panes of tests along with the central info.

Alexandre thanked ESUG, Chris T, Cincom and all who use and give feedback on these tools.

Q(Carsten) In the comparison of two versions, what did the height and width mean? Alexandre returned to the graph and showed one method that was light red (a bit slower) and another that was very red (much slower). Width shows the change in execution times. The graph also shows which methods had code changes between the versions.

Smalltalk in the Cloud, James Foster
CloudFoundry is an open framework from VMWare: “the platform as a service”.

Self-hosting means you are responsible for the building, the power, the air conditioning, the hardware and the software. A data centre gets rid of the first few. Renting space from Amazon on a virtual machine lets you install your own operating system and manage just that and the application software. CloudFoundry is an example of providing the platform as a service: someone else provides the hardware and the operating system; all you need to provide is your software. (The last stage of outsourcing your activities would be Sales Force or similar: “software as a service”.)

You choose your cloud, push your app and bind services. If later, success comes, you upgrade from one instance to a hundred. You can have a public cloud or a private cloud, or a micro cloud which just runs on your machine.

CloudFoundry.COM is the infrastructure that VMWare provides. MicroCloud can be downloaded to your own machine. CloudFoundry.org is an open-source provider.

The Browser communicates (via routers) with applications in the cloud. Underneath, the infrastructure provides a pool that manages applications, cloud controllers, a load-balancer, etc. A VNC client lets developers talk to their apps. Browser requests come in through the load-balancer and get handled by a web app running in a cloud.

```
vmc instances MyApp 3
```

will update you to 3 instances. When browsers address MyApp, the incoming requests will be allocated to one or other of them by CloudFoundry's load-balancer.

He then showed the trivial Ruby app that CloudFoundry offer as their default example. As good practice, you try the app locally

```
ruby env.rb
```

then execute

```
vmc target
```

to set the context for the following command(s). It returns James' registered domain: api.vcap.jfoster which he pinged to show it was 127.0.0.1 on his laptop. He showed that VMWare fusion was running CloudFoundry on his laptop. He did

```
vmc push ruby-env
```

and replied to the how many, what services, etc., queries and then was able to connect from his local browser on his laptop. Then he retargetted away from his micro-cloud to a public cloud

```
vnc target api.smalltalkcon.org
vnc push ruby-env
```

and I (Niall) pointed my browser at http://ruby-env.smalltalkcon.org/ and saw the same app. Now it is in a public cloud, not just in his micro cloud.

That was Ruby but we want Smalltalk. How is Smalltalk different; let me count the ways. Smalltalk is a monolithic image with full application and framework instead of being text files and app code with external libraries and frameworks that can be pre-loaded on the server.

But in fact it is not that different. Code can exist outside the image as packages and file-ins. There are files that can be under a root directory. CloudFoundry has a process "staging" which is under our control: it determines the actions taken before launching. In our staging, we must launch Smalltalk, load the code, save the image and carry on.

James showed a Pharo application (this will work with any Smalltalk dialect) with files app/aida.st, Aida.changes, Aida.image, PharoDebug.log, PharoV10.sources, etc. In our Smalltalk app, we must say what port to listen to. An environment variable provided by the cloud tells us the port we have been given by the framework.

```
portString := SmalltalkImage getSystemVariable: 3.
```

(N.B., the system will not let you compile a method unless you define who the developer is, for its timestamp system, in a line at the start of the file.) He registered Aida in the VMC's list of supported frameworks, then demoed by cd'ing to an /aida directory and doing

```
vmc push aida
```

He worked through the dialogs and then we connected to http://aida.smalltalkcon.org/ and received an Aida start page with app-specific last line ‘Listening to ESUG 2012 on port: 37082’. Next, he registered MySQLBalanceApp and ImageBalanceApp and did

```
vmc push balance
```

When done, http://balance.smalltalkcon.org/MySqlBalance showed the login page for the application. (He gave us the password and we could login.)

Tim then described how, at the Hasso-Platner Institute, second year students had to write Seaside apps and (aside from “I don’t have my beloved Mac keybindings”) they disliked the work of configuring apache to make their app visible. In February, HPI set up a micro cloud; 3 student groups used it. They found it very easy to use. Next term, all students will be told to use it. A very tiny .mcz package helps generate a droplet for pushing to vmc.

Q(Georg) pricing? CloudFoundry is open source. You can fork it on GitHub, take James’ Aida branch, install and run it. You are not locked in.

Q(Nick) GemStone? There is a package with a GemStone database and topaz API.

Q(Stephane) Infrastructure - what monitoring tools do they use to monitor these farms of clouds? These tools are provided by a separate part of VMWare. Every month James gets an email about the latest new tool or version. Various people provide clouds and various (other) people provide tools to monitor and manage them. VMWare are keeping these tools more propriatory. If you want to run a data centre, VMWare would be delighted to have you as a customer.

Petit Parser Tutorial, Guillame Leveque
Guillame programmed for 6 months in Java in J2EE. After that experience, he was very happy to discover Smalltalk. He has been programming in Smalltalk since January.

The tutorial example was to parse PHP (using a grammar obtained from the Symphony PHP framework), using standard XP test-driven development. He gave us a test, we worked on answers, then he showed as an example answer. Test

```
PHPIslandGrammarTest>>testAccessPrivate
  self parse: 'private' rule: #accessModifier
```

is passed by

```
accessModifier
  ^'public' asParser
    / 'private' asParser
    / 'protected' asParser
```

(put the most commonly expected value first, the least common last). We then wrote a test for declarations. The first character of a name cannot be a number, but it can be an underscore.

```
PHPIslandGrammarTest>>testDeclareName
  self parse: 'Esug2012' rule: #declareName.
  self parse: '_Esug2012' rule: #declareName.
  self fail: '2012Esug' rule: #declareName.
```

(star means BNF * i.e. 0 or more of the productions). This is passed by

```
declareName
  ^(#letter asParser / $_ asParser)
    ,(#word asParser / $_ asParser) star
```

Next we looked at class declarations.

```
PHPIslandGrammarTest>>testClassDeclaration
  self
    parse: 'class DemoExtension'
    rule: #classDeclaration.
```

which is passed by

```
classDeclaration
  ^'class' asParser
    , #blank as Parser plus
    , declarationName
```

He used an instance variable `'declarationName'`, not a method call of `self declarationName`, to ensure that any accidental cycles would not matter. If he had called `self declarationName` and he had a cycle in the rules at any point, the test would run forever.

He used pastebin.com to make some PHP class declarations available to the group, thus invited us to write the call in the test

```
PHPIslandGrammarTest>>testCompleteClass
  self parse: (self phpClass) rule: #completeClass.
```

solved by

```
completeClass
  ^#blank star,
    #blank star, classDeclaration,
    #blank star, classDeclaration negate star
```

where we must then parse the negated stuff in more detail (could use `not` but `negate` consumes). At the moment, this gets us the classes parsed but except for the name, we do not yet differentiate contents. (That ended the allocated time for the tutorial.)

Real-World Seaside Applications, Nick Ager
Nick has built real deployed apps and will share his experience.

