Institution: University of Kent

Unit of Assessment: 11

Title of case study: Communicating Process Architectures: the Future for Systems

1. Summary of the impact

Modern processor architectures (networked multi/many-core nodes), together with society's expectation of evermore-complex applications, require fluent mastery of concurrency. To enable this mastery, in the last two decades our group has taught, researched and developed fundamental notions of *concurrency*, new *programming languages* (occam-pi, and the KRoC toolset), *libraries* (JCSP, CCSP, C++CSP, CHP), *runtime systems* (the KRoC/CCSP multicore scheduler) and *tools* based on formal process algebra (Hoare's CSP, and Milner's pi-calculus).

Our work has had impact in providing new mechanisms for software development in a number of sectors such as chip design, large-scale real-time systems, formal interfaces and testing and the space industry. Testimonials supporting this are available from a variety of industrial and commercial sources (NXP Semiconductors, Big Bee Consultants, Philips Healthcare, 4Links Ltd. and Microsoft Research Cambridge). The breadth of impact of the work is evidenced by download statistics, as well as by third-party contributions to libraries and documentation.

2. Underpinning research

Since 1993, our research has focussed on transforming the ways computer systems are designed, verified, implemented and maintained. Our key insight, which went against received wisdom when we started this work, is that concurrency is a necessary element for all those stages. Our research has focussed on an approach based on rich mathematical theories of communicating processes (CSP and pi-calculus), combined with *well-engineered* and *highly efficient* languages, libraries, run-time systems and tools. Our work has contributed significantly to the latter, enabling concurrency to take its rightful place as a fundamental, simple and powerful tool for the production of systems, with performance and responsiveness achieved as a natural consequence, rather than as a primary aim. We call this approach *Communicating Process Architectures (CPA)*.

Amongst other things, these led (over the next 17 years) to our development of the occam-pi [1] programming language: a careful blending of the dynamics of Milner's pi-calculus (mobile channels and processes) with Hoare's CSP underpinning of classical occam, together with significant extension of CSP capabilities (e.g. *barriers*, *shared channel-ends*). Supporting this, we have researched and developed the lightest (by some margin) multicore scheduler available today, on which applications automatically and effectively scale with the number of cores supplied [2] and a fast algorithm for the resolution of general CSP choice (between barriers, input/output guards and timeouts) [3]. In parallel with this, we have researched, designed, implemented, documented, open-source published, used and demonstrated a set of libraries making available this extended occam/CSP/pi-calculus model for 'mainstream' languages: Java (JCSP), C (CCSP), C++ (C++CSP) and Haskell (CHP).

Our research has been supported through a series of EPSRC funded projects, most recently: TUNA (EP/C516966/1; £60,867; 2005 –2007; with Universities of York and Surrey); RMoX (EP/D061822/1; £210,405; 2006 –2010); CoSMoS (EP/E049419/1; £398,058 to Kent; 2007–2012).

These projects focussed on applying the ideas of CPA and massively parallel process-orientation to, respectively, the engineering of nano-scale machines, operating systems and the modelling and control of complex systems with emergent behaviours. Representative papers are [4, 5, 6]. Most recently, we have proposed extensions to occam-pi to set up and model-check verification assertions about program behaviour *within the program*: the idea being that, in the near future, verification will be so crucial that programmers will need to do this as a matter of course (and without needing to become experts in the underlying formal process algebra). Numerous people at Kent have been and are engaged on this work: Peter Welch (Professor, since 1993), Fred Barnes

(PhD/Lecturer, since 1999), Adam Sampson (PhD/RA, 2007-11), Carl Ritson (PhD/RA, since 2006), Neil Brown (PhD/RA, since 2005), Christian Jacobsen (PhD, 2005-09), Matt Jadud (PhD, 2005-09), Damian Dimmich (PhD, 2006-10), Jon Simpson (PhD, 2007-13), Jim Moores (PhD, 1995-99), Kevin Vella (PhD, 1997-2002) and Mario Schweigler (2001-06).

