The magazine of the ACCU

www.accu.org

{cvu}

Volume 23 • Issue 3 • July 2011 • £3

{cvu}

Volume 23 Issue 3
July 2011
ISSN 1354-3164
www.accu.org

Features Editor
Steve Love
cvu@accu.org

Regulars Editor
Jez Higgins
jez@jezuk.co.uk

Contributors
Giovanni Asproni, Alexander Demin, Katie Friesen, Francis Glassborow, Pete Goodliffe, Paul Grenyer, Richard Harris, Chris O'Dell, Roger Orr, Simon Sebright

ACCU Chair
Hubert Matthews
chair@accu.org

ACCU Secretary
Alan Bellingham
secretary@accu.org

ACCU Membership
Mick Brooks
accumembership@accu.org

ACCU Treasurer
R G Pauer
treasurer@accu.org

Advertising
Seb Rose
ads@accu.org

Cover Art
Pete Goodliffe

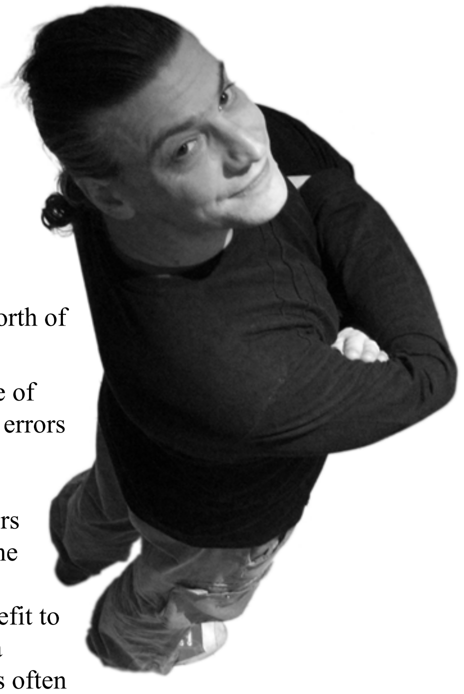
Repro/Print
Parchment (Oxford) Ltd

Distribution
Able Types (Oxford) Ltd

Design
Pete Goodliffe

accu

Screen Test



'We've seen your CV, and would like to progress to the next stage of the recruitment process. Please complete this on-line test...' The business of sifting candidates for technical jobs becomes ever more automated.

The idea, of course, is to determine the technical ability of a candidate prior to investing time and effort in actually meeting them face to face. In this way, an employer can assess many candidates apparently simultaneously, but I do question the worth of such tests.

The test content is often geared towards knowledge of technical dark corners, or the ability to spot syntax errors in (usually poorly formatted) code. Such things are popular because they are simple to *test* with a multiple choice question, and again here the answers from which to choose may differ by a single `;` or the reversal of `const` and `*`. I doubt that being able to compile code in my head is likely to be of real benefit to *any* employer. The difference between testing for a person's ability, and just trying to catch them out is often lost (although not exclusive to automated tests, of course).

It's also clear that such a test is *still* not enough; without meeting face to face it's impossible to determine whether a candidate will be good for a job or a team. It's hard enough even then. I do wonder whether the selection of people to interview 'properly' on the basis of these tests is merely filtering for people who are good at the tests. Do enough of them and the

questions become familiar.

What are your experiences with automated aptitude tests? Do you routinely ask job candidates to do them? If so, how useful do you find them?

Steve

STEVE LOVE
FEATURES EDITOR

The official magazine of ACCU

ACCU is an organisation of programmers who care about professionalism in programming. That is, we care about writing good code, and about writing it in a good way. We are dedicated to raising the standard of programming.

ACCU exists for programmers at all levels of experience, from students and trainees to experienced developers. As well as publishing magazines, we run a respected annual developers' conference, and provide targeted mentored developer projects.

The articles in this magazine have all been written by programmers, for programmers – and have been contributed free of charge.

To find out more about ACCU's activities, or to join the organisation and subscribe to this magazine, go to www.accu.org.

Membership costs are very low as this is a non-profit organisation.

DIALOGUE

SUBMISSION DATES

C Vu 23.4:	1st August 2011
C Vu 23.5:	1st October 2011
Overload 105:	1st September 2011
Overload 106:	1st November 2011

WRITE FOR C VU

Both C Vu and Overload rely on articles submitted by you, the readers. We need articles at all levels of software development experience. What are you working on right now? Let us know!

Send articles to cvu@accu.org. The friendly magazine production team is on hand if you need help or have any queries.

ADVERTISE WITH US

The ACCU magazines represent an effective, targeted advertising channel. 80% of our readers make purchasing decisions or recommend products for their organisations.

To advertise in the pages of C Vu or Overload, contact the advertising officer at ads@accu.org.

Our advertising rates are very reasonable, and we offer advertising discounts for corporate members.

COPYRIGHTS AND TRADE MARKS

It's The Thought That Accounts

Pete Goodliffe encourages us to craft great code. Using other people.

Thinking well is wise; planning well, wiser; doing well, wisest and best of all.
– Persian Proverb

I run. Every week. It's my waistline, you see. Perhaps it's a guilt thing, but I do feel I need to do something to keep it under control.

Now, let's be clear: I'm no masochist. Exercise is not my favourite thing in the world. Far from it. It definitely ranks above hot pokers being stuck in my eyes. Marginally. But there are plenty of things I'd rather do with my evenings. Many of them involve sitting down, preferably with a glass of wine.

But I know that I *should* run. It's good for me.

Is that fact alone enough to ensure I go regularly, every week, for the full distance? With no slacking or slowing of the pace?

It is not.

I dislike exercise and would gladly employ the weakest of excuses to get out of a run. 'Oh no, my running shorts have a loose thread.' 'Oh no, I have a runny nose.' 'Oh no, I'm a bit tired.' 'Oh no, my leg has fallen off.'

(Ok, some excuses *are* better than others.)

What unseen force coaxes me to continue running regularly when guilt alone can't drag me out the door? What magical power leads me on where willpower fails?

Accountability.

I run with a friend. That person knows when I'm slacking, and encourages me out of the house even when I don't fancy it. They turn up at the door, as we'd arranged before my lethargy set in. I perform the same kind of service back. I've lost count of the times that I wouldn't have run, or would have given up half-way round had I not had someone there, watching me and running alongside me.

And, as a by-product we enjoy the run more for the company and shared experience.

Sometimes we *both* don't feel like going on the run. Even if we admit it to each other, we won't let the other person off the hook. We encourage each other to push through the pain. And, once we've run, we're always glad we did it, even if it didn't feel like a great idea at the time.

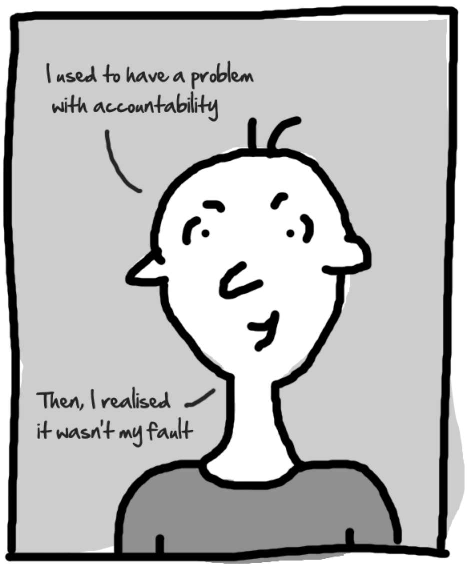
Stretch the metaphor

Some metaphors are tenuous literary devices, written to entertain, or for use as contrived segues. Some are so oblique as to be distracting, or form such a bad parallel as to be downright misleading.

However, I believe this picture of accountability is directly relevant to the quality of our code.

For all the good it does technical writers, speakers, and code prophets like myself to talk about producing good, well-crafted code, and as much as the luminaries like Uncle Bob Martin extol the (genuine) virtues of 'clean' code, and Fowler explains why we need well-factored code, it matters not one jot if, in the heat of the workplace, we can't put it into practice. If the harsh realities of the codeface cause us to shed our development morals and resort to hacking at code like uninformed idiots, what have we achieved?

We can complain about the poor state of our codebases, but who can we look at to blame?


I used to have a problem with accountability
Then, I realised it wasn't my fault

We need to bake into our development regimen ways to avoid the temptation for shortcuts, bodges and quick-fixes. We need something to lure us out of the trap of thoughtless design, sloppy, easy solutions and half-baked practices. The kind of thing that costs us effort to do, but that in retrospect we're always glad we *have* done.

The spirit is willing, but when the deadline looms, all too often the flesh is weak.

How do you think we'll achieve this?

Accountability counts

I know that in my career to date, the single most import thing that has encouraged me to work to the best of my abilities has been *accountability*, to a team of great programmers.

It's the other coders that make me look good. It's those other coders that have *made* me a better programmer.

Being accountable to other programmers for the quality of your work will dramatically improve the quality of your coding.

That is a single simple, but powerful idea.

Code++

To ensure you're crafting excellent code, you need people who are checking it every step of the way. People who will make sure you're working to the best of your ability, and are keeping up to the quality standard of the project/team you're working on.

This needn't be some bureaucratic big-brother process, or a regimented personal development plan that feeds back directly into your salary. In fact, it had better not be. A lightweight, low-ceremony system of accountability, involving no forms, lengthy reviewing sessions or formal reviews is far superior, and will yield much better results.

Most important is to simply recognise the need for such a thing; to realise that you must be accountable to other people for the quality of your code to encourage you to work at your best. To realise that actively putting

PETE GOODLIFFE

Pete Goodliffe is a programmer who never stays at the same place in the software food chain. He has a passion for curry and doesn't wear shoes. Pete can be contacted at pete@goodliffe.net

yourself into that vulnerable position of accountability is not a sign of weakness, but a valuable way to gain feedback and improve your skills.

How accountable do you feel that you currently are for the quality of the code you produce? Is anyone challenging you to produce high quality work, to prevent you from slipping into bad, lazy practices?

Accountability is worth pursuing not only in the quality of our code output, but also in the way we learn, and how we plan our personal development. It's even beneficial in matters of character and personal life (but that's a whole other magazine's column).

Making it work

There are some simple ways to build accountability for the quality of code into your development process. In one development team we found it particularly useful when the all coders agreed on a simple rule: *all code passed two eyes before entering source control*. With this as a peer-agreed rule, it was *our choice* to be accountable to one another, rather then some managerial diktat passed down from faceless suits on high. Grass-roots buy-in was key to this success of the scheme.

To satisfy the rule, we employed pair programming and/or a low-ceremony one-on-one code review, keeping each checked-in change small to make the scheme manageable. Knowing another person was going to scrutinise your work was enough to foster a resistance to sloppy practise and to improve the general quality of our code.

If you know that someone else *will* read and comment on your code, you're more likely to write good code.

This practice genuinely improved the quality of the team, too. We all learnt from one another, and shared our knowledge of the system around. It encouraged a greater responsibility for and understanding of the system.