There are many wysiwyg editors that can easily be included in your page with a javascript call. However there are bad guys out there. If you allow raw HTML upload, you risk having people upload nasty javascript that will redirect. There is a Yahoo UI editor that is configurable (choose which buttons to include or omit). You could do some parsing in the browser - it understands javascript and has a good dom - and whitelist only such javascript expressions as you can understand. However, even easier is

```
self configuration allowWikiTextEditing.
```

Nick's editor converts the HTML into Pier's wiki text and only the output from that is being passed to the image. You are therefore safe, since Pier WikiText cannot understand Javascript. This was developed for Pier but will work in any Seaside component as he only uses a renderer to convert it to HTML again. General rule: once you store it outside HTML, you can manipulate it and are safer.

Uploading files: he showed an example from the Seaside book: "A beautiful example of how elegant Seaside is, but the example is not real-world since the file is going into the image before being written, so the image balloons as the upload happens. In the real world, apache or whatever sniffs for file uploads and puts them in the file system, rewriting headers to tell the system where the files were put. Nick achieves this with an

```
html hiddenInput ...
```

call. File downloading works the same. Your front-end server is good at serving files, so you want to configure it to serve css, jpeg, etc. for you. Usually you write the files to a known location and have the server point at that.

Your site's appearance must impress. Twitter bootstrap is a collection of css for buttons, tabs, drop-downs and so on. There are good examples of people using this (which, because his internet connection had just gone down, he could not show). He showed his Pier site, which uses these.

WAFileMetadataLibrary used to require that all files be in the same directory, which is not how modern libraries like Twitter bootstrap work - they have substructure. Now, there is a subclass that can handle that; the call you need is `recursivelyAddAllFilesIn:`, and twitter bootstrap input widget features can be exploited via `twbsPlaceHolderText:`.

To harden for deployment, make your server know only about the root class of the deployed app, not any other apps you have have installed. Flushing the monticello caches before uploading makes sense.

Alas, seasidehosting.st does not work with anything later than Pharo 1.2. It was great for testing although not of course usable for any kind of production. When choosing your real data server, be aware that having low latency is important. Living in London, Nick was happy with linode (they have a data centre there) and Amazon has also been good for him. (Arocu have builds for lots of things; it would be good if it could also have a

Seaside build or two - anyone want to?)

Q. Amazon? At first it seemed great, but if you are manually configuring it then it soon goes out of date, plus it's an image - what has been done to it? Norbert wrote a config tool to address this.

Nick also gave a good talk on JQuery, which sadly I missed - see his slides After that talk, Karsten Kuche applied Nick's approach to SeaBreeze.

GIT and Metacello, Dale Henrichs, VMWare
Storing source files is not enough; you must store package structure. Otto told Dale he had a utility for that, and Dale took note.

There is a FileTree directory structure: package root has class subroots with method sub-sub-roots. You also have e.g. Object.extension roots for extensions (Dale had to jump real high to show it on his high-projected slide :-)). There are other roots for specifics, e.g. properties.json.

Git is a distributed version control system for source code. Distributed repositories can be used locally and then easily remerged into the central repository. Why Git? Vincent Driessen in 2010 wrote an answer: "It really changed the way developers think about merging and branching. It used to be scary but with Git it was easy for 00s of people around the world to contribute." Vincent showed a successful model for Git merging. (Useful slide) Master branch, hotfix branch, release branch (for preparing releases and merging in bugfixes), development branch (main work) and feature branches (for specific major extras).

Using Git makes it easy to use GitHub. You can use Mercurial or SVN or whatever with GitHub, but Git and GitHub are a natural pairing. GitHub has a simpler flow than Vincent's all-cases diagram. The development and master branches are the same, and you have feature branches. You have no hotfix or release branch. The master branch is always deployed. You used named branches off of master for all work, then a PULL request submits your work to master: a code review decides whether it goes in or not.

Dale showed the web page list of a simple PULL request, followed by review and OK and commit. Then he showed more complex one where review raised comments, well-displayed against the code diffs. Interactive code reviewed between people 8 hours apart can work well in this process.

Every commit to GitHub runs a build which runs test. Recently, they upped this so every PULL request gets a build. Dale showed where he submitted a PULL and the build found a bug (LatinMirror has been withdrawn).

Q(Stephane) Does 3-way merge? Yes. Green button if 3-way merge has no conflicts.

Q(Stephane) Jenkins bridge? Probably.

It runs tests in parallel on multiple builds. Dale showed a list of tests where

he passed in Pharo 1.4, failed in Pharo 1.3. This is done by ‘travis’ (not Travis Griggs but a system bot called ‘travisbot’).

Q(Stephane) Where are the smalltalk machines running? Serge Stinkwitch created the Smalltalk VM and Dale took it. The test downloads this VM and runs. This piggybacks on the Erlang engine, Dale thinks.

You create a file, based on a template Dale will give you. Simple Smalltalk code creates the Metacello repository, invokes tests.

Git has a million users and 112 people who work at GitHub. We can leverage this.

Cypress is a cross-Smalltalk-dialect package structure. How many share code between Smalltalk dialects? (many hands). How many like to do it? (no hands). 5 years ago Dale thought Monticello could be the common cross-dialect approach but there are reasons why not: it has too many tendrils. Cypress just says: have a directory structure.

Q(Stephane) external resources? Yes, Git includes any file so we can add that (and define it cross-dialect if we can).

FileTree is a specific implementation of the Cypress spec. It includes Monticello integration with Git. FileTree is just a repository type in Monticello terms.

https://github.com/CampSmalltalk/Cypress contains stuff incuding the picture of the napkin where he and Travis (Griggs, not the bot) defined Cypress. Cypress implementations exist for a range of dialects including Amber, Cuis, GemStone, Pharo, Squeak and VW. Travis Griggs did a VW version that talks to Git, called STIG, which Martin is now stewarding.

github://user/project/[:<SHA>][/ <path>]

A single `baseline:` method in the BaselineOf class, plus Git, does a lot of what Metacello used to do; using Git makes Metacello much simpler.

Q(Stephane) make string user=dale, project=... so others can use generically? The string is specific to GitHub saying that’s where I want it. Having done this work in last 6 months to make himself happy, he is now ready to discuss how to make everyone happy. He’s created an API to replace the standard load expression.

The Metacello stuff will be in Pharo 2.0 and Squeak 4.4. The preview is MetacelloPreview 1.0-beta.32.2, looking for a few volunteers to use it and suffer in a good cause (of making it better). Dale suffered a lot learning Git but google is his friend - others had already met his problems.

FsGit is in-image support for Git and being worked on by a couple of guys. Please come to the workshops and ask ‘stupid’ questions - every stupid question means there is something wrong with the documentation.

Q. Merging Monticello meta-data? You cannot merge meta-data. This has been left to avoid problems.

Q(Johan Brichau) Can we use Monticello for branching (people are used to it) and GitHub for merging? Stefan Eggermont argued you need to keep your branches small, so maybe that would not work too well.