Professor Welch was elected Member of IFIP Working Group 2.4 (Systems Implementation Technology) in 2002, where his presentations [section 5: U1] have impact on key researchers in major international companies (including IBM Research, Microsoft, Adobe, Hewlett-Packard, Azul Systems) and universities (such as Purdue, Irvine, Colorado, CMU, Waterloo, Linkøping, Karlsruhe, Pretoria, Cape Town, Stuttgart, Paderborn, Queensland, Pretoria). As a result of this, he was invited by the Software Engineering Institute (SEI) at Carnegie-Mellon University for a sabbatical term in the fall of 2007, where he presented a course on process oriented design and occam-pi [U2] and collaborated with senior researchers on emergent behaviour in very large scale parallel systems. This continues and two of the researchers (Wallnau, Klein) are co-authors of [6].

3. References to the research [** - numbers 1,2,5 are the most significant research contributions]

1. ** **Communicating Mobile Processes: Introducing occam-pi**. Peter H. Welch and Frederick R.M. Barnes. In A.E. Abdallah, C.B. Jones, and J.W. Sanders, editors, *25 Years of CSP*, volume 3525 of *LNCS*, pages 175-210. Springer Verlag, 2005. *[Cited 66 times in Scopus.]*
2. **** **Multicore Scheduling for Lightweight Communicating Processes.** Carl G. Ritson, Adam T. Sampson, and Frederick R.M. Barnes. *Sci. of Comp. Prog.*, 77 (6). 727-740, 2012.
3. **Alting Barriers: Synchronisation with Choice in Java using JCSP**. Peter H. Welch, Neil C. C. Brown, James Moores, Kevin Chalmers, and Bernhard H. C. Sputh. *Concurrency and Computation: Practice and Experience*, 22:1049-1062, 2010.
4. **A Process-oriented Architecture for Complex System Modelling**. Carl G. Ritson and Peter H. Welch. *Concurrency and Computation: Practice and Experience*, 22:965-980, 2010.
5. ** **Santa Claus: Formal Analysis of a Process-oriented Solution**. Peter H. Welch and Jan B. Pedersen. *ACM Trans. Programming Languages and Systems*, 32(4):14:1-14:37, 2010.
6. **To Boldly Go: an occam-pi Mission to Engineer Emergence.** Peter H. Welch, Kurt Wallnau, Adam T. Sampson and Mark Klein. *Natural Computing*, 11 (3), pp 449-474, 2012.

4. Details of the impact

Our work has had direct impact through providing new mechanisms for software development in a number of sectors, including semiconductor design, large-scale real time systems, healthcare and space. Testimonial statements are provided as evidence in each of these sectors, and download data are also presented to show the breadth of take up. The impact has been amplified by others who have contributed to the libraries and to documentation, as well as through developing their own versions of the JCSP library in different programming languages.

JCSP in Design and Build of a New Compute Platform: Dr. Aly Syed (NXP Semiconductors, the Netherlands) [S2] has used JCSP to simulate the working of a parallel distributed system of sensors and actuators in the design and development of a new type of compute engine for these nodes. Once the design was validated, the system was migrated to FPGA and a real wireless communication system without introducing logic errors. He writes: "*we were able to concentrate on designing and building a new compute platform with some sensors and actuators attached to them, using JCSP to simulate communication. This was a great help as JCSP made it easy to program parallel behaviour in our system and we did not have to spend a lot of effort on this part of the problem. Out of this work, NXP Semiconductors have filed a* ***European patent application****, number EP12/190957.6, entitled: An Interpretation Engine and Associated Method. This patent application cannot be made public at this time but in about 1 year [2014], it will be available … . "*

JCSP in Large Scale High Concurrency Projects: Dr. Rick Beton (Big Bee Consultants Ltd, UK), [S3] notes the importance of JCSP in his company's solutions for customer projects: *"We have been able to use JCSP in some very effective ways in a range of projects. The largest design-in was in the London Congestion Charge control centre. Here, JCSP orchestrates the Java server that collected the feeds from nearly 1000 cameras, with 7 TCP connections per camera. This task is highly concurrent and high volume (typically 1.5M records captured per day). … JCSP*

has been found to be an effective way of synchronising and exchanging data. In my work, JCSP is now also being used from Scala and other JVM-based languages."