We also ended up with closer collaboration as a result, enjoyed working with each other, and had more fun writing the code as a consequence of this scheme. The accountability lead to a pleasant, more productive workflow.

Setting the standard

When building developer accountability into your daily routine it is worth spending a while considering the benchmark that you're aiming for. Ask yourself the following questions:

How is the quality of your work judged? How do people *currently* rate your performance? What is the yardstick they use to gauge its quality? How do you think they *should* rate it?

- The software works, that's good enough.
- It was written fast, and released on schedule (internal quality is not paramount).
- It was well-written, and can be maintained easily in the future.
- Some combination of the above.

Which is seen as most important?

Who currently judges your work? Who is the audience for your work? Is it only seen by yourself? Your peers? Your superiors? Your manager? Your customer? How are they qualified to judge the quality of your handiwork?

Who *should* be the arbiter of your work quality? Who really knows how well you've performed? How can you get them involved? Is it as simple as asking them? Does their opinion have any bearing on the company's current view of your work's quality?

Which aspects of your work should be placed under accountability?

- The lines of code you produce?
- The design?
- The conduct and process you used to develop it?
- The way you worked with others?
- The clothes you wore when you did it?

Which aspect matters the most to you at the moment? Where do you need the most accountability and encouragement to keep improving?

The next steps

If you think that this is important, and something you should start adding to your work:

- Agree that accountability is a good thing. Commit to it.
- Find someone to become accountable to. Consider making it a reciprocal arrangement; perhaps involve the entire development team.
- Consider implementing a simple scheme like the one described above in your team, where every line of code changed, added or removed must go past two sets of eyes.
- Agree on how you will work out the accountability – small meetings, end of week reviews, design meetings, pair programming, code reviews, etc.
- Commit to a certain quality of work, be prepared to be challenged on it. Don't be defensive.
- If this happens team-wide, or project-wide then ensure you have everyone's buy-in. Draft a set of team standards or group code of conduct for quality of development.

Also, consider approaching this from the other side: can you help someone else out with feedback, encouragement, and accountability? Could you become another programmer's moral software compass?

Often this kind of accountability works better in pairs of peers, rather than in a subordinate relationship.

Conclusion

Accountability between programmers requires a degree of bravery; you have to be willing to accept criticism. And tactful enough to give it well. But the benefits can be marked and profound in the quality of code you create.

Questions

- How are you accountable to others for the quality of your work?
- What should you be held accountable for?
- How do you ensure the work you do today is as good as previous work?
- How is your current work teaching you and helping you to improve?
- When have you been glad you kept quality up, even when you didn't feel like it?
- Does accountability only work when you *chose* to enter into an accountability relationship, or can it effectively be something you are *required* to do? ■

An Analysis of a Game of Divisions

The Baron's student acquaintance analyses the game.

The Baron's most recent game consisted of a series of some six wagers upon the toss of an unfair coin that turned up one side nine times out of twenty and the other eleven times out of twenty at a cost of one fifth part of a coin. Sir R----- was to wager three coins from his purse upon the outcome of each toss, freely divided between heads and tails, and was to return to it twice the value he wagered correctly.

Clearly, our first task in reckoning the fairness of this game is to figure Sir R-----'s optimal strategy for placing his coins. To do this we shall need to know his expected winnings in any given round for any given placement of his coins.

Let us suppose that sir R----- knew that there was a probability p that the coin was biased towards heads. If he wagered a sum x upon heads he should therefore have expected to win

$$\begin{aligned}&-3+p\times\left[\tfrac{11}{20}\times 2\times x+\tfrac{9}{20}\times 2\times(3-x)\right]\\&+(1-p)\times\left[\tfrac{9}{20}\times 2\times x+\tfrac{11}{20}\times 2\times(3-x)\right]\end{aligned}$$

since the formula in the first pair of square brackets yields his expected winnings should the coin have been biased for heads and that in the second his winnings should it have been biased for tails.

Rearranging this yields

$$-3+\left[\tfrac{9}{10}+\tfrac{2}{10}p\right]\times x+\left[\tfrac{11}{10}-\tfrac{2}{10}p\right]\times(3-x)$$

If Sir R----- had no evidence as to which way the coin is biased then p is trivially equal to one half and his expected winnings should have been 3 coins, no matter how he divided his wager.

If, however, he had evidence that it was biased towards, or indeed away from, heads then p will differ from one half by some quantity, say q, and his expected winnings should have been

$$\begin{aligned}&-3+\left[\tfrac{9}{10}+\tfrac{2}{10}\times(\tfrac{1}{2}+q)\right]\times x+\left[\tfrac{11}{10}-\tfrac{2}{10}\times(\tfrac{1}{2}+q)\right]\times(3-x)\\&=-3+3+\tfrac{4}{10}\times q\times x-\tfrac{6}{10}\times q\\&=\tfrac{2}{5}\times q\times(x-1\tfrac{1}{2})\end{aligned}$$

From this it is evident that if Sir R----- believed the coin to be biased towards heads, and that q was consequently positive, he should have wagered as much as possible on heads if he desired the greatest possible expected winnings.

Likewise, if he believed it to be biased towards tails, and that q was therefore negative, he should have wagered thusly as little as possible.

Now, because of this we need not figure the actual probability that the coin is biased towards heads, just which side it is more likely biased towards. This we can do by simply observing which side has come up the more often.

To figure the expected winnings over the course of the game we should therefore assume that Sir R----- would have wagered all of his coins on whichever side had come up the more often.

If we also assume that he divided his coins equally if each side had come up the same number of times then we are free to declare which side the coin is biased towards before we perform our calculations for, whichever side we choose, the strategy and consequently the workings are the same. We shall therefore assume that the coin is biased towards heads.

The number of ways in which we can observe some specific numbers of heads and tails in a series of tosses of a coin is governed by Pascal's triangle

```
                    1
                 1     1
              1     2     1
           1     3     3     1
        1     4     6     4     1
     1     5    10    10     5     1
```

Here we may consider a move down and to the left as representing a toss of heads and down and to the right as a toss of tails. The values in each row are equal to the sum of the values to the left and right of it in the preceding row.

I made these observations known to the Baron when he described to me the rules of this game, but I fear I may not have done so with sufficient clarity.

Now, there is a simple formula, known as a combination, that expresses these values in terms of the row, say n, and the position in that row, say r, both of which we start counting from zero.

$${}^{n}C_{r}=\frac{n!}{r!\times(n-r)!}$$

Note that the exclamation marks do not indicate that we should perform the calculation with exceptional vigour, but rather stand for the product of every integer from one up to and including that to its left.

For example, consider the second value in the fifth row

$${}^{5}C_{2}=\frac{(1\times 2\times 3\times 4\times 5)}{(1\times 2)\times(1\times 2\times 3)}=\frac{4\times 5}{1\times 2}=\frac{20}{2}=10$$

From here it is but a small step to figuring the probability of observing r heads in n tosses; we simply multiply the number of ways we might do so by the probability of one such example

$$p_{r}^{n}={}^{n}C_{r}\times(\tfrac{11}{20})^{r}\times(\tfrac{9}{20})^{n-r}$$

In the first round, Sir R-----'s expected winnings would trivially have been zero, for he would have had no knowledge whatsoever of how the coin was biased.

If Sir R----- had already played an odd number of rounds, let us say $2n$-1, then one side of the coin must surely have had come up the more often and his expected winnings should consequently be

$$\begin{aligned}E_{2n-1}&=\left(\sum_{0\le r\le n-1}p_{r}^{2n-1}\times(\tfrac{9}{20}\times 3-\tfrac{11}{20}\times 3)\right)\\&\quad+\left(\sum_{n\le r\le 2n-1}p_{r}^{2n-1}\times(\tfrac{11}{20}\times 3-\tfrac{9}{20}\times 3)\right)\\&=\left(\sum_{n\le r\le 2n-1}\tfrac{3}{10}\times p_{r}^{2n-1}\right)-\left(\sum_{0\le r\le n-1}\tfrac{3}{10}\times p_{r}^{2n-1}\right)\end{aligned}$$

Note that here the capital sigmas stand for the sum of the expressions to their right over the integer values satisfying the conditions beneath them.

Now, if Sir R----- had played an even number of games, let us say $2n$, there is the possibility that both sides of the coin had turned up an equal number of times, giving expected winnings of

$$\begin{aligned}E_{2n} &= \left(\sum_{0\le r\le n-1} p_r^{2n} \times \left(\tfrac{9}{20}\times 3 - \tfrac{11}{20}\times 3\right)\right) + \left(p_n^{2n}\times 0\right)\\ &\quad + \left(\sum_{n+1\le r\le 2n} p_r^{2n} \times \left(\tfrac{11}{20}\times 3 - \tfrac{9}{20}\times 3\right)\right)\\ &= \left(\sum_{n+1\le r\le 2n} \tfrac{3}{10}\times p_r^{2n}\right) - \left(\sum_{0\le r\le n-1} \tfrac{3}{10}\times p_r^{2n}\right)\end{aligned}$$

There is a surprising relationship between these two formulae, which can be revealed by multiplying the first by unity!

Specifically, we shall consider the result of the expression

$$E_{2n-1}\times\left(\tfrac{9}{20}+\tfrac{11}{20}\right)$$

On the face of it this might not seem particularly illuminating, but if we consider the relationship between our expectation formulae and Pascal's triangle, all will become clear.

First, we define

$$e_r^n = \tfrac{3}{10}\times p_r^n$$

We can then arrange these in a triangle whose rows represent our expectation formulae.

$$\begin{array}{ccccccccccc} & & & & & 0 & & & & & \\ & & & & -e_0^1 & & e_1^1 & & & & \\ & & & -e_0^2 & & 0 & & e_2^2 & & & \\ & & -e_0^3 & & -e_1^3 & & e_2^3 & & e_3^3 & & \\ & -e_0^4 & & -e_1^4 & & 0 & & e_3^4 & & e_4^4 & \\ -e_0^5 & & -e_1^5 & & -e_2^5 & & e_3^5 & & e_4^5 & & e_5^5 \end{array}$$

Obviously we can't figure the value of an element by simply adding those either side of it in the row above, but we can exploit the relationships between these elements.

Let us begin by considering those elements of E_{2n-1} and E_{2n} for which there are fewer heads than tails. (See Figure 1.)

Note that to figure the value of an element in the second row, we must multiply those either side of it in the first row by the probability on the line between it and the element in question before adding them.

It is self-evident that this is equivalent to those terms in our product for which there are fewer heads than tails.

Through an identical argument we find that the same is true for terms where the heads outnumber the tails.

All that remains, therefore, is to figure the term in our product which corresponds to that in E_{2n} for which there are an equal number of heads and tails.

