The Association of C & C++ Users

accu

SPRING CONFERENCE 2003

Generated by: Reportlab Europe Ltd

Welcome Letter

Dear ACCU Attendee,

We are delighted that you have joined us at **The ACCU Spring Conference 2003 incorporating the Python UK Conference, April 2 – 5, 2003**, at **Holiday Inn, Oxford**, Peartree Roundabout.

The ACCU Spring Conference consists of **57 hard-core sessions** spread over **FOUR** full days. Topics ranging from "*C++ Threading*" to "*Advanced Template Techniques*" to "*Concurrent Programming in Java*". All the sessions will be presented by top experts in their field of specialisation. The Conference will offer you a unique opportunity to master all the latest tips and techniques to help you programme faster and more efficiently, as well as providing you with in-depth information on technical issues and future directions.

There will also be **4 spectacular keynote sessions**, one each day from 09.00 – 10.00. These are:

- "Why design another programming language" by Guido van Rossum
- "In the Spirit of C" by Greg Colvin
- "The Cost of C & C++ Compatibility" by Andrew Koenig
- "The Internet, Software and Computers – A Report Card" by Alan Lenton

We are very lucky to have such eminent speakers and these sessions are not to be missed.

There will also be **lunchtime presentations from our Exhibitors**. We have a tentative timetable set but please ask at the Registration Desk for confirmed times each day.

This year there will be plenty of opportunity to get together **after-hours** with **Birds-of-a Feather sessions**, as well as the regular **Blackwell's reception** on Thursday evening and **Speaker Dinner** on Friday evening.

As an attendee of the ACCU Spring Conference, you'll get source code, objects, and sample applications on the conference CD-ROM which you will find in your delegate bag.

Have a wonderful conference.

There's no better way to learn C and C + + than from the source. We look forward to seeing you there.

Sincerely,

Susan Bennett

Talks and Speakers Details

Keynote

In the Spirit of C [Keynote], Greg Colvin
day: 0, time: 9:00

C++ and Java owe much to their C heritage, and the usefulness of C has only been increased by the success of its descendants. These languages serve a community of related interests, and in my opinion need both constraints on compatibility and room to grow in order to continue to be of service. The most important maxim of the Spirit of C is "Trust the Programmer." That maxim imposes a corollary obligation on us as designers and standardizers to provide the programmer with trustworthy and effective tools for the job at hand. We will explore where and how these languages are effective today, and what we need to do to maintain that effectiveness into the future.

I Dr. Colvin learned to program in 1972, on a PDP-8. He has been coding C and doing object oriented design since 1984, C++ and generic design since 1988, has served on WG21 since 1990, is a founding Boost member, and has recently joined WG 14. He has worked in the Oracle JVM group since 2000, and is Oracle's C/C++ liaison for ISO. He is also a dedicated blues musician, recording engineer, Tennessee Walking Horse breeder, and antiwar activist in what little spare time remains.

Why design another programming language? [Keynote], Guido van Rossum
day: 1, time: 9:00

The creator of Python muses on the forces that influenced Python's design, and what lies ahead.

Guido van Rossum created Python, a major open source programming language, in the early 1990s at CWI in Amsterdam. He is still actively involved in Python's development. In 1995 he moved to the United States, where he now lives and works in Reston, Virginia for Zope Corporation -- developers of Zope, the leading open source content management system (written in Python, of course).

The Cost of C & C++ Compatibility [Keynote], Andy Koenig
day: 2, time: 9:00

This was first presented as a technical session at the Autumn meeting of WG21 and J16 (C++ Standards Committees). During the talk I will consider various aspects of maintaining backward compatibility within languages whose history extends back almost 30 years. I will be examining the way in which hardware development questions many of the assumptions that are made in language design and use. The main purpose of this keynote is to challenge your view of the C family of languages and ask you to consider how we should move forward.

Andrew Koenig is a member of the Communication Software Research Department at AT&T Shannon Laboratory, which was once part of Bell Laboratories. He has been working mostly on C++ since 1986, and is the C++ standards committee's project editor. He joined Bell Labs in 1977 from Columbia University. Aside from C++, his work has included programming language design and implementation, security, automatic software distribution, online transaction processing, and computer chess. He is the author or coauthor of more than 150 articles and the book `C Traps and Pitfalls,' and coauthor of the books `Ruminations on C++' and `Accelerated C++.' He has taught courses at Columbia and Princeton Universities and Stevens Institute of Technology, given tutorials for Usenix Association, Stanford University, Boston University, Lund Institute of Technology, ACCU, SIGS Conferences, the Federal Open Systems Conference Board, and the Federal Reserve Bank, and given invited talks for IBM, Syracuse University, ACM, IEEE, Miller-Freeman, and AT&T in Tokyo, and a keynote talk for the 10th International Python conference.

The Internet, Software and Computers – A Report Card [Keynote], Alan Lenton
day: 4, time: 9:00

The demise of the 'dot com' companies, the 'war' on terror, the assault on peer-to-peer technologies, Microsoft's Paladium initiative, the operation of the patent laws and changes to copyright laws have a great deal of potential to change the face of computing as we know it. Desk top computers are a relatively recent innovation, and they are viewed by many powerful grouping in society as a threat to their interests. This talk looks at what is happening as they try to defend their interests and how it will impact those who work with computers.

Alan Lenton is in charge of the design, development and programming of multi-player games for Interactive Broadcasting Ltd. He also handles technical matters relating to the delivery of the games over the Internet.

Multi-player games designed include Federation, an adventure/economic simulation set in a future universe; Iron Wolves, a submarine simulation; Age of Adventure, a role-playing game based in Victorian times, now in beta-test; and Barbarossa, a strategy wargame based on the German invasion of Russia in 1941, now in alpha-test.

He lives in London, and as well as his work for Interactive Broadcasting, he is a member of the BSI's C++ panel and a long standing member of ACCU.

Alan produces a weekly newsletter covering issues affecting the Internet and computing generally. Web site: http://www.ibgames.net/alan

Python UK Conference

Extreme Programming in Python, Chris Withers
day: 1, time: 10:30

This talk will introduce the basic principles of Extreme Programming, and explain Python's support for unit testing, including "unittest" and associated tools. It will draw on NIP's real world experiences of using this methodology in production.

In addition time will be provided for anyone else working with XP to recount and discuss their experiences and demonstrate associated testing tools.

Chris Withers has been developing Python and Zope applications for New Information Paradigms (http://www.nipltd.com) for the last three years. He is the maintainer of the Squishdot project as well as doing core development work on both Zope and its Content Management Framework. His areas of special interest include unit testing and content management systems.

Siena Web Service Architecture, Marc-André Lemburg
day: 1, time: 10:30

In today's business world, Web Services authoring has become a synonym for having to write JavaBeans-based Java code for deployment on J2EE compatible application servers. This talk will demonstrate an alternative solution which is built around Python with the aim of making service writing easy, flexible and productive.

CORBA? Isn't that obsolete?, Duncan Grisby
day: 1, time: 2:00

If you keep up to date with the mainstream IT media, you could be forgiven for thinking that CORBA was obsolete, and thoroughly dead. In fact, it is alive and well, and is the best solution to a wide range of distributed application problems. This talk introduces the features of the CORBA object model and services, and discusses when it is most sensible to use CORBA, in comparison to other technologies. The talk is based around using CORBA from Python, but it also shows how it can be used to build robust distributed systems with a mix of programming languages, including Python, C++ and Java.

Duncan is an experienced software architect and developer specialising in distributed and concurrent systems. He is the lead developer of omniORB, a high performance open source CORBA implementation. From 1999 until its closure in April 2002, Duncan was a researcher at AT&T Laboratories Cambridge, Europe's leading computing and communications research laboratory. Since leaving AT&T he has been working as an independent contractor on a number of large-scale distributed systems integration projects. Duncan holds MA and PhD degrees from Cambridge University, England.

Python Design Patterns, Duncan Booth
day: 1, time: 2:00

What design patterns are applicable to Python? Some patterns are an intrinsic part of Python, other patterns require some careful coding to get the best from them. What new patterns appear in Python?

This slot will also include time for anyone wishing to discuss, explain or demonstrate their own patterns.

Duncan graduated as a Computer Scientist from Cambridge University and initially worked for Torch Computers using BCPL, C and Assembler before joining the fledging RCP as the third employee in 1984. Duncan has worked on many projects in RCP, including foreign exchange dealing, real-time financial data, banking back office systems, and healthcare systems. His interest in Python was first sparked at a science fiction convention, since when it has become a major player in his choice of languages. As well as writing in Python he also ported it to the Psion 5 handheld. He is currently working with a variety of internet based systems including Python and Zope.

Parsing made easier - a radical old idea, Andy Koenig
day: 1, time: 4:00

As powerful as Python regular expressions may be, there are some things they just can't handle, such as nested data structures or sequences of symbols that aren't characters. Snobol4 solved some of these problems 35 years ago, but the language never caught on. In the Python world, the SnoPy project offers a gateway to a Snobol-style pattern matcher that is part of GNAT, the Gnu Ada Translator, but not every platform has GNAT available.

This talk describes a pure Python library that captures most of Snobol's pattern-matching capabilities. Moreover, it lets you do some interesting things that Snobol can't do, such as

- using regular expressions as pattern elements,
- making it unnecessary to imitate them;
- specifying how to put strings back together after the patterns have taken them apart; and
- storing intermediate data in safer places than global variables.

This library is a work in progress, so the exact contents of the talk will depend on how much progress I've made by then. However, I hope to be able to show some examples that are entertaining, and perhaps even useful.