The Metacello Doctor is IN, Dale Heinrichs, GemStone
I only caught part of Dale's workshop. For more info, see http://www.slideshare.net/esug/of-metacello-git-scripting-and-things.

1) Dale looked at the "No baseline pragma" error. Start by browsing `baseline101: spec`. GitHub does not clone the repository, so you can't checkin: you load from GitHub.

```
Metacello new
  repository:
    'github://delware/StoryBoard:master/packages';
  baseline: 'StoryBoard';
  load.
```

You are pointing at GitHub and need to switch to local by doing a get.

```
Metacello new
  repository: '';
  baseline: 'StoryBoard';
  get.
```

i.e. `get` instead of `load`.

2) Next, Tim had a feature request. You want to add a license file to your project and a license header "copyright <year> <name> - see licence at <location>" in every file. Can you give it the means to apply this at some point late in a project development? Yes, there is a what-we're-doing line that could be used for this. License text could be added to a repository as a property. In GitHub you can have multiple package repositories, so you could handle various licences.

You don't go through the disk to GitHub, you go direct from the image.

3) The next questioner had problem with changes. Sometimes, things go weird in Monticello: press this changes button and nothing appears changed; press another button and everything appears changed. What is going on? In Monticello, a package wants to compare to self or an ancestor and if the local file system does not have an ancestor then it can get confused. Dale guessed that the questioner had been shown the dialog 'Dirty - Proceed or Load?' and had chosen 'Proceed'. This relates to a known issue where "There's a mountain to move" but it will be fixed. (Also, in Squeak, some patches have broken common behaviour, e.g. `readAllFiles` gives strings or objects depending on the patch level.)

4) Amber lives in the world between Smalltalk and Javascript so has a choice - use some Javascript stuff or not? Metacello was originally doing some of what GitHub now can do for us. Jan Vrany is porting Metacello to

StX, not to use Monticello but to use Metacello to manage dependencies and etc. in St/X streams.

Advanced Seaside, Phillippe Marschall
This talk is about the Seaside-Core component, which could be called Seaside meta (in 3.x, Seaside was split into various components: Core, JQuery, Javascript, etc.). This component is about the basic request architecture. The Seaside-REST component is build upon the Seaside-Core so you can just load it for RESTful use. The core gives you objects, not strings. When correctly used, the overhead of using it versus programming the web server directly is negligible.

WARequestHandler gets a request and receives a response. They can handle multiple requests in parallel so do not put request-specific state in a subclass of that class (or use semaphores, but that complicates things). Request handlers can be chained: e.g. /conferences passes to /esug which passes to /2012 which handles the request.

WARequestFilter is a wrapper of a request handler. Think method wrapper - a WARequestFilter would do logging, security and similar things.

WASession handles continuations, properties. In Seaside 3 there is also WARequestContext that holds the current request and current response that you can use to get info that earlier you would have got from the session. The WARequestContext can also give you the handler stack, i.e. the stack of WARequestHandlers invoked before the current one. A Seaside application is a WARequestHandler and we have to access it to read the configuration; the need to find it is one example of why we may need to walk the stack.

WAPathConsumer: /esug/2012/... consumes /esug/ and then consumes /2012/ and so on. This simplifies coding by managing the consumed and the not-yet-consumed parts of the URL.

WADispatcher: the global Seaside dispatcher takes all requests and dispatches them.

WAServerAdapter: Seaside does not come with a web server. This class adapts generic convenience methods to the native methods understood by a particular server you use.

Philippe has created a joke web framework Frank (inspired by the Sinatra framework). He showed a trivial app - one class-side `register` method and an index and schedule for esug2012 - and showed it in the browser. Loaded from GitHub, this just needed Grease, Seaside-Core and his app.

Frank implements

```
handleFiltered: aRequest
  aRequest consumer atEnd
    ifTrue: [self listing...
    ifFalse: [self receive: ... “gets next on path”]
```

He implemented `receive:` as a trivial `respondsTo:` check and call but never do this in a real web production framework (arbitrary code could be invoked by users). `handleSelector:context:` does the `perform:`.

In Seaside 2 you could always do `WAResponse new`. In Seaside 3, the response must be more tied to the server, e.g. if you do streaming and the server wants to do buffer recycling. Now you must use the provided creation methods.

Sometimes, people ask if Seaside, with all its objects, will be fast enough. He started a basic Apache, utterly untuned, merely connected to his Seaside image. He used Apache bench, keepAlive and 100,000 requests. He asked us what we expected (we guessed it would finish within his talk time :-)) and then ran it. It handled 10,000 requests per second.

REST merely means pretty URLs for Web Services. Use pragmas.

```
index
  <get>
  <path: ‘index’>
  ‘<h1>Hello World</h1>’
```

or

```
index
  <get>
  <path: ‘index’>
  <produces: ‘text/html’>
  ‘<hello>Hello World</hello>’
```

and you can have

```
index
  <get>
  <path: ‘{name}/_all_docs?start={start}...’>
```

For development, you can set `shouldCacheRoutes` but be sure always to take it out again for production deployment.

WARestfulComponentFilter can be used on these e.g. to do a count.

Seaside-REST is in Pharo, Gemstone and VASmalltalk (it will be ported to VisualWorks during the sprint after the conference).

Philippe then spoke about Seaside 3.1 stuff.

Session tracking is now its own class that you can subclass e.g. for better cookie control. You can choose query fields only, cookies only, cookies if supported and query fields otherwise, cookie for browser but IP for crawler. Another other option is to use the SSL session id (but then you must leave it alone).

The path parameter now does not have to be a hidden form parameter.

WAMain is now gone. There is a new continuation that drives the render loop. If you used to subclass WAMain, you should now instead subclass WASessionContinuation.

Previously, JSON was part of Javascript in their approach but now it has its own package. There were enough subtle differences to require that.

Streaming now has on-demand flushing - no longer must you stream every response or none at all. Continuation can flush after rendering </head>.

They did much work on HTML5, most of which is already in 3.0.x but multiple callbacks are not there yet. Document handlers used to be stored in the same dictionary as sessions which caused complications due to the heterogeneity of content. Now document handlers are stored in the session so when the session expires, so do the document handlers.

Nick Ager is now part of the core team. He's already presented Pier 3 and Magritte 3. They like the quality of the work he has done. Now the walkback in VisualWorks opens at the correct point in the stack.

Does anyone know morphic well enough to make a control panel for Seaside in Pharo 2?

He thanked the sponsers of the Seaside sprint: 2rivers, Reza Rezzavi, etc.

Q(Hans Martin-Mostner) Web sockets? Not yet. Sink and WebClient have something - ask Sven for status.

Q(Martin Kobetic) When VisualWorks integrated streaming suppport, Seaside was trying to write the headers; it would have been better if writing the headers was in a separate callback? Seaside now doesn't write headers - there is a dictionary. Try subclassing WAResponse, as comanche does. (Further discussion and work offline.)