Now the formula for combinations is symmetric in r about ½n, which is plain from inspection of either the formula itself or of Pascal's triangle. As a consequence we have

$$\begin{aligned}&\tfrac{9}{20}\times e_n^{2n-1} - \tfrac{11}{20}\times e_{n-1}^{2n-1}\\ &\quad= \tfrac{9}{20}\times\tfrac{3}{10}\times p_n^{2n-1} - \tfrac{11}{20}\times\tfrac{3}{10}\times p_{n-1}^{2n-1}\\ &\quad= \tfrac{9}{20}\times\tfrac{3}{10}\times {}^{2n-1}C_n\times\left(\tfrac{9}{20}\right)^{n-1}\times\left(\tfrac{11}{20}\right)^{n}\\ &\qquad - \tfrac{11}{20}\times\tfrac{3}{10}\times {}^{2n-1}C_{n-1}\times\left(\tfrac{9}{20}\right)^{n}\times\left(\tfrac{11}{20}\right)^{n-1}\\ &\quad= \tfrac{3}{10}\times {}^{2n-1}C_n\times\left(\tfrac{9}{20}\right)^{n}\times\left(\tfrac{11}{20}\right)^{n}\\ &\qquad - \tfrac{3}{10}\times {}^{2n-1}C_n\times\left(\tfrac{9}{20}\right)^{n}\times\left(\tfrac{11}{20}\right)^{n}\\ &\quad= 0\end{aligned}$$

which, to our very great fortune, is exactly the expected winnings we require!

We can hence assert with full confidence that

$$E_{2n-1} = E_{2n}$$

and Sir R-----'s expected winnings over the entire contest were determined by

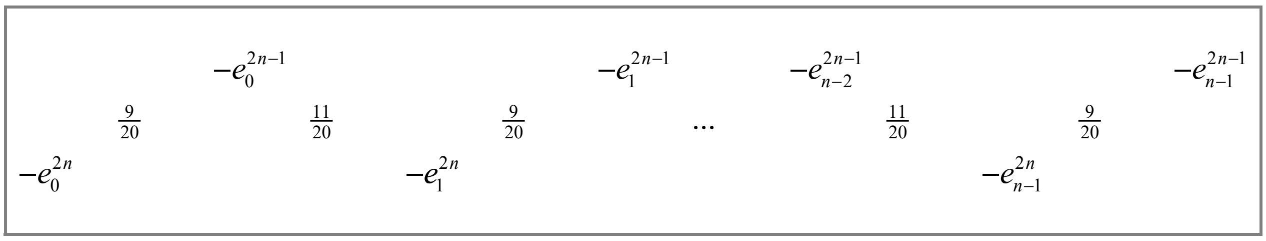
$$2\times E_1 + 2\times E_3 + E_5$$

With sufficiently careful arithmetic, we can show this to be equal to

$$\begin{aligned}2\times\tfrac{3}{100} + 2\times\tfrac{897}{20{,}000} + \tfrac{447{,}009}{8{,}000{,}000} &= \tfrac{1{,}644{,}609}{8{,}000{,}000}\\ &= \tfrac{1{,}600{,}000}{8{,}000{,}000} + \tfrac{44{,}609}{8{,}000{,}000}\\ &= \tfrac{1}{5} + \tfrac{44{,}609}{8{,}000{,}000}\end{aligned}$$

The game is therefore slightly biased in Sir R-----'s favour and I should consequently have had no compunction whatsoever in recommending that he take up the Baron's wager! ■

Figure 1



A Game of One Against Many

Baron Muncharris is offered a wager.

Sir R-----! Might I presume that you are of a mood for a glass and a wager?

Good fellow! Stout fellow! Come join me in a draught!

I suggest a game oft played in the fair land of Lyonesse which, contrary to historical record, has not forever sunk beneath the waves, but has rather through some oversight been quite forgot in all our atlases.

I had been invited to banquet with the King of that fair and fecund realm, but arrived to find his court in disarray; the Queen had been stolen away by the foul Lord Maleagant!

Naturally, I immediately put myself at my host's disposal and set off to rescue his fair bride.

Being a coward of the lowest order, Maleagant put not his security in the honest strength of arms but rather in the utter impenetrability of his fortress, built atop a vertiginous tower of rock off the coast of Kernow by the most deviously minded Masons in the land.

That he put his faith in stone over flesh and blood ultimately worked to my advantage; once inside his fortress I had no trouble seeing off his guards and making my way to his throne room.

To no great surprise he too proved himself an unworthy adversary. In the work of moments I had scored a deep cut upon his forehead, blinding him with his blood and denying him the ability to prosecute a duel.

As might be expected from a fellow of such calibre he took his defeat poorly. As I left with his hostage I heard him crying out 'curses be upon you!' and 'plague your eyes!'

The celebration of the return of the Queen was lavish beyond measure and I spent many a contented hour in wine and wager with the good folk of the court.

Upon my departure I was rewarded with a lavish bowl which, albeit a little wine stained, has proven a most decorous receptacle for fruits and nuts upon my dining table.

But here I have not told you of the nature of their sport!

Your goal is to cast a greater score upon your die than I do upon mine. Your stake will be seven coins and your prize, should you best my score, shall be thirteen.

You shall cast first and, should you be dissatisfied with your score, may elect to cast the die again for the price of a further coin. If this neither meets your satisfaction you may have a third cast for two more coins and so on and so forth, with each cast costing one coin more than the last, until you are content.

I shall have but a single cast of my die once you have declared that you are satisfied with your score and, if I cannot equal or best you, you shall have your prize.

If you do not relish the prospect of trying to best a die that I have not yet cast and you are willing to stake one additional coin I shall instead cast before you commence your play.

Upon learning the rules of this game, that damnable cur of a student of whose acquaintance I am cursed announced with uncharacteristic candour that he had come to realise that his work was backward. Now I have long since recognised that he stands some several leagues behind gentlefolk in every matter of worth, but I was so struck by his honesty that the notion to remark on it quite escaped me.

But you can surely have no interest in news of that wretch's slowly dawning awareness of his station; recharge your glass and name your sport! ■

Listing 1

```
int
roll()
{
  return 1 + int(6.0 * double(rand()) /
     (double(RAND_MAX)+1.0));
}

void
play_first()
{
  static const int stake = 7;
  static const int prize = 13;

  int balance = -stake;
  int cost = 0;
  int r_roll;

  char again = 'Y';
  while(toupper(again)=='Y')
  {
    balance -= cost++;
    r_roll = roll();

    std::cout << "You rolled a "
       << r_roll << "! ";
    std::cout << "Balance = "
       << balance << std::endl;

    again = '\0';
    while(toupper(again)!=
       'Y' && toupper(again)!='N')
    {
      std::cout << "Roll again for "
         << cost << " coins? [Y/N] : ";
      std::cin  >> again;
      std::cin.ignore(std::numeric_limits<int>
         ::max(), '\n');
    }
  }

  const int b_roll = roll();
  std::cout << "The Baron rolled a "
     << b_roll << "!" << std::endl;

  if(b_roll<r_roll) balance += prize;

  if(balance>=0.0) std::cout << "You won "
     <<  balance;
  else if(balance<0.0) std::cout << "You lost "
     << -balance;
  std::cout << " coins!" << std::endl;
}
```

BARON MUNCHARRIS

In the service of the Russian military Baron Muncharris has travelled widely in this world, and many others for that matter, defending the honour and the interests of the Empress of Russia. He is renowned for his bravery, his scrupulous honesty and his fondness for a wager.

All from a Telephone Call

Francis Glassborow reflects on the origins of ACCU.

During my life I have noticed several times when I have done something that was, at the time, apparently of little importance but that hindsight shows to have been a critical turning point in my life. This article is about one such event and some of the consequences.

In early summer of 1988 I was browsing through the small ads column of *PC World* when I noticed an advert for the C User Group (UK). Membership was £10 for six issues of its newsletter, C Vu. I had recently tried a little programming in C, adding it to various other programming languages I had already taught myself.

I thought that the user group might be interesting and possibly some of my more able pupils would be interested. I sent off a cheque for £10 and a few days later received a copy of issue 4. It was fairly typical of such publications and included quite a lot of material reprinted from elsewhere.

The Summer term ended, the Summer holidays came and went and my time as a teacher was clearly coming to an end. Stress related ill health meant that I was going through the process of being retired early. Some time in early November I came across the copy of C Vu and wondered why I had not had another one. I thought that probably the group had folded as so many enthusiast groups do. However, I noticed that there was a contact name and telephone number under the editorial and decided to ring. The telephone call was fortuitous because the organiser, Martin Houston, was thinking about arranging a meeting of the members to decide what to do with CUG(UK) (the (UK) part was important to distinguish it from the US based C Users Group that had a publication called *The C Users Journal* which eventually became a commercial publication that took over the CUG).

I learnt from the phone call that I had not actually missed any issues of C Vu and that issue 5 was going to press in the next couple of weeks. More important was that I had a conversation with Martin Houston and because of that decided to go to the meeting when it was arranged.

It was clear to me that C Vu needed original content and a regular publication schedule

I am certain if issue 5 had come out a few weeks earlier or I had delayed the telephone call a couple of weeks I would not have gone to that meeting. However, I made the call and learnt of the meeting, so despite the things happening in my professional life I turned up.

There were a couple of dozen people at the meeting, which was a pretty good turnout for a Saturday afternoon. The main topic was what we should do with CUG(UK). It was clear that there was some enthusiasm. I suggested that we had about the right number of people to form a committee and parcel out the various necessary jobs. I finished up as Membership Secretary.

It was clear to me that C Vu needed original content and a regular publication schedule. I suggested that we make it a quarterly. That was agreed as were deadlines and publication dates for the year.

Over the next few months the membership grew quite considerably mainly because I managed to get free stand space at a number of computer events. However the second of the scheduled issues of C Vu was late and as we had arranged an AGM to confirm a constitution, elect officers and a committee, I did not fancy turning up to the meeting and explaining to the membership why they had not yet had the summer issue of C Vu (Volume 2 issue 2). I can remember a somewhat acrimonious telephone call to Martin (who was now the Chairman) the consequence of which was that he hastily photocopied enough copies give to the attendees. The first AGM started late because Martin was still stapling the copies when we were scheduled to start.

A single telephone call directed my life down paths that I would not have imagined 23 years ago

I then learnt that the next issue of C Vu would be very late because the editor would not be in the country at the time it was scheduled to be published. Rather than complain I made the Committee an offer that they could hardly refuse. I stuck my neck out and offered to become editor of C Vu. I guaranteed that there would be at least 32 pages of original material even if I had to write it all myself. Furthermore I upped the frequency to bi-monthly.

In fact the smallest issue I ever published was 48 pages in A5 format and over the following years the font size went down and the line spacing shrank so that I could get the material into an issue without too high a page count.

When I learnt something about C++ it seemed clear to me that I should be publishing articles on that as well. I then learnt of a recently formed group called the European C++ User Group that intended to run high quality low cost conferences. In the event only one was ever held (in Munich). I attended that first conference as a speaker and gave a truly awful presentation on the uses of classes with only private constructors and destructors. The subject was fine, I just had no experience of presenting a paper at an academic conference.

