Conference Report for APSEC03

Carl Cook

The Asia-Pacific Software Engineering Conference (APSEC) for 2003 was held in Chiang Mai, Northern Thailand. The conference was a multi-track three day event, and focused on all areas of Software Engineering.

1 Preliminaries

After an interesting battle against both the University travel agents and what appeared to be the entire staff of the finance department (see Sharee's recent memo on procedures for organising staff travel), funding for this travel got through to my bank account after the conference, not before it. Nevertheless, after about 15 hours of traveling, I made it to Bangkok at about midnight local time. It was polluted and busy as usual, but I was glad to get some rest at the airport hotel.

1.1 Arrival in Chiang Mai - Midday, 9^{th} December

Things got off to a bad start. Thai airlines first admitted that there was a technical problem with the flight leaving Bangkok ten minutes after the scheduled departure time. At that point, the plane was still nowhere in sight. An hour later, and at a gate about a kilometer away, we boarded busses to take us to a plane sitting on the tarmac. Whilst someone with far superior Thai skills took my seat, and the child next to my new seat threw up a few times, the flight to Chiang Mai was otherwise pleasant.

2 Welcoming Reception

APSEC this year is being held at the Empress Hotel, Chiang Mai City. After registration, the conference started with a welcome reception, where I met Paul Strooper and David Hemer from the University of Queensland. I also met Andrew Martin from Oxford University. All three researchers were presenting formal methods papers.

3 Day One - Wednesday 10^{th} December

There were three concurrent streams throughout the three days of the technical program. The papers listed below are from sessions that I attended. The proceedings for the conference is available from my office upon request.

3.1 Opening Ceremony

The opening ceremony was very light hearted. The conference seems very well run in general, and in the opening session, we all learnt basic Thai phrases for hello, thank you, and "how much does that cost?".

3.2 Keynote Talk

Professor Takuya Katayama spoke about the "Science of Software Changes". The talk was very difficult to understand, but from what I learnt, he had come up with a formula to model how well a given program is in terms of its evolution.

3.3 Session: Maintenance and Reuse

3.3.1 Discovering Use-Cases from Source Code using the Branch-Reserving Call Graph

This talk showed how use cases could be derived from source code, using Gnu's DC reverse-polish notation calculator as an example. Why use cases would be only discovered after the software has been written is an interesting question, but I didn't dare ask it. The paper actually had no use cases in it, which was a bit of a surprise. Apparently, for the above simple example, precision and recall was high for retrieving use cases from code, but I'm not sure how matches are defined.

3.3.2 Understanding how the Requirements are Implemented in Source Code

This paper was very similar to the previous, but this time, source code is checked against documentation. The system simply looks at variable names, and matches them to paragraphs of text from the requirements documentation. Plenty of excel graphs to show the set of matched variables to requirements, however.

3.3.3 Regression Test Selection Based on Version Changes of Components

This paper discussed how to perform regression testing of closed source components. Normally, a lot of regression tests can be dismissed, as inspection of the source code will rule out parts of the system that have not changed since the previous version. Unfortunately, this is not possible with closed source components; this paper presents methods for regression testing via UML and the Object-Constraint Language, where precise specification of routines can be utilised.

3.4 Session: Software Documentation

3.4.1 Developing Relational Navigation to Effectively Understand Software

This was a really interesting paper for me, which presented a system called UQ* that is very similar to the system that we have proposed for this conference. UQ* allows the formal definition of languages (syntax and semantics), and they have built tools to navigate these relationships within a project to help programmers better understand the software. This paper splits projects into two views: artifacts and relationships. Relationships can also be user or tool defined as well as inferred. Unfortunately, the system appears to be very basic; whilst the concept seems fine, a lot of work needs to be done to get it up to speed with the claims made about its potential. I talked with the presenter of this paper afterwards, and we are going to keep in touch, as our research is very similar. During the conference I had many talks with the presenter, ruminating on our combined research.

3.4.2 The Software Concordance: Using a Uniform Document Model to Integrate Program Analysis and Hypermedia

This was another paper that presented a system similar to our own. The SC is a system that allows multiple views of java programs, and supports non-code artifacts such as documentation. The system also allows listeners to be registered for documents, and events propagate back out to the end-user tools. There are two end user tools at present - a documentation viewer and a code editor. The model of the java program is maintained by a separate semantic analyser named Fluid. A problem with this mechanism is that fluid discards much of the information needed by the code editor. Regardless, the system allows the user to jump from link to link within the project, but how much support for relationships is unclear.