Formalising Interfaces and Automated Testing: Dr. Marcel Boosten (Philips Healthcare, the Netherlands) [S4] writes: *"Within Philips Healthcare, the **ideas** that came out of Kent primarily (but also other Universities present at the WoTUG conferences in the years 1998..2006) have inspired [us] to look at communication protocols and especially interfaces between components / systems from a different perspective. For years, interfaces, especially their dynamic behavior, between complex software driven systems were a worry. … Since 2007 or so, Philips Healthcare cooperates with [start-up] Verum and uses their [CSP] techniques especially for formalizing interfaces and automatically testing them against formal specification, but also for software development of control software. Philips Healthcare, I believe, was the first 'big' contract for Verum, allowing them to find investors and really kick-off. So, yes, I do believe that the work of your University has paid off via its ideas, also for Philips Healthcare - as described in the history above."*

Competitive Advantage in the Space Industry: Dr. Barry Cook (CTO, 4Links Ltd) [S5] writes: *"4Links Limited has used the CSP/occam model as the basis for its design of a very successful range of test equipment for the Space industry. We are seen as a world leader in this field and export … our products … to more than twenty countries. We gain significant competitive advantage in our complex, high performance, mixed hardware/software designs from use of the CSP/occam model. The University of Kent has provided a focus in exploring ideas that guide us, validated by their producing proven implementations in KRoC, JCSP etc. as well as numerous examples."*

Sir Charles Antony Richard Hoare (FRS, Microsoft Research) [S1] writes generally of the work: *"The achievements of the Kent research group in Communicating Process Architectures is unique in its class. The team has combined two successful theories of concurrency, CSP and the π-calculus, into a single language. It has implemented the combination with worldbeating efficiency. And it continues to serve a group of appreciative industrial users in UK and Europe."*

Separate from these testimonials, evidence for the impact of our work is recorded in the published research and activities promoted by others on the web. One of the most active communities focuses on concurrency education via process-oriented design (and occam-pi), using robotics and other applications running on small low-memory low-powered devices (such as the Arduino and Raspberry Pi) for hands-on practice. This was started by six PhD students at Kent, four of whom have graduated and left for employment (USA, Denmark and UK), and who manage the supporting website [U3], mailing list and develop the technology. This technology is based around the *Transterpreter* [U4, U5], a virtual machine for occam-pi byte code, that originated at Kent and shares the compiler and application libraries from our occam-pi tool-set (KRoC). The websites contain a wealth of information for new users, especially for students and ultimately schoolchildren, including an online book (*"Plumbing for the Arduino"*, 93 pp, [U6]) and LGPL-licensed software.

As evidence of the sustainability of JCSP, there have been important third party contributions to the JCSP library from the University of Aberdeen (Alastair Allen, Bernhard Sputh) and Napier University (Prof. Jon Kerridge, Kevin Chalmers) supporting the safe termination of networks [U7] and a significant revision and upgrade to the distributed and mobile channel/process package [U8]. The ideas and mechanisms within JCSP have inspired and challenged similar developments for other programming languages, including the CSP library for Scala (*Communicating Scala Objects* by Bernard Suffrin, Worcester College, Oxford [U9]), three CSP libraries for Python (*PyCSP* by Rune Friborg at al, Univ. of Twente, the Netherlands; *python-csp* by Sarah Mount et al., Univ. of Wolverhampton; *Hydra* by Waide Tristram et al, Rhodes University, SA – papers downloadable from CPA 2009 [U10]) and the *Groovy* version of JCSP [U11] (by Jon Kerridge, Napier Univ.).

Independently written tutorials on JCSP have been written for the *IBM DeveloperWorks* Technical Library [U12, U13, U14]. These were authored by Abhijit Belapurkar, a senior technical architect at Infosys Technologies Limited, Bangalore. The main tutorial [U13] has had 12,092 views to date, and the main tutorial in the Japanese version [U15] had 2,503 views.

Evidence of the use of two of the main products of our research (the KRoC occam-pi toolset and the JCSP library and documentation) lies in download/update counts and citations. Both products are made available under the GPL and, where possible, L-GPL open source licenses. Both are obtainable from more than one place and in more than one form (raw download or SVN updates).