Glorp, Karsten Kuche
Glorp maps between tables and objects. Karsten started working with Glorp last year. He created a demo descriptor system.

```
classModelForLogItem: aClassModel
  aClassModel newAttributeNamed: ...

tableForLogItem: aTable
  (aTable
    createFieldname: 'id'
    type: platform serial) bePrimaryKey.
  aTable
    createFialedNamed: 'timetamp'
    platform datetime.
  ...
```

Now we have modelled our classes and our tables, we must model the map between them.

```
descriptorForlogItem:
  ...
  aDescriptor directMapping: #timetamp ...
```

That demo took us 4 minutes to describe the class, the table and the mapping for one class with four instance variables. So we'd rather have

tool support for this. The ObjectStudio mapping tool will be presented by Dirk and Arden tomorrow. ActiveRecord was first released in WebVelocity, taken from Ruby on Rails. ActiveRecord applies specific rules but only works if you have control of the database (can give hints).

Ideally, why not have automatic conversion: read the database tables, create classes and Glorp mappings between them. You then modify the result. In an ideal scenario, your DB is already well designed to support your desired class model. In others, you have the database, ill-designed or old, and you will modify the mappings to a new class model.

The Glorp Mapping Creator was built by Karsten. It lets you choose mappings for the DB schema elements. He opened its UI and created an empty SQLite Store database in 7.9. Then he opened his tool and made some modifications. He could have applied a rule, e.g. to strip all the TW_ prefaces. Thus he acquired a Glorp mapping. He published a package and looked at the objects in this mapping. At present, his tool checks foreign key constraints so causes one-one mappings and one-many mappings.

Orpheus is “one ring to rule them all” for database usage. Instead of designing a specific schema for a specific application, Orpheus’ approach is to have a table with single column for timestamps, a table with single column for integers, etc. (Virtual tables can then join these to present the more common kind of database views.). You just have to describe your persistent classes (i.e. provide sufficient DB type info for their instvars) and Orpheus calculates the mapping of these to the base tables.

They migrated Orpheus to Glorp. You add just a little info to the basic descriptor system. Suppose you have a class in which a person can have name ‘Karsten’, live in ‘Munich’ at zip 80331, etc.

```
aClassModel
  newVirtualTable: ....
  baseType: #TinyString
  ...
```

lets you store the TinyStrings of names like ‘Karsten’ apparently in the virtual table of the class but actually in the real table that holds all TinyStrings in its single column.

Glorp keeps proxies, it does not replace them. This is faster than replacing (using `become:` is slow) but you must remember that == does not work. In Glorp, proxies are bound to sessions, so you cannot compare with objects in another session. There is no session pool.

Orpheus uses method inlining. Consider a Glorp query for person, city and country where all persons are in Germany. You can ask

```
  where: [:each | each country name = ‘Germany’]
```

but if you define

```
countryName
  ^self city country name
```

Glorp will not let you ask

```
where: [:each | each countryName = 'Germany']
```

Allowing this is in fact easy to implement ('easy' means it took him 2 days thought). He also addressed query optimisation. Glorp can generate code like

```
[:a | a = 2 & (false | true) | NOT(true)]
```

which can be simplified to

```
[:a | a = 2]
```

but although Glorp can optimise some things, it did not catch the above; they fixed that.

What does Glorp do if you put 50 char string into varchar[20]? Glorp runs past that just fine and only later do you find you've lost 30 chars; another thing to fix.

Multithreading: one process per session but two sessions share no objects - that is a problem.

All objects in Glorp are cached. An object in a cache will not be refreshed just because it is becomes out of date in the database through another process. You must explicitly refresh the cache to see if that is so.

Modifying: you register objects before changing them. If your code path changes an object before it is registered, Glorp does not recognise it has changed. If no later change is made after it is registered then, despite registering it, the object, and its pre-registration change, will not be written.

(Niall: I agree with this as the default Glorp strategy but see also methods `registerAsNew:`, `save:` and `forceSaveOf:` to handle special cases.)

The common approach to recording the Glorp model of your application puts class names and table names into selectors: `tableFor<Name>`, `classModelFor<Name>:`, `descriptorFor<Name>:`, . One could use pragmas but support for this is not there at the moment. (Also these long-prefix methods can lose a class name under the class browser's next pane.)

Karsten closed by stressing that Glorp is superb and cool. Don't be scared by the things he has mentioned, just be aware of them when writing your Glorp-using applications.

Q(Niall) We are working on providing other Glorp schema recording patterns as alternatives to the 'class-name in method state' and also providing refactorings that include changing these parts of methods.

Q(Christian) Why has running Store on Glorp made Store slow? Karsten believes it is the session problem. The approach has been to create too many sessions, so too often reading same object again.

Presenty, Denis Kudryashov
After the usual demo hiccough opening his presentation, Denis explained that presenty was a way to separate business logic from presentation layer. He sees widgets (combo box, listbox etc.) as just designer terminology. There are many browsers but it is all the same info.

He showed some alternative browser ideas, some just to indicate what could be done rather than recommending what to do. PtyGuide drives app, PtyUser has domain knowledge and PtyUser + PtyGuide generates new tasks. PtyTask has PtyTaskContext which is the context for a PtyTaskActivationStrategy. This relates via subclasses to PtyAreaActivationStrategy. Standard smalltalk expresses the specific behaviour of these, e.g. user selects item from list.

Button is a way to execute an action, voice, gesture and shortcuts being other possible ways. Control flow is assisted by PtyReturnValueFromTask, PtyReturnToPreviousTask.

The PrototypeManager package does not depend on Presenty and is published as MIT.

Future work will look at making it possible to edit and explore objects, and to parametrise user actions.

Q(Noury) What is missing from Pharo that would help you? This is done in Pharo 1.4. He's not keen on how morphic works.

Q(Leandro) You copy many objects in the graph from a prototype. How do you copy? He uses the morphic deep copy.

Fuel, Mariano Martinez Peck
Mariano thanked Martin Dias, a key Fuel and Tanker developer, and also ESUG and others who have supported the work. Tanker lets you export and import packages.

He demoed in a Pharo 2.0 image that was one week old, with a new ClassBuilder, so beware! Pavel has been working for 7 years on creating a small kernel image. He started a VM on the command line and passed the script to export the packages using tanker. He creates two files, one for the binary of tanker and the other for the source code. It took 10 seconds to export the sources and 5 secs to export the tanker binary, representing 1500 classes in 200 packages. He showed the kernel image running as a script server on the command line, doing 3 + 4, then gave it the command to load the tanker output to make the new image. Loading took 0.5 secs to create the classes and methods but 4.5 secs to initialize them.

Tanker cannot know the order of class initialization. A hook is provided where you can define postLoad actions that call a block. Much code can be needed to initialize e.g. morphic - he scrolled up and down it. They also export objects that may be useful or hard to initialize. He opened the image and showed it had its sources and worked OK.

His next example exported Seaside, then opened another image and imported them, taking 30 seconds typically (in fact, only 16 in his demo), again with most time spent in class initialization. He ran tests and saw 4 failures (known - he took a pre-2.0 Seaside). Then he exported Pier. (In all cases, he had to write a method of code - a few lines - to do this.)