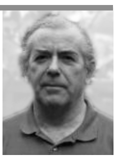
When EC++UG clearly could not maintain impetus CUG(UK) absorbed its membership and soon after re-badged itself as 'The Association of C and C++ Users'.

I then learnt that a group of enthusiasts were intending to create a Borland C++ User Group with a publication called *Overload*. I saw this as a golden opportunity to develop the C++ side of ACCU. After some fairly difficult negotiations we persuaded the Borland enthusiasts to join ACCU and make *Overload* a second ACCU publication.

I still had in the back of my mind that EC++UG had intended to run conferences, so when WG21 (ISO C++ Standards Work Group) were due to hold their second London meeting (1997) I saw that as an opportunity to hold a small, low cost conference. It was a two day event in Oxford Town Hall with several tracks on the first day and a single track on the Saturday. I do not recall the details of the Friday but the speakers on Saturday were Bjarne Stroustrup, Dan Saks, Tom Plum and Bill Plauger. The event was a great success and has steadily grown over the years to become the wonderful event that it is today.

Now all that, and much more that I have left out, came from a single timely telephone call. But much more happened to me personally. As a result of the inaugural meeting I met Neil Martin and learnt about the BSI panels

FRANCIS GLASSBOROW



Since retiring from teaching, Francis has edited C Vu, founded the ACCU conference and represented BSI at the C and C++ ISO committees. He is the author of two books: *You Can Do It!* and *You Can Program in C++*.

and started attending the C and C++ Panel meetings. That led me to attend the first London meeting of WG21 where I met Bjarne Stroustrup. From that grew my involvement with both WG14 (C) and WG21 (C++). I have met many people and gone to many places as a result of that involvement.

As a result of my being editor of C Vu I was offered a column in *.EXE Magazine* for which I was actually paid. That column expanded from an original 500 words (and the editor, Will Watts, insisted that it was 500+/−3) and eventually grew to 2000 words. The early days of that column were a great learning experience. Delivering 500 words every month was an excellent discipline and throughout the 100 columns that I wrote, I only had the editor require a rewrite once. When the news arrived that *.EXE* was going out of publication I had just delivered my 100th column (yes, I take some pride in that, as I was the only columnist for *.EXE* that managed 100 columns).

As a result of my writing for *.EXE* I was asked to present training courses on C and C++.

I suspect that some of the detail above may be slightly wrong and if any one likes to correct it I would be very happy for them to do so. I would also love to know about CUG(UK) before 1988. I know the group was founded in 1987 and I think that there was a prior group but have never been able to find out more. Perhaps our new Treasurer knows a little more than I do about those early days. (It is a sign of the strength of ACCU that someone from its first year of existence should step forward to take office 24 years later.)

I wonder how many other people have had their lives changed by ACCU

A single telephone call directed my life down paths that I would not have imagined 23 years ago. I am grateful for all the help over the years from too many people to mention. I know that CUG(UK)/ACCU changed my life and has given me a great deal in my retirement from teaching. All I can say is that without that phone call my life over the last two decades would have been very different.

Finally, I wonder how many other people have had their lives changed by ACCU. I hope there are some and that the changes have been positive ones. If ACCU changed your life please put fingers to keyboard and share with us. ■

If you read something in C Vu that you particularly enjoyed, you disagreed with or that has just made you think, why not put pen to paper (or finger to keyboard) and tell us about it?

JOIN ACCU

You've read the magazine. Now join the association dedicated to improving your coding skills.

ACCU is a worldwide non-profit organisation run by programmers for programmers.

Join ACCU to receive our bi-monthly publications *C Vu* and *Overload*. You'll also get massive discounts at the ACCU developers' conference, access to mentored developers projects, discussion forums, and the chance to participate in the organisation.

What are you waiting for?

How to join
Go to www.accu.org and click on Join ACCU

Membership types
Basic personal membership
Full personal membership
Corporate membership
Student membership

professionalism in programming
www.accu.org

cqf.com

CERTIFICATE IN QUANTITATIVE FINANCE CQF

Expand Your Mind and Career

Designed by quant expert Dr Paul Wilmott, the CQF is a practical six month-part time course that covers every gamut of quantitative finance, including derivatives, development, quantitative trading and risk management.

Find out more at **cqf.com**.

ENGINEERED FOR THE FINANCIAL MARKETS

An Introduction to the Windows Presentation Foundation with the Model-View-ViewModel (Part 1)

Paul Grenyer introduces core patterns for WPF development.

After three wonderful years working with Java I am back in the C# arena and amazed by how things have changed. When I was working with C# previously it was with .Net 1.1 and as I return .Net 4 is ready to go. I started a new contract and my client suggested that to get ahead of the game I should learn Windows Presentation Foundation (WPF), the latest Microsoft framework for creating Windows desktop (and web) applications. It replaces the likes of Windows Forms on the desktop. Two of the major features of WPF are that it is rendered entirely on a computer's graphics card and separates presentation from presentation logic.

Manning is my preferred technical book publisher, so I bought the PDF version of *WPF In Action with Visual Studio 2008* [1] and read it on my Kindle. It is a great introduction to producing Graphical User Interfaces (GUIs) with WPF, but I later discovered that although MODEL-VIEW-VIEWMODEL (MVVM) is covered, the detail is not great. The MVVM pattern is similar to Martin Fowler's PRESENTATION MODEL [2], but where the presentation model is a means of creating a UI platform-independent abstraction of a view, MVVM is a standardised way to leverage core features of WPF to simplify the creation of user interfaces. Fortunately there is a great *MSDN Magazine* article called 'WPF Apps With The Model-View-ViewModel Design Pattern' [3] that explains it simply and in a fair amount of detail.

Canon

Canon – Any comprehensive list of books within a field.

~ dictionary.com

To demonstrate WPF with MVVM I am going to incrementally develop a small application which allows the user to search an archive of books. The application is called Canon and the source code [4] is available for download from my website. I developed Canon using Visual Studio 2010 and .Net 4, but WPF applications can also be created with Visual Studio 2008 and .Net 3.5. I have assumed that the reader is following along.

Fire up Visual Studio and create a new WPF Application called Canon. Build and run the application to make sure everything works correctly, and you should see a very simple window like the one shown in Figure 1.

As with any normal window you should be able to minimise, maximise, resize and close it.

If you take a look at the project structure in Visual studio you'll see there appear to be just two source files, `App.xaml` and `MainWindow.xaml`.

PAUL GRENYER

An active ACCU member since 2000, Paul is the founder of the Mentored Developers. Having worked in industries as diverse as direct mail, mobile phones and finance, Paul now works for a small company in Norwich writing Java. He can be contacted at paul.grenyer@gmail.com

Adding WPF Applications to Source Control

As with most Visual Studio solutions you need to ensure you check in all source files, and not binaries or other build artefacts. Source file include the .xaml and .xaml.cs files.

Actually there are four source files. If you use the arrow next to each file to expand it you will see that each `.xaml` file has a corresponding `.cs` file: `App.xaml.cs` and `MainWindow.xaml.cs`. I'll explain the relationship between all four files shortly, but first I want to put all views into a View folder and the view namespace. In Visual Studio, create a project level folder called View and move `MainWindow.xaml` into it.

Listing 1

```
<Window x:Class="Canon.View.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow"
        Height="350"
        Width="525">
```

Listing 2

```
<Application x:Class="Canon.App"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        StartupUri="View/MainWindow.xaml">
```

`MainWindow.xaml.cs` will come along with it. Then go into `MainWindow.xaml.cs` and change the namespace from **`Canon`** to **`Canon.View`**. Then go into `MainWindow.xaml` and modify the **`x:Class`** attribute of the **`Window`** element so that it reads as shown in Listing 1.

Finally go into `App.xaml` and modify the **`StartupUri`** attribute of the **`Application`** element so that it reads as shown in Listing 2.

If you made all of these modifications correctly you should get the same window again when you build and run the application. Now is a good time to add the project to source control as we will only be adding to the structure from now on, rather than changing it.

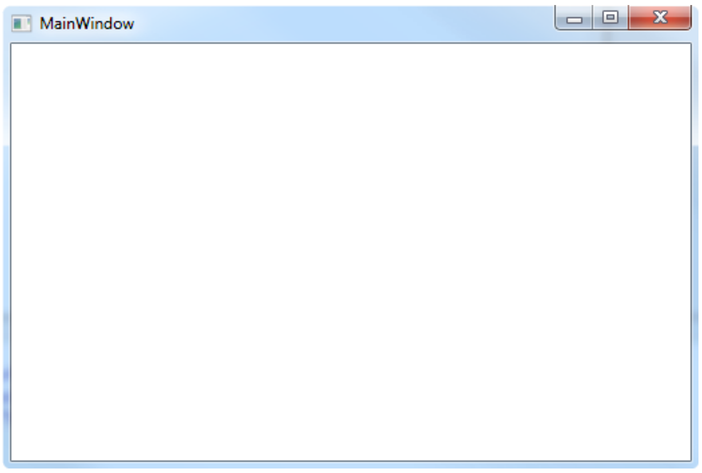


Figure 1

WPF project structure

WPF uses XAML (pronounced zammel), which stands for eXtensible Application Markup Language, to lay out User Interfaces (UIs). As we've seen all `.xaml` files have a corresponding `xml.cs` source file file. In most cases anything that can be defined in XAML can also be written in C# and vice-versa. Both files define the same class. It is not required to have both files, but in most projects there are some things you'll want to do in XAML and others in C#.

Let's start by taking a look at WPF's equivalent to **`main`**, `App.xml` and `App.xml.cs`, starting with `App.xml` (see Listing 3).

`App.xml` defines the WPF application with the **`Application`** element. The first attribute, **`x:Class`**, specifies the namespace and name of the corresponding class, which is defined in `App.xml.cs`. The next two attributes bring in the necessary namespaces for XAML and the **`StartupUri`** attribute specifies the path to the main window's XAML file. The main window is the first window displayed by the application on start-up. The **`Application.Resource`** elements are for declaring resources for use within the application. WPF In Action contains an explanation and several examples of how and when they can be useful. We'll have a look at resources later when we want to load images into the Canon application. (Listing 4)

`App.xaml.cs` defines the C# part of the application class. As you can see the class name and namespace correspond to the name and namespace defined in the **`x:Class`** attribute of the **`Application`** element. The **`App`** class is partial. Part of it is defined in XAML, including the inheritance from the **`Application`** class, and part in C#. The **`App`** class does not have any fields or values as it is currently completely defined in XAML. We'll want to change this shortly when we inject a view model.