RESTful Python, Hamish Lawson
day: 1, time: 4:00

Web Services are all the buzz just now - and for many people, Web Services means SOAP and other new protocols. But a number of notables from the Web and XML worlds are concerned that, in adopting an essentially RPC model rather than a resource-based one, SOAP discounts the architectural lessons that have been learned about what made the Web successful - so making it harder for SOAP-based Web Services to achieve the ubiquity enjoyed by the Web itself. Instead it is possible to deploy Web Services that do follow the Web's architectural principles and that use the established Web protocols we already know. These architectural principles have been captured under the name Representational State Transfer, or REST. In the first half of my talk I will discuss what these REST principles are and how they help in creating maintainable and interoperable web spaces. In the second half I will survey the support for REST provided by various Python web application frameworks, in particular Quixote.

Hamish Lawson is a software developer in the IT Services department at the University of St Andrews, Scotland. He specializes in web-based enterprise information systems.

Introduction to Python and Jython for C++ and Java Programmers, Alex Martelli
day: 2, time: 10:30

A general 90-minutes tutorial about Python and Jython for experienced C++ and Java programmers attending the rest of the ACCU conference

Alex Martelli lives in Italy and is a Senior System Developer with AB Strakt in Sweden, mostly writing Business Logic *infrastructure and modules for Strakt's CAPS Real-Time Enterprise Computing platform. Alex co-edited the "Python Cookbook", and wrote "Python in a Nutshell", both for O'Reilly. Alex is a popular poster to comp.lang.python, the winner of the "Python Activators' Choice 2002" award, a member of the Python Software Foundation, and a board member of the Python Business Forum.*

The Infinite Filing Cabinet - object storage in Python, Jacob Hallé
day: 2, time: 10:30

Editor's note - within a 90 minute slot we intend to provide a general survey of Object Storage techniques in Python. This will hopefully include a full talk on ZODB (speaker volunteers welcome) for use outside of Zope, as well as the following talk which will last 30-45 minutes:

The workhorse of the CAPS system is a database and supporting software which is called the Infinite Filing Cabinet, the IFC. Information is organised as objects and the IFC, written in Python, is the persistent storage of these objects.Objects can be searched for, retrieved and dynamic notification of object creation or modification can be requested. More complex actions can also be performed when objects transit specified states at optionally specified times. The IFC does not support the deletion of data; this is performed by deprecation. It is therefore possible to access any object in any of its prior forms.

Building GUI Applications with PythonCard and PyCrust, Andy Todd
day: 2, time: 2:00

In this session, we will demonstrate the process of building full-blown Python applications with world-class graphical user interfaces. By giving you access to the full power of Python without requiring you to master complex GUI libraries, PythonCard becomes a power tool for scripting users and professional programmers alike.

Creating graphical desktop applications that run across platforms and take advantage of Python's simple elegance is easier than you think. In addition PyCrust, the embeddable Python shell, puts the power of the Python interpreter at your fingertips. That means every GUI widget in your application can be inspected and manipulated from the PyCrust shell while your application is running. The combination of PyCrust and PythonCard brings the power and flexibility of Python to the realm of cross-platform GUI applications.

Andy has been involved in the PythonCard project since its inception in 2001. As one of the developers on the project he has contributed code, samples and documentation. His areas of special interest include storage, in particular relational databases, and object-relational mapping as well as agile methodologies.

Integrating Python, C and C++, Duncan Booth
day: 2, time: 2:00

Sometimes Python on its own isn't quite enough. Maybe you want to access a library available only in C or C++, maybe you want to script an existing C/C++ application, or maybe you just want an easier way to test your C/C++ code.

This talk introduces the Python C API, and also looks at more sophisticated interfaces such as SWIG, SIP and Boost.

Lightning Talks, Paul Brian
day: 2, time: 4:00

This session will consist of a number of 5 or 10 minute talks, as held to wide acclaim at recent Python events.

Paul will accept entries during the conference. And since many other topics are too short for a 90 minute talk, we may well fit more lightning talks into other slots too!

Scripting Java Applications with Jython, Anthony Eden
day: 2, time: 4:00

Java is currently one of the most powerful server-side languages available. Thanks to the fact that it is cross-platform, dynamically linked and has a very comprehensive set of standard and open source libraries, Java is a very good solution for server-side development. The main problem with developing server-side Java applications is the edit/compile/deploy portion of the development cycle which increases development time. Enter Jython. Since it is written in Java, Jython acquires all of the benefits of the Java platform without the need for a compile cycle. In this presentation I will explain how the JPublish web framework (http://www.jpublish.org/) relies on Jython as the language to glue together the view and the model of an MVC framework to reduce development time dramatically.

Anthony Eden is the President and Lead Developer at Aetrion LLC which was formed to provide development support for organizations utilizing open source projects. Anthony has been a Java developer for over 7 years and a recent Python crossover. Anthony develops numerous open source projects, including JPublish, a web publishing and application framework and the Open Business Engine, an open source workflow engine.

Track 1

Pattern Experiences in C++, Mark Radford
day: 1, time: 10:30

My first contact with patterns was when I first read the "Gang of Four" book in 1996. Six years later I revisit the GoF book and several other books about patterns on a regular basis. I find patterns a powerful tool in software development, and not just as a source of solutions to problems - seeing applications of known patterns leads to the increased confidence that, in hindsight, problems have been solved in ways already known to be effective!

This talk is about my experiences with patterns - taken from the GoF book and from other sources - over the last six years, and it falls into three parts. First a brief introduction to patterns, and an overview of those participating in the rest of the talk. Second my experiences of learning about patterns. Finally, I will present assorted experiences of implementing and using patterns in my C++ software development work.

Studied Maths and Physics at Trent Polytechnic, Nottingham, graduating in 1986. Having found the computing part of the *course more interesting embarked on a career in software development, using a variety of languages and platforms - e.g. VAX/VMS, Pick, Windows, Fortran-77, C, C++ and Java - over the last fifteen years.*

Independent since 1997, having initially specialised in Windows development using C++ and MFC, moved on to specialise more in C++ and object oriented design working on a variety of developments including both distributed and embedded systems.

Member of the BSI C++ panel since November 1998, and occasional member of the UK delegation to ISO meetings.

Inside Security Checks and Safe Exceptions, Brandon Bray
day: 1, time: 2:00

Buffer overrun attacks continue to be the leading security issue facing the software industry. In this talk, you will learn the details of specific methods attackers use to exploit buffer overrun vulnerabilities, from inserting arbitrary code to hijacking function pointers to hijacking the exception handling mechanism itself. Using Visual C++ as a concrete example, you will learn how C++ compilers can reduce or even eliminate different classes of security attacks, how the VC++ implement security checks and safe exceptions, and how the mechanisms work to make software more robust against attack.

Brandon Bray is the program manager for the Microsoft Visual C++ compiler front-end and language team. Prior to working at Microsoft, Brandon came from Cornell University where he concentrated in advanced programming language design and compiling for high performance architectures.

C++ & Multimethods, Julian Smith
day: 1, time: 4:00

I think that language support for multimethods could simplify some common programming tasks, and that the lack of support for multimethods in most programming languages has lead to inferior ad-hoc solutions being adopted and standardised, with little recognition that there can be a better way. A couple of examples are GUI event dispatching (Win32 message maps, Qt signals/slots), and internationalisation support. Multimethods also enable freedom from rigid interface definitions (for example they allow the equivalent of virtual functions to be added to any existing class); this freedom can be misused, but is sometimes very useful - consider how useful traits classes are when used to embellish existing types. Finally, I may talk about how multimethods can be used as part of a TeX-style typesetting engine, if some experiments in this area workout.

I live in Oxford in the UK, where I work as a contractor. I've worked for various companies in the last 6-7 years, did a PhD in Cognitive Psychology at Edinburgh University, and studied Physics as an undergraduate at Oxford University. I've written Cmm, a C++ language extension/translator that adds Multimethods to C++, and I wrote an article about Cmm and Multimethods in Overload, April 2001. Other info is at http://www.op59.net

C++ Threading, Kevlin Henney
day: 1, time: 4:00

A lot has been written about multithreading, C++ and multithreading in C++. There have a number of different higher-level threading APIs written and proposed. Some are influenced by object models that are not necessarily appropriate to C++'s own idioms, and some of them suffer from looking too obviously like API wrappers, in spite of their specific API independence.

This talk starts from basic principles to develop a different model for threading in C++. It is simple, idiomatic and generic, and its thinking is more obviously unchained from the view of thread objects as C API wrappers.

Kevlin Henney is an independent software development consultant, which covers the range of sins from development through mentoring and reviewing to training. The focus of his work is in programming languages, OO, CBD, UML, patterns and software architecture.

He is a regular columnist for C/C++ Users Journal, Application Development Advisor and JavaSpektrum, and a former columnist for a couple of magazines that have gone to the wall (C++ Report and Java Report). He contributes to other magazines, including Overload... and other magazines that have gone to the wall (e.g. EXE). He'd like to think that there isn't a causal connection. He is also a regular speaker at conferences in Europe and the US.

, Hillside Europe, C++ (BSI and ISO) and C (BSI).

Template metaprogramming in Haskell, Simon Peyton Jones
day: 2, time: 10:30

In this talk I will draw together two brilliant ideas: the rich but wild world of template meta-programming of C++, and the elegant garden of functional programming. Functional languages have served as a wonderful laboratory in which to explore and develop language ideas, such as polymorphic type systems, garbage collection, and so on. Template meta-programming has been such a success in C++ that it seems natural to try to transplant it into the context of a higher-order, typed functional programming language, and explore its properties there.