They also use Fuel as a persistence strategy for Pier. He saved, then deleted the file, then restarted image to show the error of not finding the file. He then got the debugger on this error (by `allInstances first`) and serialized it into Fuel. Then he loaded it into a new image and sent it open. He walked the stack, browsing state.

Camillo uses this to serialize failures in Jenkins. Now, every test failure is written to a Fuel file. You can load a zip of the test image, and the serialized failure, thus open the image with the failure displayed in debugger. Then Sean de Negris said, “Let’s build a UI for this”, so they did. He demoed the tool, just one class called TestReviver from which they can open a debugger. Some things do not work - they only serialize a piece of the stack, from the start of the test to the raise of the failure.

Q(Nick Ager) Customer could use this to serialize a bug and send it to a support group or a colleague? They would also need to serialize any classes referenced in the stack. Space in google code is limited.

Q(Carsten) There was a CampSmalltalk project StateReplicationProtocol? SRP’s goal was to be really portable between dialects. The goal of Fuel was to be fast. The Fuel team looked at SRP and several other frameworks including some not in Pharo. Fuel was ported to newspeak in 2-3 days. Felix Madrid is porting to VisualWorks and most of it is working.

Seamless, Nick Papoylias
(The projector suddenly died - not ideal for a talk with many demos. It came back in time and we saw the scary graphic ‘Die, Socket, Die!’)

Nicka argued one should not use low-level abstractions like sockets. You can pretend a socket is an object but it isn’t.

Seamless is a framework for remote programming / network programming. He opened two images ‘local’ and ‘remote’ to demo it. One image did an (other image’s) transcript show and the other’s transcript showed it. He then played ping-pong, demoing mutual recursion between the images. Decrementing 100 over the ping-pong gave even numbers on one side and odd on the other. The execution stack is distributed - first call with 100 does not return until will reach 0. Next he decremented in each image with a block that the other image provided.

He opened a trivial contacts editor, viewed it in the other machine and turned on logging to show the messages being exchanged as he added a fresh contact in one machine, then edited it in the other. “Every time you give your model directly to the UI, you make Smalltalk cry.” B+ trees can be distributed just like collections (he demoed).

One use is load balancing. Another is data balancing (talk time limits ment he skipped demos). In a specific application using Seamless, no doubt data integrity would be looked at - only share these objects allowing these methods.

He has examples of remote browsing images via Nautilus and remote debugging. These are not yet fully stable. The alternative is to make meta-objects distribution-aware, which he is now working on (in a project currently called Mercury - he opened a demo of its current state): when your reflection is distribution-aware, you can program remotely.

When the line between local and remote becomes fuzzy, seamless becomes 'shameless' with quantum objects that do not know what image they belong to. He demoed an example QuantumObject with one instvar and a method. He bound the varname to another environment and showed compiling it generated response on the other side.

Alan Kay: "On of the mistakes we made years ago is that we made objects too small." He created an object that compiled 666 and the 42 in a method. The method returns 42, the second compilation. When seamless is used, both 666 and 42 are returned. Thus you can build an object whose parts are in each.

Q(Georg Heeg, David Chisnall and others). This seems not new. The three big problems are concurrency, latency and distributed failure. Georg said very similar work was shown in 1993. David Chisnall and others said much the same.

Smalltalk Past and Future

Smalltalk over 31 years, Martin McClure, VMWare

This talk about the history of Smalltalk will also involve Martin's personal history. In 1968, for his 12th birthday, Martin got a book about how you built a computer from unbent paper clips and similar stuff. His mother attended a course and was told "Type R U N" so she typed "Are You In".

In 1974, Martin read a two-sided book: one side was Computer Lib: you must and shall understand computers. The other side of the book was called Dream machines - new freedoms through computer screens - sort of a preview of Apple, with quotes that sound like Alan Kay, etc. In the book's addendum the author mentioned his visit to ParcPlace and actually mentioned (in one sentence) Smalltalk.

Personal Dynamic Media was an article by Alan Kay and Adele Goldberg in 1977; it spoke of 2D graphics, 3D graphics, music synthesis, and it also mentioned Smalltalk - and showed a listing of Smalltalk-72 code. Martin saw there was a ref to the Smalltalk-72 Instruction manual. He called the publicity department, was told it was withdrawn, and 15 seconds later was talking to Adele Goldberg - who told him they'd publish something soon.

Alan and Adele got permission to do public stuff in 3 phases. The first was the Byte article. The second was a book. The third was the code.

Byte in 1981 had 13 (of 15) articles on Smalltalk in that issue. He showed some of the ads to give a feel for what computing was like in 1981: $3339 (7000 euros today) for a 10Mb disk. C was almost not used - 2 adds for C, more for commercial Forth systems.

The editorial explains what a mouse is. A diagram tries to explain the difference between 1981 development and Smalltalk development. One article explains that people who know nothing about computing find OO natural whereas those who know (knew in 1981) computers find it strange. Another article explains text selection.

“Type checking is important in most systems for four reasons, none of which is important in Smalltalk”

“More than 20 years experience shows us that bad system design cannot be hidden from the user even by a masterfully-crafted user interface.”

Dan Ingalls showed bitblt. Dan also wrote an article on the design principles behind Smalltalk. A system must be understandable within a single mind. Objects. Automatic storage management - code sprinked with storage management is not easy for humans to follow. Design around a uniform metaphor applicable to all areas. Modularity (would later be called encapsulation). Classification (obvious remarks and) new classes must be on an equal footing with the kernel classes of the system (and vice versa i.e. you can change the kernel). Polomorphism. Factoring (20 years later, this point would be rechristened by Kent Beck as “Say it once and once only”). When a system is well-factored, users get leverage. Dan included having a Virtual Machine specification in his list (Martin agrees that the well-defined interface spec is valuable). The Reactive principle: every component accessible to the user can present itself for meaningful inspection and manipulation - i.e. the inspectors, workspaces, etc.

Dan argued against having an operating system - the tradition continued by SqueakNOS but least achieved in general.

Natural Selection is Dan’s final principle. We have since then seen language fads. A language that is technically better, so needing fewer developers and managers, can for that very reason get outvoted when groups merge, etc. However we are still here talking about Smalltalk.

Peter Deutsch wrote on building control structures in Smalltalk. He presented the block. As an example, he built a case statement. He then impemented comma on block context to show just how easy it was to make such changes. He also showed exiting from a loop while remaining within the method. Martin tried the code example in GemStone and it works - but we are quite happy without this feature so he’ll leave it that way. Finally, he explains that Smalltalk enables this because it has blocks, can access control state directly, and because anything in the system can be changed.

Article: “Is Smalltalk-80 for children?” using a program that looks a little like EToys. Another article showed a drawing tool more useable by artists

than typical computer tools of the time.

Finally Ted Kaeler wrote about memory management: main memory 50-199k bytes, disc 5-10 million bytes.

Elsewhere in that issue of Byte, a rumour that IBM planned to release a small computer, and that IEEE would publish some standards.