Now that we understand how a WPF application is defined let's take a look at how a window is defined by examining `MainWindow.xaml` and `MainWindow.xaml.cs`. `MainWindow.xaml` is in the View folder we created earlier. Its name and location correspond to the value of the **`StartupUri`** attribute in the **`Application`** element in `App.xaml`. Therefore it is the first window that will be displayed. (Listing 5)

In the Window element the **`x:Class`** attribute specifies the name and namespace of the corresponding C# class and the next two elements bring in the XAML namespaces. The **`Title`** attribute specifies the title that is displayed in the window and the **`Height`** and **`Width`** attributes specify the height and width of the window. The **`Grid`** element declares the type of layout that the window will use to display its controls. I'll explain more about layouts when we create the UI controls later. (Listing 6)

`MainWindow.xaml.cs` defines the code behind the window. The class name and namespace correspond to the name and namespace defined in the **`x:Class`** attribute of the Window. The **`MainWindow`** class is partial as part of it is defined in XAML – including inheritance from the Window class – and part in C#. Inheriting from Window in the source file is therefore redundant and can be removed. The **`MainWindow`** class's only member is a constructor which calls **`InitializeComponents`**. **`InitializeComponents`** behaves in exactly the same way as it does in

Listing 3

```
<Application x:Class="Canon.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="View/MainWindow.xaml">
    <Application.Resources>

    </Application.Resources>
</Application>
```

Listing 4

```
namespace Canon
{
  public partial class App : Application
  {
  }
}
```

Listing 5

```
<Window x:Class="Canon.View.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow"
        Height="350"
        Width="525">
    <Grid>

    </Grid>
</Window>
```

Listing 6

```
namespace Canon.View
{
  public partial class MainWindow : Window
  {
    public MainWindow()
    {
      InitializeComponent();
    }
  }
}
```

Listing 7

```
<Application x:Class="Canon.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
...
</Application>
```

Listing 8

```
public partial class App
{
    public App()
    {
        new MainWindow().Show();
    }
}
```

a Windows Forms application and initialises the components defined in MainWindow.xaml.

Injecting a viewmodel

Before we can inject a view model into a view we need an instance of a view to inject it into. To get the instance of the main window you can remove the **StartupUri** attribute (Listing 7), add a constructor to the **App** class and instantiate an instance of the view there instead. To actually display the main window you need to call **Show** on it. (Listing 8)

If you run the application again now (you need to add:

```
using Canon.View;
```

of course), you will see exactly the same window. All we've done is move the creation of the first window from XAML to C#. Now we have an instance of a window to inject a view model into.

A view model need be nothing more complex than a normal class. It does not require any special base class, interfaces or members. It's just about the data. Create a project level folder called ViewModel and create the class shown in Listing 9 in it (don't forget to add it source control).

Every WPF view has a **DataContext** property of type object. This property is null unless a view model is injected into the view. When a view model is injected WPF sees that **DataContext** is no longer null and uses it. We'll cover an example of simple binding shortly. The **DataContext** property is also available within the view. This means the view knows about the view model it has, but the view model continues to know nothing about the view that's using it. You can **Inject** the view model into the view by creating an instance of it and setting the **DataContext** property on the view (see Listing 10).

If you run the application again there will be no difference. Something in the view must be bound to a property in the model to see a difference in the UI.

Listing 9

```
namespace Canon.ViewModel
{
  public class MainWindowViewModel
  {
  }
}
```

Listing 10

```
public partial class App
{
  public App()
  {
    new MainWindow
    {
      DataContext = new MainWindowViewModel()
    }.Show();
  }
}
```

Listing 11

```
public class MainWindowViewModel
{
  public string AppTitle
  {
    get
    {
      return "Canon";
    }
  }
}
```

Listing 12

```
<Window x:Class="Canon.View.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
```

Listing 13

```
<Window x:Class="Canon.View.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="{Binding AppTitle}" Height="350" Width="525">
```

A slight case of over binding

Binding is the WPF way of transferring values between a UI component and a property in a view model. It can be very simple or quite complex. Binding is explained in quite a lot of detail in WPF in Action1.

I think the best way to demonstrate binding is with a simple example. In this one we'll bind the main window's title to a property in the view model. Let's start off by adding the property to the view model, as shown in Listing 11.

Once the binding is in place the main window will display the string returned by the **AppTitle** property. To bind the window title to the property we have to modify the **Title** attribute of the **Window** element in MainWindow.xml from Listing 12 to Listing 13.

The curly braces tell WPF that we do not want to display the literal value. The key word **Binding** tells WPF we want to bind to a property in the view's view model and **AppTitle** is the name of that property. Remember that the **x:Class** attribute specifies a C# class and WPF knows it can bind to that class's **DataContext** property. If you run the application again now, you will see that the main window's title displays 'Canon' instead of 'MainWindow'.

The Canon model

Now that we have a view and a view model, we need a book model for our archive. (See Listing 14)

This simple **Book** class contains a unique nullable id for each book, its title, author, publisher and ISBN. If a **Book** instance is created with a null id it means that it is a new book. If the id has a value it means that the book has been saved before. (See Listing 15)

The book repository interface, **IBookRepository**, contains two simple persistence methods, **Search** for searching for books and **Save** for saving books. Create a project level folder called Model and add the **Book** class and the **IBookRepository** interface to it. The view model will make use of the interface so add it as a field and a constructor parameter, as shown in Listing 16.

This of course will prevent the project from building. To get it building again we need an implementing instance of **IBookRepository** to pass

SimpleBookRepository

The **SimpleBookRepository** mock object is fairly straight forward. It persists a list of books in the books list.

If the **SearchFields** method returns **true** the **Search** method knows it's found a matching book, stops iterating through the books and returns the current book. Of course there might be multiple matches, but the **Search** method only returns the first match.

The **Save** method can both update existing books and save new ones. New books are identified as having a null Id. If the book being saved has an id and is already in the book list, it is removed. This may seem a little odd. However, if the existing book is just added to book list it will be in there twice. Also remember that books are compared for equality by their ids. By the time the bottom of the **Save** method is reached the book has an id and does not exist in the book list, so it can be added without fear of duplication.

Listing 14

```
namespace Canon.Model
{
  public class Book
  {
    public long? Id { get; set; }
    public string Title { get; set; }
    public string Author { get; set; }
    public string Publisher { get; set; }
    public string ISBN { get; set; }

    public Book()
    {
      Title = string.Empty;
      Author = string.Empty;
      Publisher = string.Empty;
      ISBN = string.Empty;
    }

    public override bool Equals(object obj)
    {
      if (ReferenceEquals(null, obj))
         return false;
      if (obj.GetType() != typeof(Book))
         return false;
      return Equals((Book)obj);
    }

    public bool Equals(Book other)
    {
      if (ReferenceEquals(null, other))
         return false;
      return Equals(Id, other.Id);
    }

    public override int GetHashCode()
    {
      return Id.GetHashCode();
    }
  }
}
```

Listing 15

```
namespace Canon.Model
{
  public interface IBookRepository
  {
    Book Search(string searchTest);
    Book Save(Book book);
  }
}
```

Listing 16

```
public class MainWindowViewModel
{
  private readonly IBookRepository repo;

  public MainWindowViewModel
     (IBookRepository repo)
  {
    this.repo = repo;
  }
  ...
}
```

Listing 17

```
public class SimpleBookRepository :
IBookRepository
{
  private readonly IList<Book> books
     = new List<Book>();
  public SimpleBookRepository()
  {
    Save(new Book { Title = "Redemption Ark",
                    Author = "Alistair Reynolds",
                    Publisher = "Gollancz",
                    ISBN = "978-0575083103" });
    Save(new Book { Title = "The C++ Standard
                       Library",
                    Author = "Nico Josuttis",
                    Publisher = "Addison Wesley",
                    ISBN = "978-0201379266" });
  }
  public Book Search(string searchtext)
  {
    Book foundBook = null;
    foreach (var book in books)
    {
      if (SearchFields(book, searchtext))
      {
        foundBook = book;
        break;
      }
    }
    return foundBook;
  }

  public Book Save(Book book)
  {
    if (!book.Id.HasValue)
    {
      book.Id = getNextId();
    }
    else if (books.Contains(book))
    {
      books.Remove(book);
    }
    books.Add(book);
    return book;
  }

  private long getNextId()
  {
    long id = 0;
    foreach (var book in books)
    {
      id = Math.Max(book.Id.Value, id);
    }
    return id + 1;
  }

  private static bool SearchFields(Book book,
     string searchText)
  {
    searchText = searchText.ToLower();
    return
      book.Title.ToLower().Contains(searchText)
      ||
      book.Author.ToLower().Contains(searchText)
      ||
      book.Publisher.ToLower()
      .Contains(searchText) ||
      book.ISBN.ToLower().Contains(searchText);
  }
}
```

Listing 18

```
public partial class App
{
  public App()
  {
    new MainWindow
    {
      DataContext = new MainWindowViewModel(
         new SimpleBookRepository() )
    }.Show();
  }
}
```

Listing 19

```
<Window x:Class="Canon.View.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="{Binding AppTitle}"
        MinHeight="200"
        Height="200"
        MinWidth="450"
    Width="450">
```

Listing 20

```
<Window x:Class="Canon.View.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="{Binding AppTitle}"
        MinHeight="200"
        Height="200"
        MinWidth="450"
        Width="450">
    <DockPanel>

    </DockPanel>
</Window>
```

Listing 21

```
<DockPanel>
    <ToolBarTray DockPanel.Dock="Top">
        <ToolBar>

        </ToolBar>
    </ToolBarTray>
</DockPanel>
```

Listing 22

```
<DockPanel>
  <ToolBarTray DockPanel.Dock="Top">
    <ToolBar>

    </ToolBar>
  </ToolBarTray>
  <Grid>

  </Grid>
</DockPanel>
```

to **`MainWindowViewModel`**'s constructor. For this I knocked up a memory based mock implementation. (See Listing 17)

A **`SimpleBookRepository`** instance can be injected into the main window view mode as shown in Listing 18, and the project will build again. However there's still no change in the main window when the application is run.

Building the user interface

Next we need a UI to manipulate the model. Figure 2 shows a very simple UI that can be knocked up with a few lines of XAML.

The first thing you might notice is that the Canon UI is smaller than the default window pictured in Figure 1. This is because I modified the Window element in `MainWindow.xaml` to specify a starting height and width and a minimum height and width (Listing 19).

The window starts with a height of 200 and a width of 450 and can be expanded, however it cannot be contracted below 200 high and 450 wide. You're probably thinking that I could have just specified the minimums and you're right, I could have. However, the window would have started off somewhat bigger to begin with. Play around with different sizes until you get a feel for it.

WPF uses layouts for arranging controls on a UI. WPF layouts are a little bit like Java layouts. The window in Figure 2 consists of a **`DockPanel`** layout, a **`Grid`** layout and a **`StackPanel`** layout. A **`DockPanel`** consists of five sections, top, bottom, left, right and centre. A section is only visible if a component is put into it. For example you could have a window with a tool bar across the top, a status bar at the bottom, an explorer view to the left, a help view to the right and a text editor in the middle. The Canon UI has a toolbar at the top that holds a **`StackPanel`** (another type of layout we'll look at in a minute) which in turn holds the search box and search button. The remaining space in the **`DockPanel`**, the centre section which gets the components added last to the **`DockPanel`**, holds a **`Grid`** layout with the rest of the UI components. To create a **`DockPanel`** simply declare it as a child element of the **`Window`** element (see Listing 20).