In this talk I will describe how we have done this for the functional programming language Haskell. I will not assume that you already know Haskell, but I hope to leave you with some idea of what Haskell is and why I love it. Our Template Haskell project is at a fairly early stage, but the strange perspective of functional programming may perhaps leave you with a different view of templates in C++.

Simon Peyton Jones is a researcher at Microsoft Research, Cambridge, where he has been since 1998. Prior to that, he spent nine years as a Professor at the University of Glasgow. His research interests are centered on the design, implementation, and application of programming languages, especially functional languages such as Haskell. He leads the team that developed and supports the open-source Glasgow Haskell Compiler (GHC). Home page: http://research.microsoft.com/~simonpj

Secrets and Pitfalls of Templates, Nicolai Josuttis
day: 2, time: 2:00

(co-presented with David Vandevoorde)

Although templates have been part of C++ for well over a decade (and available in various forms for almost as long), they still lead to misunderstanding, misuse and/or controversy. However, the effective C++ programmer should know about some important C++ template issues that are not widely known. This talk presents some of them.

Nicolai Josuttis (www.josuttis.com) is an independent systems architect, author, and consultant. He designs mid-sized and large software systems for the telecommunication, traffic, finance, and manufacturing industries. He is well known both in the C++ Community and to attendees at ACCU Conferences. He not only speaks and writes with authority about C++ (being the author of 'The C++ Standard Library') but is also an innovative presenter.

He has also written other books and articles about object-oriented software development and programming in general. He is a partner at System Bauhaus, a German group of recognized object-oriented system development experts

Metaprogramming and the Boost Metaprogramming Library, David Abrahams
day: 2, time: 4:00

The Boost C++ template metaprogramming library (MPL) is an extensible compile-time framework of algorithms, sequences and metafunction classes. The library brings together important abstractions from the generic and functional programming worlds to build a powerful and easy-to-use toolset which makes template metaprogramming practical for real-world environments. The MPL is heavily influenced by its run-time equivalent – the Standard Template Library (STL), a part of the C++ standard library. Like the STL, it defines an open conceptual and implementation framework which can serve as a foundation for future contributions in the domain. The library's fundamental concepts and idioms enable the user to focus on solutions without navigating the universe of possible ad-hoc approaches to a given metaprogramming problem, even if no actual MPL code is used. It also provides a compile-time lambda expression facility enabling arbitrary currying and composition of class templates, a feature whose runtime counterpart is often cited as missing from the STL. This talk explains the motivation, usage, design, and implementation of the MPL with examples of its real-life applications, and offers some lessons learned about C++ template metaprogramming.

David Abrahams is a founding member and moderator of Boost, and an active member of the wider open-source community. He has been an ANSI/ISO C++ committee member since 1996, when he developed a workable theory of exception-safety and an exception-safety specification for the C++ standard library. In his 14-year career he has developed applications for the desktop and embedded devices in the fields of music software, speech recognition, and circuit simulation. In 2001 He founded Boost Consulting, a company dedicated to providing professional support and development services for the Boost C++ libraries and associated tools.

Prying Eyes: Generic Observer Implementations in C++, Andrei Alexandrescu
day: 3, time: 10.30

The Observer design pattern has many ways of being applied and very different implementation favoring speed, information flow, control flow, flexibility. This talk categorizes Observer implementations and builds a policy-based implementation for generic Observers. (It also offers a sneak preview into Andrei's upcoming book, The Return of C++.)

Andrei is a world-class expert in software development using C++. In the C++ community, he is best known for his book, Modern C++ Design (Addison Wesley, 2001). Also, Andrei is a former columnist for the C++ Report, a columnist for C/C++ Users Journal, and a sought-after speaker at conferences worldwide.

After working in large-scale projects ranging from financial software on Wall Street to networking software to user interfaces, Andrei is pursuing a Ph.D. in Computer Science at University of Washington.

Reflective Metaprogramming, Daveed Vandevoorde
day: 3, time: 2:00

We're nearing one decade of template metaprogramming in C++. In that decade several metaprogramming challenges have been resolved, several others have been found unsolvable using template-based techniques, but perhaps most of all, metaprogramming has been found to be a sound tool for program development. Furthermore, reflection appears to be a very desirable facility in the metaprogramming arsenal. In this talk I will sketch a complete reflective metaprogramming language extension for C++. The ideas underlying this extension draw from C++ template metaprogramming experience and from a personal research language called Xroma.

David Vandevoorde is an engineer at the Edison Design Group, where he codeveloped the first 100% ISO-compliant C++ compiler. He is an active member of the ANSI C++ Standards Committee, and a cofounder of the newsgroup comp.lang.c++.moderated. He is the author (with Nicolai M. Josuttis) of "C++ Templates: The Complete Guide". A graduate of the Brussels Free University and the Rensselaer Polytechnic Institute, his interests include algorithm development, programming languages, and teaching. See www.vandevoorde.com.

Binding C++ to Python with the Boost Python Library, David Abrahams
day: 3, time: 4:00

In many ways, the Python language makes a perfect complement to C++: as an interpreted, dynamically-typed language it allows the programmer to quickly and compactly build large-scale applications that apply idioms from C++ generic programming. C++ and Python can be combined in several popular ways: for example C++ can be used to provide high-performance core components for Python applications, or Python can be used as a plug-in or extension language for applications written in C++. Effectively binding Python and C++ requires a deep understanding of the internals and idioms of both languages. The Boost Python library is a framework of tools which allow users to quickly expose C++ types and functions to Python by writing a kind of simple "Interface Definition Language" (IDL) directly in C++ code. By using the compile-time capabilities of C++ and the techniques of metaprogramming, Boost.Python avoids many of the problems of other wrapping systems, which must implement most of the internals of a C++ compiler to work automatically. The library also includes facilities for manipulating Python objects from C++ using a "Python-like" interface.

Multi-Platform Software Development; Lessons from the Boost libraries, Beman Dawes
day: 4, time: 10:30

Developing software in one platform or environment (operating system, compiler, company, department, whatever) and then later trying to port it to some other platform is often a recipe for disaster. Yet experience both at Boost and with industrial applications shows that certain software development practices can remove much of the risk from cross-platform development. Learn about these practices and how to apply them to real-world software development.

Although based on experience with C++ cross-platform efforts, much of the talk focuses on development practices which apply to any programming language.

Before the conference, please send a description of your cross-platform problem to <> so solutions can be discussed during the talk.

Beman is the founder of Boost (www.boost.org), the source of portable free C++ libraries and a proving ground for future C++ Standard Library additions. He has been a voting member of the ANSI/ISO C++ Standards Committee since 1992, and chaired the Library Working Group for five years. He develops portable C++ libraries for a living, and is the author of the StreetQuick geographic atlas library.

Sauce: An OO recursive descent parser; its design and implementation., Jon Jagger
day: 4, time: 2:00

This talk recounts the design, patterns in, and implementation of Sauce. Sauce is an extensible tool that you can use to detect syntactic and lexical fluff in your code. (Sauce came into being when one day I imagined a tool that could check all your binary operators were braced by matching whitespace. All the lint tools I could find were purely syntactic!) I will also discuss the differences encountered, and lessons learned, when implementing Sauce in more than one language (C++ and C#).

Hi, I'm Jon Jagger, an independent software consultant/trainer/mentor specialising in C#, C++, Java, OO, design, patterns, and process improvement. I am a UK C#, C++, and C standards panel member and a regular contributor to the ACCU Overload journal. My interests include training excellence, design, problem solving, and monty python (required knowledge for all software developers). I'm very very good at sleeping, breathing, and drinking. All of which I practice a lot.

Recently:

I wrote most of a 5 day instructor led training course on C# that now forms part of the official Microsoft curriculum (Introduction to C# Programming, course 2124).

I co-authored (with John Sharp) the Microsoft Press book Visual C#.NET Step by Step which is getting good reviews on Amazon.

I converted the ECMA C# language specification into a hyperlinked HTML presentation using PERL, XML, and XSL (available from my website).

Honey, I Shrunk the Threads: Compile-time checked multithreaded transactions in C++, Andrei Alexandrescu
day: 4, time: 4.00

Would you like to have your compiler tell you "Error: This call will lead to deadlock"? This talk presents new idioms for C++ multithreaded programs, including multithreaded transactions involving multiple objects. You can have your compiler detect not only race conditions, but deadlocks as well.

Track 2

What is the Type of std::toupper()?, Gabriel Dos Reis
day: 1, time: 10:30

The template-argument deduction facility found in standard C++ template machinery heavily depends on the ability ro ascribe types to expressions. That some expressions in C++ don't have (native) types can unncessarily complicate ways in which some natural expressions may be written in a generic framework. This talk explores some points in the space of paths connecting functions and function objects, all in the framework of polymorphism.

Former student of the École Normale Supérieure de Cachan (France), Gabriel Dos Reis got his PhD in Mathematics (Differential Geometry) at Université de Paris VII in 2001. His research interests include applying numerical and computational methods in Geometry and especially in the construction of constant mean curvature surfaces. He began lobbying for C++ in scientific computations since 1996 when he worked on the European scientific project FRISCO (Framework for Numerical and Symbolic Computations) at INRIA Sophia Antipolis (France). He is the author of the numerical components of the new GNU libstdc++-v3. Currently, he is holding a post-doctoral position at the GALAAD project of INRIA Sophia Antipolis. He is also the co-maintainer of libstdc++-v3.

Studying at a Distance, Panels
day: 1, time: 2:00

The objective of this panel is to explore, with the audience, ways in which 'distance learning' can be made to work effectively. The main thrust of this panel will be considering what students can do to improve the value they get from learning this way.