In 1982 GemStone was founded. Monty was there at the start and is here in the audience. In 1983 Martin got the blue book, the green book and in 1984 the orange book. He read the books but still couldn't run Smalltalk till 1985 when Apple released Apple Smalltalk, later the basis for Squeak. He sent $50 for the stack of floppy disks. At the first meeting (in his house) of the user group that he'd founded (all 5 of them), the disks hadn't come yet, so at next month's meeting he was the expert because he'd been using it for 3 weeks.

His first major career decision was whether to work in hardware or software. Using Smalltalk was the most fun so in 1994 he started in Smalltalk and today he's at GemStone, still programming in Smalltalk, still having the time of his life.

Q(Georg) Two extras: another issue of Byte is 3 years older and it introduced Pascal. That issue had the ivory tower of Smalltalk - "the craggy aloofness of Smalltalk" (shown as in the middle of the Fortran ocean).

Next year, we have the 30th anniversary of the first external computing use of Smalltalk (via another Xerox division who produced computers for one big customer, who still exists). Mark Roberts, documentor at Cincom, is one of the kids who learned Smalltalk in the 70s.

James Foster read an article by someone who, as a child, spent 3 or 4 years visiting the Xerox Parc office from junior high school to do Smalltalk.

Smalltalk in a C World, David Chisnall, University of Cambridge
Etoile wants to reinvent everything, but reuse some existing code. No application should need more than 1000 lines of application code. (Number of bugs per line of code is a constant.)

Smalltalk has moved up from 70 to 39 in the TIOBE rank of languages - essentially a measure of how many people are hiring, so it indicates how much existing code there will be in that language. C is top of the list, with Java 2nd (lots of code - every object needs a factory) C++ 4th, ObjectiveC 3rd but with much less public-domain code (partly because each line achieves more).

For example, sorting unicode strings in a locale-aware way (i.e. to be right for French, German and Spansh) is really tricky and really boring. When it has been done right, you don't want to do it again.

The Smalltalk goals of Etoile are no VM, just native code, with both JIT

and static rcompilation (e.g. change method on class at runtime, replace static with JIT recompilation of method, etc.). They also want to interoperate with ObjectiveC (and C and C++), and easy embedding in existing systems. Smalltalk performance must appear OK or people will stupidly avoid using it. And they want automatic persistence.

100% compatibility with Smalltalk-80 class library is not a goal. There is vastly more experience in the world in Apple libraries, even if Smalltalk were 100-times more productive.

Objective-C is their foreign function interface. He showed the most convoluted hello world program he's ever seen. It was ObjectiveC (actually ObjectiveC++). He showed calling C's sqrt in such programs.

He showed the architecture: see his slide. The OMeta parser is almost finished and will replace the C parser. At the top, Smalltalk Apps rest on a couple of Smalltak support libraries that parallel the ObjectveC apps and top-level libraries. They use immediates like Smalltalk. They recently added 7-character ASCII strings to the immediates on 64-bit and it turns out they are everywhere (15000 created in a program that takes 0.5secs - XML parsing uses them a lot, etc.)

Lastly, hardware (Alan Kay quote). He is working with a hardware group (funded by DARPA) that will soon release a 64bit MIPS softcore platform that provides hardware-assisted GC. Because David works in an academic group that also does hardware, when he wants new instructions he just asks and in a week or two they're there.

They want to have a platform where they can start with Smalltalk which is productive. As we head into multi-core and distributed, Smalltalk may start showing its age and then new languages may be added.

Q. Reuse: what code is worth reusing? ("Good question") Some code solves the general case and you need it for a particular case. Other code starts specialised and you discover over time that it has wider applications.

Q(Stephane) NativeBoost? Good example of not reusing. The LLVM register allocator is 30,000 loc - maybe 150,000 with all optimisations. If the goal is to have a flexible experimental platform, NativeBoost is good. If the goal is to run fast, then if Smalltalk is 100-times more productive than its rivals, that still translates to many years of effort.

Show us your project(s) in 10 minutes maximum

At the end of the second session, Stephane remarked that we are getting good at staying within the 10 minutes.

New ObjectStudio UI Framework, Andreas Hiltner

ObjectStudio comes with DLLs. When debugging reaches one of these, magic happens on the screen, not the ideal Smalltalk debug experience. The new UI framework project will be all Smalltalk, no DLLs. People in financials and etc. do not like receiving bugfixes that are DLLs - are they

safe? A five-line change of Smalltalk code is much easier to review for safety at a customer site than a 900k DLL. Clients can also change the code to their liking. It will also be a lot easier to add new widgets.

Some of this will be in the next major release. They now use the standard C API for CommonControls (the progress bar and the marquee progress bar), and the font cache. Menus are now more customisable.

Andreas demoed opening a simple window with a button, while having the transcript show all the events being received by the window button, indicating the degree of control available in the framework. All this is made visible via announcements and you subscribe and thus get reaction to UI.

They now have an ipAddressControl widget where you can limit the range, set the net mask and so on. Likewise he showed a new listbox, nothing special but ObjectStudio now picks up the theme easily. While demoing a password editor, he changed the size dynamically, trivial in Smalltalk but with a C widget you would need to recompile.

smalltalkedVisuals, Christian Haider
He looked up VMWare - a publicly listed Smalltalk company (rumour has it they do other stuff as well) - in Bloomberg in the left pan of his program. (He got the Frankfurt entry first and used that.) Thus he got the data for a graph. The right font was not installed on this laptop so the look and feel was not quite what it will be in actual use.

The art director of the newspaper sets the chart layout and the tool enforces that (very important to customers) so only some things can be changed by standard user (if you recognise this layout, you know one newspaper that is a customer). User can change the scaling but Christian's algorithm for the scaling is very good and they rarely need to. You can change the duration of course. He chose a later time and saw VMWare value going up (must be after the acquired GemStone). He played with a few layout options, superimposing different-coloured other lines, etc. He displayed in PDF where of course it looks better.

ARM Jitter, Lars Wasserman, Summer of Code
ARM is a common platform for small devices and their performance is low. A JIT can make the difference between usable and not. ARM is the logical next step for Cog to use as a processor to support.

He started by getting the CogVMSimulator running. Using this, he mapped from the intermediate representation (RTL Opcodes). He worked on generating memory chunks. Finally he was able to compile.

The simulator used in GDB was the one he used, extracting the parts he needed. (It is an old simulator, last commercially worked on in 1994). He created a plugin that disassembles libopcodes and then the ARMEmulator could execute them.

The compiler generates most of the bytecodes correctly (all correctly, and

getting it running, is to do). So he needs to convert/check his cogit changes to be SLANG. He needs to check the organisation of constants. Three instructions load a word and a fourth handles the whole word. He does not have a Mac so has not built the plugin on Mac yet, just Windows and Linux.

Concrete Type Inference, Santiago Bragagnolo, Summer of Code
He worked on symbolic (non-run-time) type inference. He selected some expressions, inferring the return types of the returned object and its instvars. It returns ? when there is a type which it cannot infer.