Next we want to add a tool bar and tell the **`DockPanel`** that we want to display it at the top (Listing 21).

Tool bars usually sit within a **`ToolBarTray`** which helps give them the usual Windows look and feel and can host multiple tool bars. To insert the **`ToolBarTray`** into the **`DockPanel`** you just make it a child element. You'll notice that the **`ToolBarTray`** inherits the **`DockPanel.Dock`** attribute from its parent and uses it to specify that the **`ToolBarTray`** should be displayed at the top. Child controls inheriting properties from their parents is a common occurrence throughout WPF and makes for far less verbose XAML. WPF In Action discusses this in more detail [2]. The **`ToolBar`** is a child of the **`ToolBarTray`**.

If you run the application again now you will see that the toolbar takes over the whole client area of the window. We only want it to be a thin strip across the top and we want the rest of the area to be a **`Grid`** layout. All we have to do is add a **`Grid`** to the **`DockPanel`** (Listing 22).

I'll discuss the **`Grid`** layout in more detail once we've completed the toolbar, but I wanted you to be able to run the application and see the toolbar across the top of the window and the empty **`Grid`** in the remaining client area. You'll notice that the dock position is not specified for the **`Grid`**. You can only specify a dock position of **`Top`**, **`Bottom`**, **`Left`** or **`Right`**. Any panel or control that does not have a dock position specified is placed in the centre section of the **`DockPanel`**. Child ordering effects

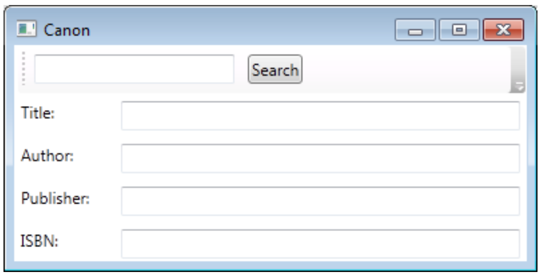


Figure 2

Listing 23

```
<ToolBarTray DockPanel.Dock="Top">
    <ToolBar>
        <StackPanel Orientation="Horizontal">
            <TextBox Margin="5,5,5,5" Width="150"/>
            <Button Margin="5,5,5,5" IsDefault="True">Search</Button>
        </StackPanel>
    </ToolBar>
</ToolBarTray>
```

Listing 24

```
<Grid IsSharedSizeScope="True">
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition SharedSizeGroup="A"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>

</Grid>
```

Listing 25

```
<Grid IsSharedSizeScope="True">
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition SharedSizeGroup="A"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Label Grid.Column="0" Grid.Row="0">Title:</Label>
    <TextBox Grid.Column="1" Grid.Row="0" Margin="5,5,5,5"/>
    <Label Grid.Column="0" Grid.Row="1">Author:</Label>
    <TextBox Grid.Column="1" Grid.Row="1" Margin="5,5,5,5"/>
    <Label Grid.Column="0" Grid.Row="2">Publisher:</Label>
    <TextBox Grid.Column="1" Grid.Row="2" Margin="5,5,5,5"/>
    <Label Grid.Column="0" Grid.Row="3">ISBN:</Label>
    <TextBox Grid.Column="1" Grid.Row="3" Margin="5,5,5,5"/>
</Grid>
```

positioning because the `DockPanel` iterates through its child elements in order, setting the position of each element depending on remaining space.

The toolbar consists of a text box that is used to enter a title, author, publisher or ISBN number to search for and a button to initiate the search. These can be placed directly into the `ToolBar`, but using `StackPanel` creates a better looking layout (Listing 23).

A `StackPanel` stacks its children horizontally or vertically. This is ideal for us as we want to group the text box and button together in the tool bar. We want them horizontally, so we set the `Orientation` attribute to `Horizontal`. The `Vertical` orientation could also be used, but that would look rather odd. To add a `TextBox` and a `Button` to the `StackPanel`, just declare them as children. Both controls have a `Margin` attribute which puts a border around the outside of each control. Each comma delimited number specifies the spacing around the top, left, bottom and right of the control in that order. The text box's `Width` attribute speaks for itself. Without it the text box would be very narrow and would grow as content was typed into it. Setting the width gives it a sensible starting width and maintains it. The button's `IsDefault` attribute is also set to `true` as we want the search button to be the default action. The text box's label is specified between the open and closing elements. This is also quite common for WPF controls. If you run the application you can enter text into the text box and click the button. The button does not do anything yet as it does not have a command associated, I'll discuss commands in the next section.

So far we've looked at the `DockPanel` and `StackPanel` layouts. These are two of the most important WPF layouts, but by far the most useful and therefore the most commonly used layout is the `Grid` layout. It has rows and columns like any other grid and allows you to to put any control in any sell or across many cells. In most cases rows and columns are defined using `RowDefinition` and `ColumnDefinition` elements (Listing 24).

Rows and columns can be defined just by placing the appropriate empty element (e.g. `<RowDefinition/>`) in the appropriate section. This would give the rows and columns a default height and width and is almost certainly not what you want.

Setting the `RowDefinition Height` attribute to `auto` will adjust the height of the row to match the controls contained in each cell. This is ideal as all the rows in the Canon UI contain a label and a text box and are therefore all the same height.

The first column contains the labels for all of a book's fields and the second column holds the text boxes for the values of the fields. The cells in the first column should all be the same width as the longest label. To achieve this we set the `Grid`'s `IsSharedSizeScope` attribute to `true` (the default is `false`) and set the first `ColumnDefinition`'s `SharedSizeGroup` attribute. The name given to it is unimportant. If we had more than one column that we wanted to be the same width, we'd specify the same name in all of those columns. Without using shared size scoping we'd have to set a specific width for the column. We want the second column to take up the remainder of the UI's width, so we set its `Width` attribute value to an asterisk to tell it to stretch out as far as it can. All that's left is to put the controls into the cells (Listing 25).

As you can see, each `Label` and `TextBox` has a `Grid.Column` and `Grid.Row` attribute that specifies its position in the `Grid`. As with a `Button`, the text for the `Label`s is specified between the opening and closing `Label` elements. Each of the text boxes also has a `Margin` set so that there is a reasonable gap between each of them.

Commands

The search text box and search button are closely related (but not coupled!). A user won't see the result of their search until they have typed something into the text box and clicked the button. The button shouldn't really be enabled unless there is content in the text box. To achieve this, we need to bind the text box to a property in the view model and bind the button to a command. I'll explain a bit more about WPF commands in a moment. To bind the search text box to a property in the view model, we first need the property:

```
public class MainWindowViewModel
{
  public string SearchText { get; set; }
  ...
}
```

and then add a bound text attribute:

```
<TextBox Margin="5,5,5,5" Width="150"
   Text="{Binding SearchText"/>
```

As with the `AppTitle` binding, `TextBox` binding is a simple case of using curly braces, the `Binding` keyword and the name of the property to bind too. The one difference is that the `SearchText` property has both a getter and setter. This means that as well as the value of `SearchText` being displayed in the search `TextBox`, any change to the search `TextBox` by the user is also written to the `SearchText` property. This is two way binding and is worked out by WPF automatically.

WPF has a version of the COMMAND PATTERN [5]. WPF In Action describes WPF's implementation of the command pattern in detail and WPF Apps With The Model-View-ViewModel Design Pattern describes an ICommand based implementation that can be bound when using MVVM3. What we're interested in is binding a button to a command so that we can perform an action when that button is pressed and telling that button whether it should be enabled or not. Listing 26 is the WPF Apps With The Model-View-ViewModel Design Pattern implementation.

Showing how it is used should provide enough explanation of it for our purposes. If you want to understand it in more detail see WPF Apps With The Model-View-ViewModel Design Pattern. Some people recommend lazy loading `RelayCommand` objects (Listing 27) but I really don't see the need. It's a lot of extra code, including a null check and the property is accessed as soon as the window is displayed and bound anyway. So I just do this what you can see in Listing 28.

The getter of the `RunSearch` property is public so that it can be bound to, but the setter is private so that it can only be set internally. The `RelayCommand` object itself is created in the view model constructor:

```
  RunSearch = new RelayCommand( o => Search(),
     o => canSearch() );
```

Listing 26

```
public class RelayCommand : ICommand
{
  private readonly Action<object> execute;
  private readonly Predicate<object> canExecute;

  public RelayCommand(Action<object> execute)
     : this(execute, null)
  {}

  public RelayCommand(Action<object> execute,
     Predicate<object> canExecute)
  {
    if (execute == null)
    {
      throw new ArgumentNullException("execute");
    }

    this.execute = execute;
    this.canExecute = canExecute;
  }

  [DebuggerStepThrough]
  public bool CanExecute(object parameter)
  {
    return canExecute == null ? true :
       canExecute(parameter);
  }

  public event EventHandler CanExecuteChanged
  {
    add {
      CommandManager.RequerySuggested += value; }
    remove {
      CommandManager.RequerySuggested -= value; }
  }

  public void Execute(object parameter)
  {
    execute(parameter);
  }
}
```

Listing 27

```
private RelayCommand _saveCommand;
public ICommand SaveCommand
{
  get
  {
    if (_saveCommand == null)
    {
      _saveCommand = new RelayCommand(...);
    }
    return _saveCommand;
  }
}
```

Listing 28

```
public class MainWindowViewModel
{
  ...
  public string SearchText { get; set; }
  public ICommand RunSearch{ get; private set; }
  public MainWindowViewModel
     (IBookRepository repo)
  {
    ...
    RunSearch = new RelayCommand(o => Search(),
       o => canSearch() );
  }

  private bool canSearch()
  {
    return !string.IsNullOrEmpty(SearchText);
  }

  private void Search()
  {

  }
  ...
}
```

Take another look at the `RelayCommand`'s two parameter constructor:

```
  public RelayCommand(Action<object> execute,
     Predicate<object> canExecute)
```

The first parameter is an `Action` delegate, which encapsulates a method that has a single parameter and does not return a value. A lambda expression is used to specify the method to call when the command is executed. As it's a delegate you could do all sorts of in-line command implementations, but I find it clearer to delegate to another method. The second parameter is a `Predicate` delegate, which represents a method that defines a set of criteria and determines whether the specified object meets those criteria. A lambda expression is used to specify a method that determines whether the command should be enabled. (The `o` parameter is ignored as it is not needed in this scenario). To determine if the command should be enabled, we look to see if `SearchText` is not null or is empty:

```
  private bool canSearch()
  {
    return !string.IsNullOrEmpty(SearchText);
  }
```

The next stage is to bind the command to the button. This is achieved by by adding a `Command` attribute to the search `Button` element:

```
  <Button Margin="5,5,5,5" IsDefault="True"
     Command="{Binding RunSearch}">Search</Button>
```

Listing 29

```
<Window x:Class="Canon.View.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="{Binding AppTitle}"
        MinHeight="200"
        Height="200"
        MinWidth="450"
        Width="450"
        FocusManager.FocusedElement="{Binding ElementName=searchBox}">
        ...
        <TextBox Margin="5,5,5,5"
                 Width="150"
                 Text="{Binding SearchText,
UpdateSourceTrigger=PropertyChanged}"
                 Name="searchBox"/>
        ...
</Window>
```

If you run the application now you will see that the search button is disabled and does not enable until you enter something into the search text box and it loses focus. This is because, as with an edit box in a browser, the event which indicates that the contents have changed is not fired until the text box loses focus. To have the event fired every time the contents of the text box have changed, we need to modify its binding:

```
  <TextBox Margin="5,5,5,5"
           Width="150"
           Text="{Binding SearchText,
            UpdateSourceTrigger=PropertyChanged}"/>
```

If you run the application again you will see that the button immediately enables or disables depending on whether the text box has content. However, when clicked the button still does nothing. In the next section we'll look at finishing the binding and getting books from the repository.