The growth of the Internet, email and other forms of electronic communication has added new dimensions to the traditional mechanisms of distance learning. However the drop out rate continues to raise questions as to how students can get full benefit from such study methods.

Plenty People Programming: C++ Programming in a Group, Nicolai Josuttis
day: 2, time: 10:30

A session, where the audience writes some little programs (using me as the "typing system") for tasks provided by me and where we try out different compilers. Thus, I give a task and wait for input from the audience to put it into the laptop.

And we see, what happens when we run different compilers (such as EDG, G++, Visual C++).

Being a Mentor, Panels
day: 2, time: 2:00

This panel will focus on all aspects of mentoring, both in person and remotely as part of a distance-learning project. The panel will draw on a wealth of personal experience to challenge the audience to consider how they can become mentors, and if they already are, how they can better meet the needs of those they are guiding.

GNIRTS ESAC REWOL - Bringing the UNIX filters to the C++ iostream library., JC van Winkel
day: 2, time: 2:00

The UNIX technique of chaining filters to manipulate streams of data can now also be used in C++ in combination with the iostream library. A framework allows users to build ostream objects without having knowledge of the iostream structure. These ostream objects build upon existing ostream objects, but manipulate the data being output.

Examples of these filtering techniques are: an uppercase/lowercase filter, an ``logging'' filter, that precedes every line with a timestamp, and a ``tee'' filter that allows multiplexing output to several ostreams.

JC van Winkel has a B.S. and an M.S. in computer science (the M.S.from the Vrije Universiteit Amsterdam). He works at AT Computing, a small courseware and consulting firm in Nijmegen, the Netherlands. There he teaches UNIX and UNIX-related subjects, including C++. He has presented C++ tutorials at nine OOPSLA's since 1993.

A common vendor ABI for C++ – GCC's why, what and what not, Nathan Sidwell
day: 2, time: 4:00

The itanium C++ ABI has been adopted by a number of compiler vendors to provide compatibility between itanium compilers. The generic parts of the ABI can be used on other architectures, and GCC took the opportunity to revise its C++ ABI for all targets. The previous ABI had grown by agglomeration as C++ matured. Now there is a complete standard a better ABI can be designed from the ground up. However, common practice can restrict what can be done. The talk will cover both how the ABI came about, some of the design decisions, how it can affect user programs, the promises it makes, and the promises it does not make. We'll also demonstrate concrete use of the ABI to build libraries or tools for cross-modules, cross-compilers, or cross-language systems.

Short bio:Nathan is a consultant at CodeSourcery, and has been involved with G++ for around 4 years. Previously he has worked as a processor architect, before falling into compilers, and then teaching at the University of Bristol. Prior to that he pretended to be a physicist, and knows how to use a soldering iron.

The Roadmap to Generative Programming With C++, Ulrich Eisenecker
day: 3, time: 10:30

Today most applications are developed as individual systems. Very often variants of these systems are required eventually, because of the needs of different customers and diverse contexts of use. Thus, following an approach for engineering software system families from beginning is of significant advantage. Generative Programming is a software engineering paradigm based on modeling software system families such that, given a particular requirements specification, a highly customized and optimized intermediate or end-product can be automatically manufactured on demand from elementary, reusable implementation components by means of configuration knowledge.

Because of its specific implementation of generic polymorphism, C++ templates offer a unique way for static metaprogramming. This allows the implementation of advanced compile-time configuration generators that are at the heart of Generative Programming.

After introducing the fundamentals of Generative Programming, techniques for implementing domain specific languages, configuration generators, and elementary implementation components in C++ will be highlighted.

Finally, I will present an overview of how to add powerful genericity to non-template languages using a frame-processor thus enabling Generative Programming.

Dr. Ulrich W. Eisenecker is a professor for computer science at the University of Applied Sciences Kaiserslautern, Zweibruecken. He spent nearly a decade in industry and closely cooperates with industry today. For some time he was national agent of ACCU in Germany. He edits KOMPONENTEN-Forum, a regular supplement of the journal OBJEKTspektrum, dedicated to component-technology, and co-authored with Krzysztof Czarnecki the book "Generative Programming: Methods, Tools, and Applications" published by Addison-Wesley.

Advanced Template Issues and Solutions (double session), Herb Sutter
day: 3, time: 4:00

This in-depth talk assumes you already know the basics about template programming, including specialization, and have a basic understanding of Koenig lookup. It delves into specific nifty techniques you can use and subtle pitfalls you need to avoid, including answers to the following questions:

First, to get our feet wet, we'll look at a not-well-known corner of Standard C++ templates: What are dependent names, why does the standard mandate two-phase template name lookup in the first place, and how does the two-phase lookup work? Most importantly, how can you make it work for you, and how can you account for the fact that today's compilers do the two-phase lookup variously well?

Second, a look at a cutting-edge issue that has only recently been discovered and debated among the inner circle of world's top C++ library writers, about writing templates safely: Why do you sometimes want and sometimes not want to have Koenig lookup work in your templates, and how can you turn it off when you don't want it? The issues involved can be subtle, they affect your templates today, and they have only this year become better understood.

Herb Sutter (www.gotw.ca) is the author of more than 160 technical articles and of the widely acclaimed books Exceptional C++ and More Exceptional C++ (www.gotw.ca/publications). Herb is convener and secretary, respectively, of the ISO and ANSI C++ standards committees, contributing editor and columnist for C/C++ Users Journal (CUJ), C++ community liaison for Microsoft, and former editor-in-chief of C++ Report.

Generic Build Support – A Pragmatic Approach to the Software Build Process, Randy Marques
day: 4, time: 10:30

Software Engineering was always so simple: A few directories here, a makefile there. Cup of coffee with the guys.

Now suddenly most of my builds fail, builds are not reproducible and a re-build generates a different executable. Something wrong with the coffee?

Or are we with too many people trying to build twenty versions of one million lines of code for three platforms the same way we used to build two versions of 4000 lines of code for one platform? Do we really think that with the experience of building a tool-shed we only need to extrapolate to be able to build a skyscraper?

Do we really need a new dedicated directory- and make-structure for every new project?

Or can we state, in analogy of the construction business, that the build process of a building is always the same irrespectively if you are building a concert-hall, hospital or town hall?

As the title says: a pragmatic approach. No theories. About directory-structures, SCM, makefiles, Compilers and Linkers.

Open to anyone with a basic knowledge of Software Engineering.

Randy Marques is a CASE-Consultant with AtosOrigin – Technical Automation – In-Product Software. Originally from Curaçao, an island in the Caribbean, he now lives in the Netherlands. His main jobs concern the usage of the C-language, Code Quality Assessments, Coding Standards, Code Architecture and Build Management especially in the area of Embedded Software. In general he tries to fix the gap between SPI (CMM) and the work floor (Programmers) He is also a member of the Dutch and International SC22/WG14 Committee and a teacher of the 'Safer C' course developed by Les Hatton.He started with Software Engineering in 1971, he is a real 'old hand' with long-term experience of all levels of Software Engineering. The first computer he worked on was an IBM 1410 with 4K of core memory, at that time one of the biggest in Europe.In his spare time he is a father of 3 kids, a diving instructor and on non-rainy days he likes to take his Kawasaki Vulcan 750 out for a spin.

The Stability of the C++ ABI, Steve Clamage
day: 4, time: 10:30

As C++ evolved over the years, the Application Binary Interface used by a compiler often needed to change in order to support new or changing language features. Programmers expected to recompile all their binaries with every compiler release. An unstable ABI is incompatible with the Solaris philosophy of shared libraries, and is a nightmare for library and middleware vendors. With the advent of the C++ Standard in 1998, there is new hope for a stable C++ ABI on Solaris platforms. This paper addresses the main issues for ABIs using our experiences with Sun C++ on a Solaris Platform.

Steve Clamage has been involved in C++ compilers since 1980. He is responsible for C++ compiler development at Sun Microsystems. He is chair of J16, the ANSI C++ Commitee, and moderates the comp.std.c++ newsgroup.

Fun and Functionality with Functors, Lois Goldthwaite
day: 4, time: 4:00

Functors, or function objects, bring power and flexibility to C++'s Standard Library, but their contribution is too often overlooked or underestimated. This talk explains what function objects are, how to create and use them, and what pitfalls to avoid. In passing, there is also some discussion of ways to achieve similar functionality in other programming languages.

Lois Goldthwaite has been programming professionally since 1983, specialising in C++, C, and Java. As convenor of the BSI C++ panel since 1997 she represents the UK at international standards meetings. She is also convenor of BSI's Posix and C# panels, a member of the C panel, and ACCU's Standards Officer. After a lengthy period of steady employment, she has returned to being a computer mercenary. For recreation she takes out her aggressions on the squash court.

Track 3

Linguistic Variables: Clear Thinking with Fuzzy Logic, Walter Banks
day: 1, time: 10:30

Linguistic variables represent crisp information in a form and precision appropriate for the problem. For example, to answer the question "What is it like outside?" one might observe "It is warm outside." Experience has shown that if it is "warm" and the time is mid-day, a jacket is unnecessary, but if it is warm and early evening, it would be wise to take a jacket along (the day will change from warm to cool). The linguistic variables like "warm", so common in everyday speech, convey information about our environment or an object under observation. We will show how linguistic variables can be defined and used in a variety of common applications, including home environment, product pricing, and process control. The use of linguistic variables in many applications reduces the overall computation complexity of the application. Linguistic variables have been shown to be particularly useful in complex non-linear applications. Linguistic variables are central to fuzzy logic manipulations, but are often ignored in the debates on the merits of fuzzy logic.