The core class is called KwisatzHaderach (because type inference sort of predicts the future actual execution :-)).

```
KwisatzHaderach callGraphFor: [...code...]
```

looks at the byte codes that execute and thence does the inference.

Q. Blocks? Each block is its own type.

Q. Symbolic execution? No, just type inference.

Improving the Debugger, Karsten Kuche, Heeg
Karsten created a seaside application that renders a diff for 100 levels of rendering (gives a pretty pattern). He breakpointed `renderLevel:on:`. His debugger displays overridden methods in red and emboldens your break context to make it easier to get back there.

You cannot scroll down if you have 1 million frames: if you try, you scroll down doubling the stack each time until it crashes through reifying all those intermediate frames you don't care about. Now he can jump 1 million frames instead of the impossible scrolling.

From an inspector he can open an 'inquisitor'. A dialog asks, 'What do you want to observe?'. Karsten inspected a window and chose to observe its height. He then resized the window and saw his inquisitor observing its changing height. This is great for finding when a damaged rectangle has a certain value while debugging.

Suppose you want to debug `at:put:` for a certain dictionary only. A breakpoint popup lets you choose that one dict (from `allInstances`) and he showed that the debugger breaking in that dict but not in another.

Karsten then presented his Cocoa adaptor. He showed the XCode outlets and inputs for a widget. He wrote standard UI methods in VisualWorks with <method> pragma and another pragma giving the return type, since ObjectiveC needs that (e.g. <returnType: #void>) Thus he can create applications with native Mac UI. He showed another Cocoa widget that drew a gradient and also had a text pane where you could evaluate expressions to change it (e.g. rotate it).

Roassal in Amber, Vanessa Pena
Roassal has what Mondrian had but with more freestyle application. She

showed filling a screen with coloured balloons. Similar to `inspect`, the method `visualize` opens a Roassal screen on an object. The resulting graph was a bit like the object explorer but with better colour, more dynamic with showing data when clicked on and etc.

Roassal now works in Amber. It's a bit slower but the examples she showed were usable. She showed the collection hierarchy analysis example.

Rizel: multi-dimensional Profiling, Juan Pablo Sandoval
Rizel handles two dimensions: the benchmark version and the current ('silver') version. In VisualWorks, they use the ATProfiler. In Pharo, they use MessageTelly (same kind of tool).

They can see (red colour, dimensions) which methods are slower and are absolutely much called or slow. Select and open the comparison tool to see code compare. In XMLSupport, between versions 6 and 7, we can see some changed methods and two new (in yellow), and the colours and dimensions show where to look for the cause of the slowdown.

Zinc Websockets, Sven Van Caekenberghe
Zinc HTTP Components is a framework Sven wrote a while ago, on top of Zodiac because they needed security. Zinc WebSockets makes it possible for the server to initiate communication - previously communication was always initiated by the client.

He demoed from the client side, going (in web browser) to a website WebSocket.org which explains the technology, and also answers back anything you send it.

```
(ZincWebSocket to: 'http://echo.websockets.org/')
  sendMessage: ...
```

and he got his message back. He showed the Javascript of the WebSocket site's echo behaviour, then implemented it in Smalltalk. On the server side, `runWith:` runs in a loop and responds when anything is sent.

This week, people have worked with this and now have Amber Smalltalk speaking to Pharo Smalltalk.

SourceCode TourGuide, Hwajong O
Hwajong O is from South Korea. The system browser shows everything by name. It is like a dictionary, not a book. It is good if you already know the name, less so otherwise. His project is to make the browser more like a tour guide who tells a story and helps you find things.

He opened the tour guide, then drag-dropped various classes, methods, etc., into the tour guide's list. He finds this useful for e.g. bookmarking the methods involved in a Metacello baseline, for use when he wants to create a new one. Other uses: help new team members, conference presentations.

Q(Stefan) You can navigate in this browser? Yes, when not in the list view.

Presenty, Denis Kudryashov
He showed adding behaviour to his demo talk of yesterday using his framework. When he drew a rectangle in the drawing tool, the up arrow in the browser was activated, so everytime his drawing was enough like a rectangle, it selected the class above.

PHANtom, Johan Fabry, Daniel Faidames, Universidad de Chile
A modern aspect language for Pharo (probably also works in Squeak). Aspect. e.g. logging, represents a cross-cutting concern. An aspect contains behaviour - what will it do - and quantification - where the pointcuts are.

When a method is being executed, that is passed to all the predicate WHEN classes. Those that are interested then run their WHAT behaviour.

```
PhPointcut receivers: TestCase selectors: 'assert'

PhPointcut receivers: TestCase
  selectors: HASH(assert 'assert:description')
  context: HASH(receiver selector argument)

PhAdvice after/around ...
```

`PhAdvice` is a class you can install or uninstall. When it is active, it's advice will run whenever its pointcut is matched.

Q(Martin) performance? It's slow, because method wrappers intercept the points of interest. A student is looking at recompiling stuff to improve the performance.

Q. Compare to AspectS? They derive from it but have added more features. Lets you specify in a more smalltalk way.

DeltaWerken, Stefan Eggermont, Diego Lont
DeltaWerken is a framework they use to create Seaside applications. Storyboard is their demo application - a planning application, showing iterations and stories, coloured showing status. They added a task (to make a screencast of this talk). They clicked and added the documentation task. They dragged it around in the list.

They showed the code framework. It's based on Magritte's approach but with some changes. They have users and announcements and code to wire it together. A range of Seaside components can be used to display e.g. a Story. They have drag-drop, and support multiple views on same object. The framework is exercised on Pharo 1.3 and 1.4, and persisted in GemStone. It should work on other dialects - any dialect with Seaside.

ASN1 Parser, Norbert Hartl, 2denker Gmbh
Norbert builds stuff in Smalltalk to make people want it. Abstract Syntax Notation 1 describes how data is encoded transmitted and decoded over the wire. It's used in SNMP, LDAP, etc. He has a contract from Iceland Telecoms to help do a GSM network (http://www.on-waves.com).

The BNF is 384 production rules. PetitParser handles this one-one. It

creates a runtime model, no classes are compiled (at the moment, because it is fast enough). 90% of the rules are translated; he omitted XML as he knows no use case for it.

He demoed by connecting to esug for registration and checking the certificate. Via openssl, he downloaded the certificate and checked it. Using various protocols, he can get data over the wire and parse it.

It will be open-sourced within 2 months, on on-waves.

Robotics with Smalltalk, Noury Bourakadi, Ecole de Mines
Several projects in this domain are being done in this area. Arduino drives robots from Pharo. This has been upgraded to run over a network. He showed an amateur video. There is a project to use robots to explore an area and map it. A related project estimates how efficient the search is for given numbers of robots (he showed another video).

URBi ST drives a human-shaped robot with middleware called URBi. Robot follows red box moved on screen (video).

Next week, the ROST project will work to provide a Smalltalk front-end to ROS middleware and so explore use of robots in shopping malls.

File versions in Store, Carsten Harle, straightec
Datenzaentrale's large application wants to store files in Store. Files should be handled the same as methods, etc. File timestamps must be preserved. File deletion must work.