Before we move on to finish the binding there is an irritation about the UI we should fix. When the application starts, the focus is not on the search text box (Listing 29).

As you can see, the **`FocusedElement`** of the **`FocusManager`** is bound to an **`ElementName`**, which must be specified. For this to work we have to set the **`Name`** attribute of the search **`TextBox`**. When you run the application the cursor will be waiting for you in the search text box.

Searching for books

Before we can search for and display books, we need to bind the Title, Author, Publisher and ISBN text boxes (Listing 30).

The Title, Author, Publisher and ISBN text boxes use two way binding just like the search text box. If a book is found it is used to set the view model's properties:

```
private void Search()
{
  Book book = repo.Search(SearchText);
  if (book != null)
  {
    Title = book.Title;
    Author = book.Author;
    Publisher = book.Publisher;
    ISBN = book.ISBN;
  }
}
```

The above **`Search`** method uses the current value of the **`SearchText`** property to call **`Search`** on the repository. Remember that the view model's private **`Search`** method is called by the **`RunSearch`** command and the command can only be executed if the **`SearchText`** property is not null or empty. So by the time the **`Search`** method is called, **`SearchText`** is guaranteed to be valid. If a book is found a valid **`Book`** object is returned, otherwise null is returned. If a valid **`Book`** object is returned, its properties are used to set the view model's properties. However, if you run the application now you will be disappointed. Even if you enter a matching search criteria and click the search button, you will not see the Title, Author, Publisher or ISBN text boxes populated. This is because we haven't told WPF that the properties have changed. WPF will automatically

Listing 30

```
public class MainWindowViewModel
{
    private readonly IBookRepository repo;

    public string SearchText { get; set; }
    public ICommand RunSearch{ get; private set; }

    public string Title { get; set; }
    public string Author { get; set; }
    public string Publisher { get; set; }
    public string ISBN { get; set; }
    ...
}

...
<Label Grid.Column="0" Grid.Row="0">Title:</Label>
<TextBox Grid.Column="1" Grid.Row="0"
     Margin="5,5,5,5" Text="{Binding Title, UpdateSourceTrigger=PropertyChanged}"/>
<Label Grid.Column="0" Grid.Row="1">Author:</Label>
<TextBox Grid.Column="1" Grid.Row="1"
     Margin="5,5,5,5" Text="{Binding Author, UpdateSourceTrigger=PropertyChanged}}"/>
<Label Grid.Column="0" Grid.Row="2">Publisher:</Label>
<TextBox Grid.Column="1" Grid.Row="2"
     Margin="5,5,5,5" Text="{Binding Publisher, UpdateSourceTrigger=PropertyChanged}}"/>
<Label Grid.Column="0" Grid.Row="3">ISBN:</Label>
<TextBox Grid.Column="1" Grid.Row="3"
     Margin="5,5,5,5" Text="{Binding ISBN, UpdateSourceTrigger=PropertyChanged}}"/>
...
```

Listing 31

```
public abstract class PropertyChangeEventBase : INotifyPropertyChanged
{
  public event PropertyChangedEventHandler PropertyChanged;

  protected virtual void OnPropertyChanged(string propertyName)
  {
    if (PropertyChanged != null)
    {
      PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }
  }
}
```

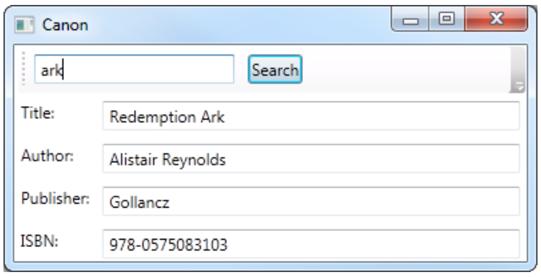
Listing 32

```
private void Search()
{
  Book book = repo.Search(SearchText);
  if (book != null)
  {
    Title = book.Title;
    Author = book.Author;
    Publisher = book.Publisher;
    ISBN = book.ISBN;

    OnPropertyChanged("Title");
    OnPropertyChanged("Author");
    OnPropertyChanged("Publisher");
    OnPropertyChanged("ISBN");
  }
}
```

Listing 31).

This implementation from WPF Apps With The Model-View-ViewModel Design Pattern is so useful that it's worth putting it in an abstract base class and then inheriting from it in the view model. This makes the **OnPropertyChanged** method available to the view model and when called fires an event with a **PropertyChangedEventArgs** object containing the name of the property that has changed. WPF picks this up and uses the appropriate binding to update the UI. You can see this if you modify the view model **Search** method as seen in Listing 32.

Now if you run the application, enter a matching search criteria and click the search button, you will see that book details are displayed!



ark

Title: Redemption Ark
Author: Alistair Reynolds
Publisher: Gollancz
ISBN: 978-0575083103

Figure 3

Finally

This is where this article leaves the Canon application. Here I have introduced you to simple WPF UI development and the Model-View-ViewModel pattern including simple binding and commands.

There is more to come in part two where I look at using images, menus, tool bars and system commands to make the Canon application more aesthetic and user friendly. ■

References

[1] *WPF In Action with Visual Studio 2008* by Arlen Feldman and Maxx Daymon. Manning. ISBN: 978-1933988221

[2] Presentation Model by Martin Fowler: http://martinfowler.com/eaaDev/PresentationModel.html

[3] WPF Apps With The Model-View-ViewModel Design Pattern by Josh Smith. MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/dd419663.aspx

[4] Canon 0.0.1 Source Code: http://paulgrenyer.net/dnld/Canon-0.0.1.zip

[5] *Design patterns : elements of reusable object-oriented software* by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Addison Wesley. ISBN: 978-0201633610

One Test or Two?

Chris O'Dell and Paul Grenyer debate the best granularity for units.

'I know what you're thinking: Is that two tests or only one? Well, to tell the truth, in all this agility, I'm not sure myself. But this is Test Driven Development, the most powerful development technique in the world, and could give you clean code, you've got to ask yourself one question: Do I feel expressive? Well, do ya, punk?'

Paul: I have a Windows Presentation Foundation (WPF) application that loads a list of widget names from a database and uses them to populate a drop down box. There is no default widget, so the user must select one from the list and click the load button to load it. The behaviour is such that before a widget name is selected the button is disabled and cannot be clicked.

Test classes should not be treated that much differently to normal classes

One pattern often employed in WPF applications is MODEL-VIEW-VIEWMODEL [1]. The view model can use variations of the COMMAND PATTERN [2] for processing commands from and giving feedback, such as the enabled state of a button, to the View. MVVM is intended to give separation between a view and its logic. Therefore the view model is usually devoid of User Interface (UI) code and easy to instantiate and run unit tests against.

I have a unit test that instantiates a view model and checks that the load command cannot be fired until a widget name is selected. A simplified version is show in Listing 1, without the view model instantiation.

Recently I have also been involved with the ACCU Mentored Developers project based around *Growing Object Orientated Software Guided By Tests* [3] by Steve Freeman and Nat Pryce. In this book they write a lot about the naming and granularity of tests. So I posted the above code to the list and asked what the project members thought: 'Should the above test be one test or two?'.

Chris: Without blinking I immediately replied that the above should definitely be split into two distinct tests so that a failure would be obvious from the test name. The above test has two asserts and as such either of these could fail.

Paul: Chris, and all the others who replied are of course right on the money (no one suggested it should only be one test), but something didn't sit quite right with me. In this small example the extra code of two tests is not very much: the method definition, a pair of curly braces and some spacing. The problem for me is that test classes should not be treated that much differently to normal classes and any class with a large number of methods becomes difficult to maintain and even to navigate, although modern IDEs do make this easier with regions (C#) and code collapsing (Eclipse).

Chris: I firmly believe that it is worth the extra code – as many others also remarked, the five minutes now and extra curly braces could easily save twice that if you need to hunt down a related bug in future.

Listing 1

```
public void
testThatExecuteIsOnlyPossibleIfNameIsSet()
{
  Assert.IsNull(model.Name);
  Assert.IsFalse(model.CanExecute());
  model.Name = "Something";
  Assert.IsTrue(model.CanExecute());
}
```

I went on to explain that in terms of application logic the two scenarios 'with a name' and 'without a name' are expected to be mutually exclusive and the two tests will ensure this by isolating each scenario and addressing them individually – with explicit test names.

In regards to the large classes this can be tackled by breaking your tests down into smaller classes, generally per scenario. For example, group all of the positive 'happy path' tests together into one class and all the negative, error handling tests into another class and store both classes in a folder given the name of the class under test.

Paul: Again, Chris is of course right. Although the implication that a test method should only have a single assert is a whole other discussion.

Then Nat Pryce came along with another suggestion altogether:

> To really play Devil's advocate, I think there should be THREE tests!
>
> 1.The model initially has no name
>
> 2.The model cannot execute when it has no name
>
> 3.The model can execute when the name has been set.
>
> You could squeeze that into two:
>
> 1.The model initially cannot execute
>
> 2.The model can execute when the name has been set
>
> But I think the state transitions and constraints are clearer with three tests.

Even with there now being the extra code for three functions, rather than one, this is of course the answer. It even satisfied the idea of one assert per method. The most difficult idea for me was not testing preconditions specific to a test if they were already tested in a another test. I have consequently modified my way of thinking and a lot of my test code. ■

References

[1] Model-View-ViewModel Pattern: http://msdn.microsoft.com/en-us/magazine/dd419663.aspx

[2] *Design patterns: Elements of reusable object-oriented software* by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Addison Wesley: ISBN: 978-0201633610

[3] *Growing Object-Oriented Software, Guided by Tests* by Steve Freeman & Nat Pryce. Addison Wesley. ISBN: 978-0321503626

PAUL GRENYER

An active ACCU member since 2000, Paul is the founder of the Mentored Developers. Having worked in industries as diverse as direct mail, mobile phones and finance, Paul now works for a small company in Norwich writing Java. He can be contacted at paul.grenyer@gmail.com

CHRIS O'DELL

Chris is a C# Web Developer working in London at 7digital who is constantly learning. When not programming she can be found with her head in a fantasy book or a manga comic.