Walter Banks is the president of Byte Craft Limited, a company specializing in software development tools for embedded microprocessors. His interests include highly reliable system design, code generation technology, and programming language development and standards. Walter Banks is a member of the Canadian delegation to ISO WG-14, where he co-authored WDTR 18037 (a technical report on C language extensions to support embedded processors). He has co-authored one book, and numerous journal and conference papers.

Coding Standards – Given the ANSI C Standard why do I still need a Coding Standard, Randy Marques
day: 1, time: 2:00

A few years ago, there was an advertisement of BMW on the British television.

The text was something like: "Contains more computing power than it took to take man to the moon". If you are of the opinion that this is good, you should not miss this presentation.

It is about the flaws in the definition of the C-language, about flaws in Compilers, about flaws in management decisions concerning choice of processors and compilers, about flaws in how programmers handle the language and about what you can do to keep the damage to a minimum.

It will not be a theoretical talk but real-life experience.

Open to anyone with a basic knowledge of C-programming.

The Organisation Strikes Back, Alan Griffiths
day: 1, time: 4:00

In "Reworking the Organisation" (presented at the ACCU Conference in 2002) Alan described his first six months at a new employer in the form of a case study.

While there were some early gains there were a number of import questions unanswered: What would happen next? Were the changes dependent on Alan's continuing involvement? How would management intervene? How well would the changes be internalised by the organisation?

In "The Organisation Strikes Back" Alan and Sarah extend the case study to cover a two-year period and (plus some of the history) and, in the process, answer these and other questions.

Alan Griffiths (alan@octopull.demon.co.uk) is a long-standing contributor to the ACCU journals and mailing lists and chair of the ACCU. He is also a member of the BSI C++ Panel (although not a very active one). In working life Alan (alan.griffiths@microlise.com) is "Software Development Champion" at Microlise Limited. He has responsibility for introducing both developers and management to more effective ways of developing computer software. For most of Alan's technical articles and various other goodies visit:

Design and Implementation of the Boost Graph Library, Jeremy Siek
day: 2, time: 10:30

The Boost Graph Library (BGL) offers an unprecedented level of flexibility and power to its users due to its generic programming and generative design. This talk describes the motivation for the core abstractions of the BGL. We will take a close look at a key family of graph algorithms, and how generalized version of these algorithms came to be written as function templates in the BGL. We then will switch focus to the graph data structures provided in the BGL, and look under the hood of the adjacency_list class to see how generative programming techniques were applied to create this Swiss-army knife of a graph class.

Jeremy Siek is a Ph.D. student at Indiana University Bloomington, studying programming languages in the Open Systems Laboratory directed by Dr. Andrew Lumsdaine. Jeremy is a coauthor of several libraries that have become part of Boost, including the Boost Graph Library and Boost Iterator Adaptor Library. Jeremy is a coauthor of the book "The Boost Graph Library: User Guide and Reference Manual". His master's thesis, the Matrix Template Library, applied generic programming to numerical linear algebra. On the industry side of things, Jeremy did a one-year internship under Alexander Stepanov in the SGI C++ compiler and libraries group, and a summer internship with Bjarne Stroustrup at AT&T Research Labs.

Concurrent Programming in Java (double session), Angelika Langer
day: 2, time: 2:00

Support for programming with multiple threads is a core feature of the Java programming language, yet multithreading issues are ignored by many if not most Java developers. Even innocent classes that do not actively start and manage threads should be prepared for use by multiple threads, which raises issues of thread-safety and synchronization.

In the first]art of this talk we give a fairly quick refresher of the basics of thread-safety and concurrency control in general and the Java-specific features in particular. We will also address typical problems such as the nested monitor problem and its potential for deadlocks and we will learn how to apply common techniques such as the conflict set method.

Next we tackle some of the more challenging details of the Java thread API. Specifically, we look into thread termination due to an interrupt request or due to an uncaught exception. We will answer questions such as: How does thread interruption work? Why are the methods suspend() and resume() deprecated? What do we do instead? How do we gracefully terminate a thread in response to a thread interruption? What is the effect of uncaught exceptions on active threads? How can we avoid that an uncaught exception prematurely terminates a thread?

Another less commonly known topic is the Java memory model, which has certain problems with atomicity and sequential consistency of access to volatile variables. The Java language specification gives guarantees for volatile variables, but not all Java implementations actually meet these requirements. As Java developers we would like to know which guarantees we can rely on and which areas we should better avoid.

In the second session we discuss patterns for concurrency control and how they can be implemented in Java. We will look into various incarnations of the adapter pattern and will se which kinds of adapters are relevant in concurrent programming (immutability adapters, synchonization adapters). We will learn how to implement a reader-writer pattern in Java, including the before-after-technique. We will talk about patterns for thread completion (like future, callbacks, group proxies).

The tutorial is of interest to Java programmers, but also to developers who implement concurrent programs in related languages such as C++.

Angelika Langer is a freelance trainer/consultant working and teaching in the area of object-oriented and component-based software development in C++ and Java. She is a recognized speaker at conferences world-wide, co-author of the book "Standard C++ IOStreams and Locales" and author of numerous articles about C++ and Java. A more comprehensive biography can be found at http://www.langer.camelot.de/AboutMe/CV.html.

The Embedded C Extensions to C, Willem Wakker
day: 2, time: 2:00

Embedded C is the name of a language extension to currently being formalized in a technical report by the ISO. Embedded C originates from an industry initiative called DSP-C developed by ACE. It aims to provide C application programmers with access to common performance increasing features of processors used in the domain of DSP and embedded processing.

Embedded C adds fixed-point data types, memory qualifiers and hardware I/O to C. Fixed point data types are frequently used in embedded applications and routinely present in DSP processors. Embedded C adds this primitive type to allow the compiler to make the connection between the two. Similarly, multiple memory banks, typically present in embedded processors to provide the ALU with enough bandwidth, can be addressed using memory qualifiers. Hardware I/O provides a standardized abstraction layer for accessing hardware ports while still allowing for maximal efficiency. This improves portability of, for example, device driver code.

Embedded C is currently being developed by the ISO C working group (WG 14) as a technical report. It is not part of the C language yet. Its aim is to provide common ground for initiatives that currently exist within the industry to give the programmer control over certain hardware features. The report is an important step in streamlining these initiatives and making the development of application support for these features more efficient. The technical report is expected to be ratified in 2003.

The presentation will explain about the meaning and use of the new language extensions, including examples of their use and resulting DSP processor code. Additionally, the rationale behind the design will be discussed, including alternative proposals that did not make it, and the relation between Embedded C and C++.

The presentation will discuss the meaning and use of the language extensions and provide examples of each. Additionally, the motivation behind Embedded C and the rationale of certain design decisions will be explained as well as the status of the technical report. The relation with C++ is also addressed.

Willem Wakker has been active in the world of formal standards and programming language standardisation since 1988. He also has extensive industrial experience in this area having joined ACE in 1977 and led their compiler development team until 1995 when he was appointed Director of ACE's Consulting operation. Willem is currently in WG14 and is the project editor of the Embedded C Technical Report. It should be noted that ACE defined and introduced the predecessor of Embedded C, namely DSP-C.

The Timing and Cost of Choices, Hubert Matthews
day: 2, time: 4:00

As programmers and designers, we are contantly make choices. We think very hard about these choices but how often do we think about the choice process itself? What are the consequences of choosing now versus delaying, and when do we validate those choices? This talk compares the effects of early versus late binding for code, process and requirements and shows that there are patterns that relate: product choices and process choices, binding times and the timespan of choices, the effects of being able to revisit those choices later, and the choice of emphasis on error prevention, removal or tolerance.

Hubert is a freelance software consultant and trainer based in Oxford. His main areas of expertise are architecture and design and he has been an architect for a number of clients, both small and large. He has also given training courses across the globe in a range of subjects including C++, Java, UML, EJB, OOA/D, patterns and components. Other areas of consultancy include technical strategy, architectural audits, and process issues.

Following a degree and doctorate in engineering at Oxford (LMH) and a student apprenticeship with GEC in Rugby, Hubert worked at CERN in Geneva before returning to Britain to indulge his other passion: singing. Since leaving the Royal Northern College of Music in Manchester, Hubert has continued to juggle his two careers as a professional concert and opera singer and as a freelance software consultant (including the obligatory dot-com nightmares and other abortive get-rich-quick schemes).

In his abundant spare time, Hubert coaches rowing, dances salsa and drives too fast.

Design Patterns in C++ and C# for the Common Language Runtime, Brandon Bray
day: 3, time: 10:30

Writing code for the common language runtime necessitates a certain number of considerations. Among them are efficiency in the face of a managed execution environment, verifiability, and library portability between languages. This talk will address some of the frequent design patterns used by programmers familiar with C++ and C#.

Pattern Writing: Live and Direct, Frank Buschmann
day: 4, time: 2:00

Pattern Writing: Live and Direct

The overwhelming majority of developers familiar with patterns would consider themselves consumers rather than producers of patterns literature. How would you get started writing a pattern? And how would you continue? One way to understand something is to take it apart; another is to make one. This tutorial aims to do both of these for patterns.

It starts with an anatomical dissection of the common pattern, and places patterns in the context of a pattern language. The two presenters then pair-write patterns in an actual pattern language, switching between writing, making meta-level comments about the issues encountered, and taking suggestions and questions from the attendees.

Frank Buschmann is senior principal engineer at Siemens Corporate Technology in Munich, Germany. His interests include Object Technology, Frameworks and Patterns. Frank has been involved in many software development projects. He is leading Siemens' pattern research activities. Frank is co-author of "Pattern-Oriented Software Architecture – A System of Patterns" and "Pattern-Oriented Software Architecture – Patterns for Concurrent and Networked Objects"

Track 4

Extreme Hour (XH): (workshop) - Jutta Eckstein and Nico Josuttis, Jutta Ecstein
day: 3, time: 10:30

Experiencing extreme programming (XP). The XH is "the smallest project in the world", which lasts an extended hour, where people experience most of the aspects of XP, but without actual programming.