They have 9,000 classes, 100,000 methods and 9,000 files: 6400 are Cobol files and the rest are icons, HTML and Smalltalk server pages.

Basic file support is in Store but it has almost no UI. Carsten made it preserve the file timestamp. He implemented loading of deletions. He made merge work including files. He could not change Store so ensured backward compatibility (by ingenious exploitation of a bossed objects ability to hold another object).

His add-ons give Store extra menu items: compare files, add files, etc. (He can also lock files or bundles if you want to avoid optimistic merging. This supports some processes at Datenzaentrale.) The merge tool can handle 3-way merges of files and shows a useful browser. There is a file version browser, like a method browser.

In the second part of his talk, Carsten presented his merge alogithm tweaks. He can merge files and his algorithm is faster and avoids spurious matches in complex bundle-subpundle cases. The use case is a trunk, plus branch with 2 methods and one files change a bundle with complex structure and 216 packages. Carsten's 'Merge with Base' approach allows the tool, with the user's help, to discover it need only regard changes in 3 items instead of 166 items and 83 conflicts, with gain in speed and clarity.

Virtual Plat machines, Vitali
The domain is Life Sciences. Many tools must be integrated: info on soil, water, environment, modelling, etc.,. from Fortran, MatLab, etc. The virtual plant models how the genotype maps to the phenotype. After 10.5 months of work they delivered at the time required and sent the invoices the day after.

SUnit: pluggable Suites and Results, Niall Ross
Using SUnit, we all subclass TestCase. I showed patterns for subclassing TestSuite and TestResult. These will be available in the open repository and elsewhere. Feedback on these is welcome to see which of many possible patterns will be most useful.

The standard SUnit TestSuite runs tests in the same order every time. I showed a subclass that ran them in a random order that is remembered (in case there are errors to debug, revealed by that reordering).

When tests developed by different people are grouped together, e.g. in a product suite, their resources can conflict. For example, perhaps the system can only connect to one database at a time, and the overall suite finds itself with two resources that connect to two different databases.

- Some testers will be willing to have a slower run rather than see a non-significant failure. For example, batch testers, whose suite is run overnight or over the weekend, have plenty of time and may be unable to alter tests supplied to them.
- Others may have the opposite priorities. A developer who is running a integration test suite may want to see a clashing resource error (so they can tag their new clashing resource as in a conflict set, needing distinct setUp/tearDown). If the suite is that development group's one-at-a-time bottleneck for doing integration, the alternative of making the suite run slower may be very unacceptable.

I demoed how different subclasses could be made to run tests

- either optimistically: assume resources do not conflict unless they are tagged as in a conflict set
- or pessimistically: assume any two resources in the suite may conflict if they are not jointly used by some test.

trading speed of execution against the risk of unmarked resource conflicts.

Lastly, I showed holding the results and times of the last run of a test in the image, keyed against that test's compiled method so that changing or unloading a test flushes it from the cache. This plugs into the RB display.

Q(Stephane) provide setter for subclassed suite and/or result in the framework, not in specific tools? That is preferable but may be tricky.

Q(Stephane) Would another use be for remote debugging? It's an idea.

Import Business Objects With Automatic Test Generation, Martin UnterHolzner & Martino Trosi, Lifeware

Lifeware have a web-based app to manage insurance contracts. They won a contract to retire a rival system (that used Oracle for its persistence) on Friday evening and restart in their VisualWorks system (that uses GemStone for its persistence) on Monday morning. Thus they needed to get events from Oracle on Friday and replay them on their system.

They had to test this. They copied data from the relational database and generated the tests automatically from the events. He demoed, introducing a bug and showing the test now failing.

Pier 3, Nick Ager

Pier runs on Magritte 3 which is Magritte 2 new and improved: it drops naming conventions that clashed, drops class-side descriptions that caused validation issues, and uses pragmas. Nick has written a refactoring that maps Magritte 2 class-side descriptions to Magritte 3 instance-side descriptions. He has produced a video on Magritte 3. See twitterboosting.seasidehosting.st for the example he showed of what Magritte 3 can look like.

Pier 3 is a base Pier that uses Magritte 3. Pier WYSYSWIG lets you edit Pier markup in a Pier editor. Pier Admin and Pier Setup assist managing Pier. He went to Pier Admin Setup. It offers some predefined templates: the blog template, the book template and others. He chose the book template, created a user, added sample content. It created an instance of the book and asked if he also wanted a book-admin instance to administer the book.

A Pier kernel is an object graph. In Pier 2 and Pier 1, the template would have to include the stuff for managing it and the css could often confuse this, but in Pier 3 these are separate. One of the really nice features is the ability to pull in code for your image, using Shout to colourise, using the same syntax highlighter. This could progress to a self-documenting system that pulled in changed code.

He edited some book text using the usual WYSYWIG editor, which works by putting Pier markup into the basic text.

Next he created another instance from the blog template, and imported a blog. He took Mariano's wordpress blog. ("There he is in the audience, wearing the same shirt" :-)) The blog was promptly there in Pier format, having done good stripping, so there was no nasty HTML in the blog.

A problem he has met is importing from old Pier.

Other Discussions

Hans Martin-Mostner of Heeg is looking at Raspberry Pi. It can't run Ubuntu (it's too small / simple) but it is really cheap: $35 for the B model and even cheaper for the A model. (There is no memory on it: everything is plugged in from USB.) Squeak runs on it but is slow. Scratch has been made to run (sort of) by a student but is really slow. Work is progressing on

a better Scratch implementation on it.

The Lifeware guys showed me a refactoring tweak they thought might be good for the refactoring framework base: adding data to the context of a refactoring to assist refactorings related to their internationalisation work. They would also like a 'safe remove' that removes all implementers of an unsent message. When a class is removed, any subclasses should be (offered to be removed or) reparented to the superclass. Fede of Lifeware also sat with Martin to review socket issues. Lifeware are looking at using Polycephally to enhance their remote testing framework.

Felix Madrid is working at Abit in Dusseldorf (formerly he worked at Smalltalk/X). He ported an ObjectStudio system from v7 to v8. Now he is working on a Seaside app.

Hwajong O is one of a small South Korean Smalltalk community (~ 7 people who meet every 3-4 months). They want to find a Smalltalk project task external to the group to increase visibility. Stefan suggested a data conversion project as a low risk project to seek out. Hwajong knows of a WhiteHawk project running on old (2.5) VisualWorks (missile tracking system, Israeli in origin: it teaches the missile to track the target, running on an Onyx 10000 operating system).

Roel Wuyts is now working on a hyper-performance computing project (no Smalltalk involved, alas). His Smalltalk students at VUB are finishing so he visited the Camp Smalltalk to see what was on and whether any more Smalltalk involvement was possible.

Conclusions

It is much more relaxing to attend an ESUG when you are not the organiser (I organised last year's conference in Edinburgh :-)). And it's pleasant to meet a range of commercial Smalltalk users and have several of them stand up and present what Smalltalk can do.

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

---

* End of Document *