4 Day Two: Thursday 11^{th} December

4.1 Keynote Speech: Professor Bertrand Meyer, "Blueprint for Real Progress in Software Engineering"

This was a very interesting talk. Bertrand talked about current good practices in SE, including design patterns, configuration management, and testing. What pleased me was that he argued the same points as we are arguing... configuration and control of SE artefacts is imperative, and the nightly build and RCS is not great but the best we have at the moment.

An interesting aside was that Bertrand claims that "diagrams in lieu of design" is a real threat to the benefits of object oriented program construction, and that refactoring can also be dangerous if substituted for design. Another interesting point was that of evaluating software engineering projects. Bertrand argued that papers where a professor takes his/her own system, a bunch of

students, and evaluates the system against a traditional system is pointless, as we always know which system will be statistically better. He suggested that instead, we should reduce the number of papers that are very similar to everybody else's, work on systems for several years that are fundamentally new and useful; the publications at the end will be worth the wait.

A final point that Bertrand made was that perhaps we should be teaching programming from the outside-in. He is currently doing just that for his students, with positive results so far. Students work on an application that shows traffic routes through a city. Initially, they just use the system, then they add new routes via xml, then they start learning how to add new functionality to the system. It is definitely well worth getting a copy of Bertrand's slides from the APSEC website.

4.2 Session: CSCW and Software Engineering

4.2.1 Session: Awareness Support in Group-based Software Engineering Education Systems

Again, this paper had several similarities to the work presented in our paper. This is not surprising... we are in the same session. The study in this paper examined feedback support between students as they develop systems in a group-based setting. The system basically logs user activity and posts the current results on a bulletin board system for all to see. The system also supports the posting of user comments, etc. 22 students were split into six groups (huh?!), and after using the system for some period of time, they were asked to complete several questionnaires. All results indicated that the users found the system useful.

4.2.2 My paper

My paper presentation went quite well. I had many questions at the end of the presentation, and I'm pretty sure that every answer was well received. I also had a couple of people come up to me afterwards and we talked for quite a while... both researchers are working on similar, but much smaller systems.

5 Excursion: Wat Phra That Doi Suthep

We took a bus up a hill to a rather impressive temple. This gave us a view over the city of Chiang Mai, which is of a similar size to Christchurch, but I'm sure there are many more people here. We also took a look at a jade shop, a jewelery shop, and an umbrella shop. All places had good shopping, but we also got to see how everything was made.

6 Conference Dinner: Khum Khantoke Restaurant

This was a great evening. We had heaps of yummy Thai food, and we also saw many traditional dances. The night ended with fireworks and mini hot air balloons that traveled for miles in the night sky... I'm sure there is a real term for these things, but I'm not sure what it is.

7 Official Conference Conclusion

The conference chairperson, Wanchai Rivepiboon from Chulalongkorn University, gave a closing address at the start of the final day. This had the usual pleasantries, but the speaker did make an apology for the earthquake in Taiwan that took out the internet connection for a day. Why the conclusion was before the final day remains a mystery.

8 Day Three: Friday

8.1 Session: Component-Based Software Engineering

8.1.1 COTS Characterization Model in a COTS-based Development Environment

This paper proposed a model to characterize reusable software components. The model was derived from previous work describing aspects of software components. The paper also proposed some selection methods for components, based on the given model.

8.1.2 An Environment for Evolutionary Prototyping Java Programs based on Abstract Interpretation

This paper presented an interesting theoretical architecture for software prototyping. The framework provides a "proxy object" for every class in the project that has not yet been developed. By using such objects, even in the early stages of top-down design, the system as a whole can be implemented, as long as all the interface declarations have been declared.

8.1.3 Case study: Reconnaissance techniques to support feature location using RECON2

This paper evaluated an existing system, RECON2, for identifying features within code. Features can be a particular method, or perhaps a higher-level functional requirement, although the example given in this paper showed evaluation of boolean expressions only. The thrust of this paper is to enhance program comprehension by extracting documentation-level information from the code. As far as I can tell, this system basically does a diff between source files and runs some test cases over the before and after programs.

9 Conclusion

This was the most useful conference that I have ever attended. I learned of two new systems that have similar goals and approaches as mine, and it will be very useful to collaborate with these researchers. They also have me new insight into research possibilities, and I'm sure I did the same for them. My paper was well received, and I am more confident about the direction of my research from conversations with many of the conference attendees. It was also great to go to a conference specifically catering for software engineers; in terms of degree of domain understanding when presenting papers, and in terms of inspiration for new ideas.

As a summary, the conference really re-centered me in terms of the general discipline of software engineering, and where my research should be going, and what contribution it makes to the overall field.

9.1 Research funding

This conference exhausted my Ph.D. funding; partial funds had to be allocated from other areas in order to participate. As no more funding exists, I have decided to focus on completing my studies, and resume publishing to major conferences after graduation, when renewed funding will be available.