Jutta Eckstein is an independent consultant and trainer from Munich, Germany. Her know-how in agile processes is based on over ten years experience in developing object-oriented applications. She is an experienced XP coach and trainer. She worked with teams of different sizes mainly in the finance industry to help them using agile processes successfully. Besides engineering software she has been designing and teaching OT courses in industry. Having completed a course of teacher training and led many 'train the trainer' programs in industry, she focuses also on techniques which help teach OT and is a main lead in the pedagogical patterns project. She has presented work in her main areas at ACCU (UK), OOPSLA (USA), OT (UK), XP (Italy) and XP and Agile Universe (USA). She is a member of the AgileAlliance (http://www.aanpo.org) and a supporter of the Manifesto of Agile SoftwareDevelopment (http://www.agilealliance.org).

What can MISRA-C (2nd Edition) do for us?, Chris Hills
day: 3, time: 2:00

MISRA-C a small coding guide for the UK Automotive industry has escaped not only from the automotive industry but is now used worldwide. It has sold at a rate of over 1000 a year in the last 4 years!

This paper will look at why MISRA-C was needed, what it gives us and what is new for the second edition. The new version of MISRA-C will be published in March 2003 but as I am on the working group I can give an insight into the changes and the process used to produce what will be the new version.

Embedded Engineer. My background covers most types of non-PC based programming from Unix systems to smartcards and various forms of electronics. Currently the BSI convenor for IST/5/-/14 (the BSI's C Standard's Panel to you and me!) Also a member of the MISRA-C working group. Past papers presented to the ACCU are at http:// QuEST.phaedsys.org

Using Aspect Oriented Programming for Enterprise Application Integration, Arno Schmidmeier
day: 3, time: 4:00

Aspect oriented programming is a new but urgently needed software development paradigm, to modularize tangling code at all levels.

Aspect oriented programming is widely used for low level elements like synchronizing, tracing, logging, exception handling, etc.

But it also unleashes its power in EAI-projects, as it has proofed in several very successful projects.

Arno Schmidmeier is an independent consultant focusing on aspect oriented software development and enterprise application integration. During his very successful time as Chief Scientist at Sirius Software GmbH, he was responsible for the commercial adoption of new technologies like AOP. He deployed several large-scale projects based on AspectJ. He offers consulting services for use and introduction of AspectJ in commercial projects.

He was an independent expert on JSR 90.

How to Handle Project Managers: a survival guide, Barb Byro
day: 4, time: 10:30

A practical guide to project managers. Having a good working relationship with your project manager can make or break a project. We'll discuss strategies to evaluate and work with project managers. Not all project managers appreciate programmers, but they can be trained. We'll also touch on how this can make ISO compliance less onerous.

Barb Byro is a project manager who has spent the last 3 years managing Telecom network and collocation projects for customers of Metromedia Fiber Networks in both New York and London. She's also lead ISO compliance teams, devising practical strategies to make ISO into a useful tool rather than a time wasting, irrelevant exercise. Barb's also managed online computer game software development projects for over 10 years. In that capacity she's worked with paid and volunteer staff, both in-house and external.

The Future of Programming Languages, Goldfish Bowl
day: 4, time: 2:00

Changes in hardware have challenged language designers to consider adding new features to languages. On the other hand, issues of backward compatibility make changes to languages unpopular.

In this interactive session the audience under the guidance of a 'management team' will discuss ways of resolving the tension between the needs of the future and those of the past.

Guidance will be given at the start of the session to those who have not previously experienced this type conference item. You will be able to remain in a purely observational role if you wish but opportunities for active participation are an essential ingredient of a 'goldfish bowl.'

Agile Enough?, Alan Griffiths
day: 4, time: 4:00

The group of development methodologies recognised under the banner "Agile" have gained a lot of attention recently. They are clearly successful for a range of problems, but is this range broad enough?

My experience is that there are many projects for which "key" process elements cannot be followed, for technical, business or organisational reasons. Indeed, most of the projects I've worked on fall into this category.

How important are these types of projects? Can they really be forced into an "Agile" straitjacket? Can they be delivered effectively?

Is "Agile" sufficiently liberal in scope?

Track 5

The Lambda Library : Unnamed Functions for C++, Jaako Jarvi
day: 3, time: 10:30

More often than not, small and simple function objects are needed solely to be passed to an STL algorithm, having no further use in the program. Defining new functions or classes for just this purpose is verbose and adds unnecessary names to the program. The STL programming style would greatly benefit from lambda abstractions: unnamed functions that can be defined where they are used, at the call sites of STL algorithms. As unnamed functions are not part of core C++, the standard library provides

binders, adaptors and the set of elementary function object templates (less, negate, ...) as an alternative. However, these tools have many restrictions and leave much room for improvements.

The Lambda Library (LL) offers these improvements, providing a rich set of tools for defining unnamed functions, with an intuitive syntax.

This talk explains why LL makes using STL easier, simpler and more fun.

The talk describes the main features of the LL, and outlines its design and implementation.

Jaakko Järvi has a Ph.D. in computer science from the University of Turku, Finland. He is currently a post doctoral researcher in the Open Systems Laboratory, one of the Pervasive Technology Laboratories at Indiana University. His research interests include programming languages and generic programming. Besides academia, Jaakko has worked as a software engineer at ABB Corporate Research, and as the CTO of Atuline Ltd., a university spin-off company on its way to maturity. Jaakko is the principal author of the Boost Tuple and Lambda Libraries and participates actively to the C++ standardization work.

Guido van Rossum Interview

Guido van Rossum is well-known as the creator of the Python programming language and has steered its development for over 10 years

The last time I hand-wrote a real letter was ...

too long ago to remember. Honest.

Its still a shock when ...

journalists can't spell "it's".

(Whoops ed.)

I can still remember ...

when computers used punched cards. My biggest creation at the time filled up an entire box (2000 cards).

I first realised I was in the right career when ...

I got a part-time job as a system programmer while still in college, the first one I ever applied for.

There's terrific snobbery ...

amongst many Linux users, who scoff at all things Windows. Not that I'm a Windows fan (far from it), but some Windows things are well done, and it's important to learn from those (as well as from Windows' mistakes).

People can waste their lives ...

tweaking HTML.

My worst sporting moment at school was ...

every sports event I ever participated in.

The most beautiful thing I saw today was ...

my son Orlijn, sleeping.

Plants in my house survive for ....

about 15 minutes.

I couldn't live without ...

my wife, Kim.

I got into trouble at school ...

for not paying attention when I was too far ahead of the rest of the class in math and physics. :-)

I believe your son, Orlijn, recently turned one year old. (Congratulations)

Apart from sleep deprivation, how has he changed your life?

Thinking about Orlijn's future made me view my own future in a different way. The responsibility for such a beautiful person made me rethink my priorities.

Python has grown enormously since 1991, and it seems to benefit from work from a wide variety of people. What was it like realising that people, even on the other side of the world, were interested in your work, - that it could grow into more than "your" project.

It sunk in relatively slowly, but in 1994, it really dawned on me when there was a long thread in comp.lang.python about what would happen "if Guido was hit by a bus." (You can search Google to read the thread. :-) I began to realize that Python was more important than my "day job". When, as a result of that discussion, the first Python workshop ever was organized, I was very excited to meet some of the faces and voices belonging to the people who weren't much more than a name and an email address to me.

FORTRAN is often hailed as the first high level computer language, and is 45 years old. Would you still want to be involved in computer languages when Python reaches that watermark? And, given how far we have come since 1957, do you have any idea what languages will look like in 2036?

I would love to be around then, and I hope that Python will have made a mark on the design of future languages. I doubt that I'll be designing the "next" big language any time soon, even if I tried: creating a hit like Python was pretty much a lucky shot. I don't mean to say that I don't deserve the success, but I believe that many other factors besides the technical quality of a design contribute to its ultimate success. Python is not particularly innovative, but its design is "well rounded" in a number of ways. I like to call it the "Goldilock" language: not too hot, not too cold, but juuuuuuust right. But that's only in hindsight: Python landed in quite a different niche than what I had in mind when I first designed it. For example, I wasn't thinking of embedding Python in other applications, I didn't aim it to be an educational language (even though it was based on an educational language, ABC), and I didn't plan for it to be the main development language for large systems like Zope.

You have acted as a guiding hand for Python for ten years, but some people disagree with the directions choosen, and are occasionally less than polite in pointing this out. How does this affect you?

I tend to get less than polite in my responses. That doesn't always work, though, and I've had to accept the fact that in a large enough community, nothing remains uncontroversial.

Given the choice between life with Python and your family, and life as a Pina Colada tester on a beach in the Carribean with your family, which would it be?

Python, definitely. I'm not a beach person. My wife is, though, so we spend some of our vacation doing "beachy" things. Our first "geek cruise" was a big success for all three of us, and I've already signed us up for another one next year.

The PC, whilst seeming ubiquitous, is still unused by 5.6 bn of the Earth's population. How do you see the other 5.6bn progressing - will we all have a Dell or is there another way?

I think it's inevitable that the others will want PCs too, and since we tend to give ours away when they're only 2-3 years old, they won't be behind the curve too far. They'll probably run open source software, because they can't afford to pay Microsoft. So there's a social side to the open source movement (besides fighting monopolies).