From the original KRoC website [U16] and in the period from January 2008 through July 2013, there were approximately 6,500 raw downloads of KRoC. Over the same period, there were over 325,000 visits to the site from over 27,000 unique IP addresses. However, access through Kent *CSProjects* (U17) was established in October 2007 and this has since become the main source: from January 2008 through July 2013, over 12.5 million visits (48,598 unique IPs) were made and 118,974 SVN *updates* recorded (either new user downloads or existing user updates to get latest versions). In April 2013, KRoC sources moved to Git [U18] and we have no figures for downloads or updates from there. Since June 2009, another website [U19] was set up by Dr. Adam Sampson (an ex-Kent PhD and research associate, now a lecturer at the University of Abertay, Dundee), with collected links to KRoC occam-pi, JCSP and all the CSP libraries for other languages (C, C++, Scala, Python), and XC (the XMOS CSP extension to C). In the years 2010, 2011 and 2012, this site attracted 7,763, 6,619 and 26,280 unique IP address visitors respectively. For 2013 to the end of July, there have been 22,409 different visitors. This shows continuing healthy growth.

JCSP version 1.1 (a major revision) was released in October 2007. In the period from January 2008 through July 2012, there were over 15,300 downloads of JCSP from the main JCSP website [U20]. Over the same period, there were almost over 2.4 million visits to the site from over 60,000 unique IP addresses. The JCSP packages and documentation are also mirrored at a Codehaus *git* repository [U21]), for which it has been impossible to obtain any statistics.

In summary, JCSP and KRoC occam-pi have proven impact in the software development industry across a variety of sectors (semiconductor, testing, space, real-time), as well as broad take up through open source distribution. Both technologies have attracted third-party contributions of code and documentation, as well as triggering similar developments for other programming languages.

5. Sources to corroborate the impact

Corroborating statements provided by

[S1] http://www.cs.kent.ac.uk/research/REF2014/Hoare.txt
[S2] http://www.cs.kent.ac.uk/research/REF2014/Syed.txt
[S3] http://www.cs.kent.ac.uk/research/REF2014/Beton.txt
[S4] http://www.cs.kent.ac.uk/research/REF2014/Boosten.txt
[S5] http://www.cs.kent.ac.uk/research/REF2014/Cook.txt

References from sections 2 and 4

[U1] https://www.cs.kent.ac.uk/research/groups/plas/wiki/IFIP_WG24
[U2] http://www.cs.kent.ac.uk/projects/ofa/sei-cmu/
[U3] http://concurrency.cc/ [U4] http://www.transterpreter.org/
[U5] http://www.wotug.org/paperdb/send_file.php?num=126
[U6] http://concurrency.cc/pdf/plumbing-for-the-arduino.pdf
[U7] http://www.wotug.org/paperdb/send_file.php?num=214
[U8] http://www.wotug.org/papers/CPA-2009/ChalmersKerridge09/ChalmersKerridge09.pdf
[U9] http://www.cs.ox.ac.uk/people/bernard.sufrin/CSO/cpa2008-cso.pdf
[U10] http://www.wotug.org/paperdb/show_proc.php?f=4&num=27
[U11] http://www.soc.napier.ac.uk/~cs10/#_Toc314839707
[U12] http://www.ibm.com/developerworks/java/library/j-csp1/ (Java thread problems)
[U13] http://www.ibm.com/developerworks/java/library/j-csp2/ (Main JCSP tutorial)
[U14] http://www.ibm.com/developerworks/java/library/j-csp3/ (Advanced JCSP tutorial)
[U15] http://www.ibm.com/developerworks/jp/java/library/j-csp2/ (also j-csp1 and j-csp3)
[U16] http://www.cs.kent.ac.uk/projects/ofa/kroc/ [U17] http://projects.cs.kent.ac.uk/projects/kroc/
[U18] https://github.com/concurrency/kroc [U19] http://pop-users.org/
[U20] http://www.cs.kent.ac.uk/projects/ofa/jcsp/ [U21] http://xircles.codehaus.org/projects/jcsp