Eventually I expect we'll settle on a different form factor, or form factors, depending on use: handheld organizers will become ever more powerful and popular, and where larger input or output areas are needed, maybe tablets (which won't look much like the tablet prototypes that were launched recently). For household use, the tv-set form factor will probably survive, for playing games, watching movies, and as a base station for the portable devices to back up their data. Eventually computers will have many different form factors, determined by considerations of ergonomics rather than ease of manufacturing. I don't believe in the widespread use of "combo" computers: cell phones that are also organizers, MP3 players, cameras, games, wireless email access, and so on. While for some people it makes sense to carry only one brick with all that functionality, usually the combos don't perform any function as well as more focused devices, and it's usually difficult to use two different functions at once. I'll stop now, since I read most of what I said here in the Sunday Washingington Post. Yes, I use a printed newspaper as my main information source, even about technical stuff that's in the periphery. I don't watch tv, nor do I read slashdot; I do listen to public radio. And that's it for rambling. :-)

Guido van Rossum was kind enough to allow himself to be interviewed by Paul Brian. Thanks.

Martin von Loewis Interview

Martin von Loewis is well-known as a Python contributor and c.l.p. regular.

The last time I hand-wrote a real letter was ...

... for Christmas. I don't see hand-writing personal letters as antiquated, but I would hope that Word-written-then-printed letters go away one day, and get replaced with email. I find that I write most of these to authorities which haven't heard about eGovernment.

Its still a shock when ...

... I see people answering a phone call on the street.

I can still remember ...

... never hoping to travel to the Western world one day. I enjoy the grace of "late birth", where the iron curtain fell shortly after I finished school.

I first realised I was in the right career when ...

... I managed to impress my physics teacher with an assembler program I entered into a Z80 machine after I assembled it to machine code on paper.

There's terrific snobbery ...

... having your employer pay for non-business things you do on a business trip.

I realised the meaning of Schadenfreude when ...

... I was a kid. This being a German word, I can't think back that far.

Nice answer.:-)

People can waste their lives ...

... thinking about retirement.

Artists should always ...

... consider themselves as servants of their audience.

If I could change places with anyone it would be ...

... Ruud Lubbers (the current UNHCR).

My worst sporting moment at school was ...

The entire subject of physical education was terrible.

Plants in my house survive for ....

... two years. It actually got better recently as I got more plants that can stand being forgotten for a couple of weeks...

I couldn't live without ...

... television.

Aha - an honest answer.

I got into trouble at school ...

... because I said I did not want to join the army. They were effective in convincing me to change my mind.

You serve on the board of the PSF. Do you ever feel that the politics gets in the way of the language (not necessarily between board members, but the issues normal users have).

Licensing is in the way often in surprising ways. Some contributions cannot be accepted because of licensing, often to the surprise of the contributor. This happens rarely, though. An ongoing issue is that there are so many prior copyright holders of Python (none of them being Guido van Rossum); I do hope the PSF can eventually obtain the intellectual property from those organizations, so that they stop having a legal standing in the future of Python.

US export restrictions used to be politics that get in the way of free software in general (Python itself isn't that much affected); fortunately, those restrictions have be lifted to codify the de facto practice.

People sometimes term issues of personal dislikes as "politics", e.g. sympathies and antipathies between contributors, or of a contributor towards certain technology. This is not the business of the PSF: all technical aspects of the Python development are dealt with on python-dev.

FORTRAN is often hailed as the first high level computer language, and is 45 years old. Would you still want to be involved in computer languages when Python reaches that watermark? And, given how far we have come since 1957, do you have any idea what languages will look like in 2036?

It is difficult to make predictions, especially about the future.

I notice that some old-timers of computer science still provide valuable contributions to language development, so I hope I can contribute as long as my contributions are considered as valuable. However, I don't have any idea of how computers will work in 2036. I do think the year 2038 problem will be solved much before that.

The PC, whilst seeming ubiquitous, is still unused by 5.6 bn of the Earth's population. How do you see the other 5.6bn progressing - will we all have a Dell or is there another way?

The majority of these people are just too poor to afford a computer, and, if asked, I doubt many of them will bring up computers as the top one luxury that they would like to have.

I expect that computing devices will further specialize, so that the tasks people solve with the computer most of the time will be solved with special devices in the future: games, interactive communication, messaging, online shopping, etc. I don't think that the current offerings of specialized devices are suitable (except for the game consoles), so the PC will be with us for quite some years - but humanity may grow faster than the PC market.

Martin von Loewis was kind enough to allow himself to be interviewed by Paul Brian. Thanks.

Marc-Andre Lemburg Interview

Marc-Andre Lemburg is well-known as a Python contributor and the author of the widely used mx package.

The last time I hand-wrote a real letter was ...

That must have been sometime in 1993 while I was studying in France. Not everybody had a email back then and some friends didn't like electronic mail. That has changed since then. I am more into writing email, IRC and related messenger technology these days. Unfortunately, that still doesn't keep me from doing tons of business related paperwork.

Its still a shock when ...

... I realize that I've found a bug in a software release I made minutes ago.

I can still remember ...

... my first computer program: a short BASIC program implementing a simple clock back in the summer of 1980.

I first realised I was in the right career when ...

Now that's a good question. When I decided to go to university, I had the choice of becoming an artist, study computer science, physics or math. I eventually ended up starting out with physics and after feeling uncomfortable with their usage of approximations in theoretical mechanics switching over to math. Since then I have been programming a lot and loved it so much that I started a company. Still, I sometimes feel that I would have probably been better off working as an artist rather than trying to run a business.

Who knows, perhaps in a few decades ahead I'll switch over (once again).

People can waste their lives ...

... by not stopping every now and then to think about how great life really is.

My worst sporting moment at school was ...

[Not sure what to answer here: I'm not very much into sports, except Modern Dance, so there probably was none.]

The most beautiful thing I saw today was ...

... the crystal clear blue morning sky.

Plants in my house survive for ....

Just moved into a new apartment: lucky for the plants -- we haven't bought any yet.

I couldn't live without ...

... my partner. She has given me more enjoyable moments in life than anyone else in this world.

I got into trouble at school ...

I didn't get into any real trouble, but I often lost interest in the boring things the teachers were trying to teach us. Whether that's to blame on the teachers or on the German school system is a different topic.

You are well known for the mx packages (including mxODBC). How did you make the (brave) decision to make a living from your own software?

eGenix is not making a living from mxODBC and that was never intended. We are mainly focused on consulting and doing customer projects which involve the complete set of mx tools, a few of which are not available to the general public.

Selling mxODBC to commercial Python users was more an experiment to see how well it got accepted. At the time, mxODBC was the only Python extension which you had to buy (at least as far as I remember). It turns out that there is indeed a small but growing market for selling Python software components.

FORTRAN is often hailed as the first high level computer language, and is 45 years old. Would you still want to be involved in computer languages when Python reaches that watermark? And, given how far we have come since 1957, do you have any idea what languages will look like in 2036?

I probably won't be involved in Python language development anymore. I am more interested in designing complex software systems than actually writing the code for it -- even though writing Python is much more fun than coding C, Pascal, Assembler or BASIC as I did in the past.

That said, I am convinced that Python will have a long stay in the computer business and, just as Perl, will reserve itself a place in history.

What I'm still a little worried about is that the lack of business interest in the language will cause Python to remain outside the scope of project managers. I wish that a company of an IBM or Sun scale would invest into the language and push its marketing.

The computer industry could save a lot of dollars by turning away from system languages for project based development and moving towards rapid application design paradigms involving modern very high level languages such as Python and Extreme Programming techniques. Of course, consulting companies know this. The problem is that their customers usually don't. As a result the consulting firms spend more time by deploying on system languages and thereby making more profit.

I believe that this is mostly due to the different structure of mind set within the management level of large companies. The saying "nobody ever got fired for buying IBM" still holds true very much today and you can port that attitude to the software development process as well. Managers know that they are on the safe side by deploying on Java and C++. If something goes wrong it's easy to find consultants who can fix the problems. This doesn't (currently) hold for languages like Python.

Without support backup from large companies, I think the only way to get Python into the minds of managers is by building on success. The Siena Architecture was such a success story for us and I'm sure others in the Python field have had similar experiences. However, it takes a lot of convincing and open minded managers to get a process like this going.

The potential I see for Python is in very diverse companies. Mid-sized companies tend to go by one strategy. Large companies often display a more diverse IT infra-structure. Small companies care for the money they have to spend. My feeling is that the latter two are most likely to become potential Python users.

Given the choice between life with Python and your family, and life as a Pina Colada tester on a beach in the Carribean with your family, which would it be?

If someone would give me enough money, I wouldn't mind moving to a nice warm island in the Pacific, equipped with a satellite uplink to keep contact. In reality, however, I'd probably miss the dark small cinemas, art exhibitions, bars and clubs that metropolitan cities have to offer. It's still a nice thought, though :-)

The PC, whilst seeming ubiquitous, is still unused by 5.6 bn of the Earth's population. How do you see the other 5.6bn progressing - will we all have a Dell or is there another way?

I don't think that the current "PC" kind of computer is going to last. The future will give us smaller, less noisy machines with smarter input and output facilities. Those machines will eventually replace Hifi and TV, and could indeed spread to a much larger share of the world population than the current "personal heating systems"; mobile phones and PDAs will lead the way.

Apart from that, I believe that the world has more urgent problems to solve. What we are currently seeing in form of globalized terrorism is different from the uprise of the "third world" people have seen coming for years, but the originating cause is nethertheless the same. The "first world" is currently busy industrializing knowledge. Luckily, the Internet has given the other two worlds a chance to catch up and I see a perspective here which hopefully makes a difference.

Marc-Andre Lemburg was kind enough to allow himself to be interviewed by Paul Brian. Thanks.

Eric S. Raymond Interview

Eric S. Raymond is the author of "The Cathedral and the Bazaar" and a well-known writer and speaker on open source issues.

You are often seen as a spokesperson for open source software, and your support for Python has brought new "converts". Do you ever feel like public property ?

No. It can get pretty damn stressful nevertheless.

To help on the stress front, if you could ban anyone from asking you one particular question about open source ever again, what would it be?

"What's the future of open source?" Cripes. If I knew how to do prophecy I'd found a religion or something. Er, no, wait, I've already done that. Twice.

(that was the next question - ed.). OK, Given the choice between life with Python (and your family), and life as a Pina Colada tester on a beach in the Carribean (with your family), which would it be?

Python. Beaches are cool but I don't drink.

OK - some rapid fire questions if you don't mind.

"The last time I hand-wrote a real letter was ..."

Oh, around...1975, I think.

"Its still a shock when ..."

Hardware just keeps getting relentlessly cheaper and more capable.

"I can still remember ..."

The musty yellow paper rolls on ASR-33 teletypes -- my first interactive computing, back around 1972.

"My partner and I dislike ..."

Stupidity. Television. Easy-listening music. Bland food. Victimology.

Sorry - what's victimology?

The manufacturing of grievances, especially through politics, and *especially* through identity politics.

Thank you.

OK - "I first realised I was in the right career when ..."

I found myself marvelling that I was getting paid for what I was doing.

"Artists should always ..."

remember that if they're not reaching an audience, they're just masturbating.

"The most beautiful thing I saw today was ..."

My wife Catherine.

Bonus points there I think. :-)

"Plants in my house survive for ...."

Pretty much forever. I have a green thumb. Yes, I know that's odd in a hacker.

"I couldn't live without ..."

Um. Food? Water? Oxygen?

Silly question. "My worst sporting moment at school was ..."

Every single one of them.

"My first home computer was ..."

An Osborne-1

Have a look here (http://www.imarshall.karoo.net/osbourne_1.htm)

miss that computer because ..."

Pull the other one, I don't miss it for a nanosecond.

The PC, unlike the Osbourne-1, might seem ubiquitous, but is still unused by 5.6 bn of the Earth's population. How do you see the other 5.6bn progressing - will we all have a Dell or is there another way?

There's another way, yes, but it's not here yet. Handhelds with a 24/7 wireless internet link. Do most people buy computers to run Mathmatica? Nope. They use them for email, for the Web, for word processing. Most people have more need for communication than computation.

Very quotable:-)

So, what is most important to you outside the world of software ?

Freedom. That I have it, and that others have it too.

And with that thought we shall leave it. Thank you Mr Raymond.

Thank you.

Eric S. Raymond was kind enough to allow himself to be interviewed by Paul Brian. Cheers.

Alex Martelli Interview

Alex Martelli is a long time python enthusiast and contributor. He recently edited the Python Cookbook from O'Reilly, has another book coming out soon and shall be speaking at the April conference.

The last time I hand-wrote a real letter was ...

in the summer of 1976. I remember the occasion well -- I was on vacation, without access to my beloved electric typewriter (which my dad had given me years before, as a gift for by 7th birthday), yet I did have to write at once, couldn't wait to get back home (matters of the heart...).

Its still a shock when ...

I stop to consider how much RAM is in the PDA in my pocket -- more than in the whole mainframe I learned to program on.

I can still remember ...

the thrill of running my very first program -- a pack of punched cards in Fortran -- the card reader slurping it in, the line printer emitting the listing and results, all without a hitch. Three of us students had been reviewing that code for days... Then, of course, after that one first program, hubris set in, and rarely has a substantial program of mine compiled and run without a hitch first go -- which is why I still remember that one magic first time, of course!

I first realised I was in the right career when ...

...the floppy drive I was trying to assemble for my "jupiter Ace" (a forth-based British PC similar to Sinclair's ZX basic-centered ones) finally crumbled in my hands, obviously irretrievable. That's when I _knew_ I had made a lucky choice in shifting careers to make a living doing software rather than hardware...!

There's terrific snobbery ...

...in boasting about having HAD to bootstrap a mini via console switches, cut and splice paper tape back together to "edit" a program, solder one's own chips to the motherboard -- about having lived through the times when we did have to do that. In fact it all mostly depends on having about the right age, i.e., just on getting old!

Artists should always ...

...remember they're just the same breed as technicians. Ars and Techne are one another's *translations* in Latin and Greek respectively -- there are different arts (also known as technologies), but only one Ars, Art, Techne.

People can waste their lives ...

...in many different ways, most of them deadly boring and unplesant. My tip: if you choose to waste your life, pick an interesting and pleasant way to do so.

My worst sporting moment at school was ...

...just about every one of them. I loved study, couldn't stand sports.

Plants in my house survive for ...

...as long as I don't try to help out with them -- i like plants, but I think it's not mutual -- I have to admire them from a distance, or else...

I couldn't live without ...

...music -- Bach, Offenbach, Mozart, Nymann, Monteverdi, Sullivan...

I got into trouble at school ...

...for helping friends out too much -- teachers didn't take well to my whispering suggestions during interrogations or passing sheets of paper with the answers on written quizzes. I realize NOW I wasn't actually *helping* them, but it sure felt that way at the time!

My first home computer was ...

...a hand-assembled ZX80.

(Wha hey ! - ed.) - ...I miss that computer because ...

...well, on second thoughts -- I *don't* miss it! the mainframe and minis we had at the university, now THOSE were something. But I didn't own a home computer worth missing until I got a (pre-assembled) Acorn "Atom"... but that wasn't the first!-)

You are going to be speaking at the Python UK Conference in April (shameless plug to be inserted here). Can you please tell us why you choose to speak there?

The fact that it's held together with ACCU's conference was a big attraction to me -- maybe we can lure some great C/C++ programmers to give Python a chance...!

Given the choice between life with Python and your family, and life as a Pina Colada tester on a beach in the Carribean with your family, which would it be?

Python! Too easy -- I don't LIKE Pina Colada.

Here's hoping. If you ever lost your (infectious) enthusiasm for Python, what would be the last aspect you would lose?

The deep respect for Python's design -- perfect balance of simplicity and power, of purity and pragmatism, of so many things.

Python is sometimes described as a near perfect balance between Object orientated, procedural and functional programming. Which of these three is most underused and does that matter?

Functional programming is vastly underused, both in Python and out of it. In Python, because the balance isn't quite perfect -- procedural and object capabilities are superb, functional is somewhat less so. In general, because it's hard to learn to *think* functionally (well, for all but math majors, perhaps) AND it's harder to get good performance with programs that don't modify data (i.e., functional programs). It matters, a bit, because the potential of functional programming is quite substantial for tasks that are amenable to it.

And finally, what is the most important lesson people can learn from/about Python?

Great design is not about making no compromises, it's about making just the RIGHT compromises. That's why it's never easy.

Alex, Thank you.

Thank you.

Alex Martelli was kind enough to allow himself to be interviewed by Paul Brian. Thanks.

Sponsors:

Microsoft Visual C++ .net Microsoft is proud to participate as a sponsor for the Association of C and C++ Users Spring Conference 2003! This year marks the 10th anniversary of the Visual C++ developer-tool product, and it is fitting that an all new release is currently pending. Visual C++ .NET 2003 will have hundreds of updates including many that raise the ISO C++ conformance rating of the compiler to over 98%. We challenge every ACCU attendee to try the C++ techniques they learn at the conference with Visual C++ .NET 2003!

PERFORCE SOFTWARE Perforce is the Fast SCM system that runs on over 50 platforms. Its focus on performance, reliability and usability particularly appeals to developers that find excessive process an obstacle to productivity. Perforce is used by organisations with dispersed development teams working to get quality new products to market on schedule.

BLACKWELL'S live life buy the book Blackwell's first bookshop opened its doors at 50 Broad Street, Oxford in 1879. We now have over 60 bookshops in 25 university cities and towns throughout the UK, supporting the information needs of academics, students, professionals, libraries and businesses. We aim to hold the most comprehensive range of books in your speciality with a choice of ways to access our service: a combination of retail, mail order and online bookselling. Full details of Blackwell's shops, services and details of more than 1.5 million books, both in and out of print, can be found at **www.blackwell.co.uk**

Exhibitors:

ROGUE WAVE SOFTWARE Field-proven C++, XML and GUI components from Rogue Wave® give development teams a head start on building applications that solve business problems, increasing productivity and the ability to deliver products on time. Rogue Wave help extend existing developments in C++ with their new web integration products and boost performance of applications with ATS, their drop-in memory allocator.

QBS SOFTWARE QBS Software has been supporting and selling software for over 14 years and is a leading reseller of programming tools in Europe. We work with over 250 publishers and sell nearly 800 software tools to the developer, the corporate customer and to over 200 resellers throughout the world. Tools sold include those that support .NET, C++, Java, VB and Delphi and range from code, comms, interface, system, reporting and data controls, to installation, help creation, web development and desktop publishing software. QBS has always maintained its independence and is well-respected in the industry for product knowledge, fast and friendly service and fair pricing.

PCG PROFESSIONAL CONTRACTORS GROUP The Professional Contractors Group is the trade association that represents the interests of freelancers and is a not-for-profit organisation run by freelancers for freelancers.
The PCG has evolved to become the freelancers champion, campaigning on issues that matter to the freelance community, irrespective of industry focus. It is committed to promoting members commercially and supporting their development.
Please see the link to our homepage:**www.pcg.org.uk/**