Perforce Cross-Platform Patcher

Alexander Demin demonstrates a deployment tool for Perforce.

Perforce is a great version control system (VCS). In the world of centralised systems Perforce does great job and almost all routine tasks in Perforce are supported easily: checking in and out, diff'ing, merging and branching.

I want to focus on one particular aspect of Perforce. It becomes an issue when you have to deal with multiple development machines. For example, in my current company we release our product on many different platforms: Linux, Solaris, AIX, HP-UX, Windows and zOS. It means that every change set must be verified on all platforms before a submit.

Usually the workflow looks like this: you checkout files on main development machine, for example, Linux, and do your stuff. When it is ready to submit, you have to also try your changes on other machines. You analyse what you've done via 'p4 diff ...', somehow pack the changes and move it across to other boxes. Then you compile and test on those machines and likely if everything is fine, submit everything from your original machine. But often in development, things go wrong and most likely your changes will fail on other platforms (in compilation, in tests or somewhere else). In this case you have to fix it locally, on every machine where it fails, and then get all the fixes together and apply them (somehow, probably manually) on the original machine. Likely, this cycle will be repeated many times.

When merging branches you may end up in a situation involving hundreds of source files. You have to do a lot of manual work: copy files across, checkout, replace existing files and finally revert it back. What about if you have to process dozens of files or more and repeat 10 times in a row?

When merging branches you may end up in a situation involving hundreds of source files

Of course, any more or less experienced UNIX developer can write a script to diff, pack and move files automatically, but ideally when a bunch of files are copied from the original machine it is better to have them properly checked out, rather than simply overwritten. If the files are not checked out with the 'p4 edit ...' command it is difficult to track and revert changes.

Okay, I hope I've scared you enough and you are ready to listen a solution I propose.

First, Perforce, starting from the version 2010.2 has a very neat feature – shelving. It solves the problem completely. Shelving allows you to upload a change set to the Perforce server without submitting, but still making it available for download from other workspaces, so you can send change sets from one machine to another. But Perforce is a very robust and stable system, and sometimes people don't upgrade because it is a critical part of their process, and upgrades are an upheaval. So, shelving is not available for them.

A utility I'd like to talk about does roughly the same as shelving – it automates the process of managing change sets across different machines. It is called 'p4patch' or 'p4p' and is hosted on Google Code [1].

It is implemented as a single source file written in ANSI C. It doesn't require any external utilities or interpreters, only a C compiler.

ALEXANDER DEMIN

Alexander Demin is a software engineer with a PhD in Computer Science. Constantly exploring new technologies he is always ready to drill down into the code with a disassembler to prove that the bug is out there. He may be contacted at alexander@demin.ws.

You just need to download its source [2] and one of the scripts to build it [3], depending on your platform.

On Linux and AIX:

```
cc -o p4p p4p.c
```

On Solaris:

```
cc -o p4p -lsocket -lnsl p4p.c
```

On HP-UX:

```
cc -D_XOPEN_SOURCE_EXTENDED -o p4p p4p.c
```

On Windows:

```
cl p4p.c wsock32.lib ws2_32.lib
```

Once you've built it you are ready to try it out. Running **p4p** without any parameters will print out all available command line options.

First of all, on every machine where you are going to send change sets from the main machine, you need to start a server: **p4p server**.

The default port is 20050 but you can specify different one via the **-p** flag.

Now let's assume on your main development machine you have a change set and **p4 diff** shows what is in there. Now by executing **p4p diff** you will create a TAR archive (`patch.tar`) containing your change set and a list of files.

By default, **p4p diff** includes all checked out files but you can customize it using the **-o** option which defines a list of files included in `patch.tar`. Such a list can be created with **p4 opened | grep ...** where **grep** can filter for the required files.

Okay, we are ready to transmit the changes to a remote machine (for this example, its address is 10.44.5.9) via **p4p patch -h 10.44.5.0**. The command **p4p** transmits `patch.tar` to the remote host 10.44.5.9, unpacks it there, checks out every individual file from the list using **p4 edit** and overwrites it with the corresponding file from `patch.tar`. Afterwards you go to the target machine and execute **p4 opened**, and you'll see all the files you've just moved.

You can also do it from your main machine using the

```
p4p exec -h 10.44.5.9 -p4 opened
```

command. Similarly

```
p4p exec -h 10.44.5.9 -p4 opened
```

prints out all checked out files on 10.44.5.9.

Now you can compile, run and test the change set on 10.44.5.9.

You've done your testing and, for example, found some issues. To revert all the changes on 10.44.5.9 you simply execute: **p4p revert -h 10.44.5.9** on the main machine. This command carefully reverts the changes on 10.44.5.9 via applying **p4 revert** to every single file.

While the p4p server is running on a remote machine you can fully control the remote Perforce client: apply or revert your patch, or execute arbitrary Perforce commands. For example, **p4p exec -h 10.44.5.9 -p4 -v** prints out the Perforce version on that remote host.

When **p4p** applies the file change on the remote machine it always tries to preserve the current line ending used on the remote platform. It means you can safely propagate files from a Windows to a UNIX box and vice versa.

Bletchley Park Climbs to New Heights

Astrid Byro climbs to Everest Base Camp to raise funds for Bletchley Park.

What could motivate a self-confessed 'over-the-hill, overweight, out-of-shape, 30-a-day smoker' to attempt an assault on Everest Base Camp? Independent Bletchley Park supporter Astrid Byro thinks raising money for Bletchley Park is the answer to that question. Astrid, who has previously helped Bletchley Park by organising annual fundraising conferences, is now embarking on a much more personal campaign. On 16 August she will attempt a gruelling 8-day trek to Everest Base Camp.

CEO of the Bletchley Park Trust Simon Greenish said, 'This incredible challenge Astrid is embarking upon in aid of Bletchley Park is absolutely wonderful. Over the last few years we have had a great level of public support by so many committed people who have so generously given their time and energy in order to help us develop however I think this particular exploit is possibly the most unusual! We are enormously grateful to Astrid for her dedication to help Bletchley Park and I very much look forward to seeing her progress.'

Mount Everest is the highest mountain in the world at 8,848 metres and the base camp is an amazing 5,545 metres above sea level, which is more than four times the height of the UK's tallest mountain Ben Nevis. As Astrid says, 'You must understand the context of this endeavour. I'm afraid of heights and this will challenge my fears on a daily basis with multiple crossings of rickety bridges across torrential gorges. In addition, I will be doing this at the end of monsoon season so there is the ever-present danger of flash floods as well as the menace of leeches. I hate leeches.'

Astrid has not only set a tough climbing challenge but has also set her fundraising target at £50,000. She is hoping to achieve this target by donations as well as corporate sponsorship so if you would like a photo of your corporate logo flag flying at Base Camp, want her to wear sponsored logo clothing, or you have a stunt in mind, she's open to negotiation. Astrid adds, 'If Jimmy Choo wants a picture of me wearing some strappy heels at Base Camp, I'm game. But I get to keep the heels!'

You can follow Astrid's progress on her blog as she pursues her training programme, at www.abc-ebc.blogspot.com and you can support her by making a donation at www.justgiving.com/Astrid-Byro

Corporate sponsorship opportunities are available. Please contact: Kelsey Griffin, Director of Museum Operations, Bletchley Park Trust, kgriffin@bletchleypark.org.uk or phone 01908 272655

Bletchley Park mansion

Interior of a hut before restoration work

Restored hut, preserved for the future

Perforce Cross-Platform Patcher (continued)

To recap, this utility helps you transfer active change sets in Perforce from one machine to others, apply and revert them. If your Perforce is more recent than version 2010.2, I recommend using the standard Perforce shelving mechanism.

This utility saves me many hours of dull work when doing massive merges with hundreds of files and testing them on many different machines. I hope it will also help you. ■

References

[1] http://code.google.com/p/p4patch

[2] http://p4patch.googlecode.com/hg/p4p.c

[3] http://code.google.com/p/p4patch/source/browse/

A Software Experience

Simon Sebright shares his frustrations with popular tools.

I write this not as a software development article, but as an observation on the usage of software. I wish product managers and software developers alike would take more care and pride in what they do. Above all, listen to and watch real users.

I am a software developer and as such, like many I expect, I am the IT support department of our household 24x7 (apart from when I don't really feel like it, or when the customers are asleep). My chief customer is my wife, who is comfortable using her day-to-day applications, but when it comes to doing anything new or tricky, often needs my help.

And so it came to pass that she acquired a smart phone made by a company we shall call Banana. The model we shall call fPhone to make sure I am not pursued in law by the real makers of the said device (and to reflect the level of frustration it can sometimes bring).

I needed to form a mental model of how Banana had written this software . . . simply to use the darn thing

The task was simple enough – load a couple of Apps onto the device from the AppShop (as we shall call). Great, first fire up fMelodies, the piece of software you need to do anything with your fPhone. That was painless. Now what to do? From here on in, I can only say that as a novice user it was a minefield. Here are some of the points I remember:

- Connecting the fPhone with USB didn't do anything at first. It appeared to have too little charge, although didn't tell us that. OK, charge it up then we'll try again later, dear...
- Once charged, connecting the fPhone worked and it appeared as a device under fMelodies. Great, except that it looked disabled, and was pretty unresponsive to mouse clicks. Using a touchpad, one is inclined to 'click' a little more often than with a conventional mouse, and so it was that she managed to click something which exposed a menu and the menu item to remove the device within half a second. It was gone! Only the usual unplug, plug it back in again approach worked.
- Hmm, I am now starting to lose my patience with the Banana company, feeling that they have slipped up (pun intended of course!) OK, let's go to the AppShop. I tell her to click on the appropriate menu. Nothing happens for several seconds. 'There,' I say. 'I don't see it,' she says. Aha, there are not one, but two menus! One for the main application and one in the 'Device' area (which we managed to select, but still looks disabled). Which one are we supposed to use? It wasn't clear to me at all.
- Having finally got to the AppShop, we realise that there are at least two search boxes to be seen, and use trial and error to get the right one.
- We had to set up an account for her, which actually worked quite well, although they did want rather a lot of data from us, considering we were only going to download a couple of free Apps.
- Once we had found what we wanted, the Apps got downloaded to the fMelodies application and we had to somehow get them on the phone. We dragged and dropped, synced, did it again, and again. What should have been intuitive was not working very well. This was compounded by the fact that in the middle of a sync, she cancelled it. 'Aha, that's your own fault,' you say. Not so, the phone looks in that state exactly as it does when in standby. The text is different of course ('Slide to cancel' instead of 'Slide to unlock', but as it is green, you don't read that, and there was no confirmation.
- To further compound matters, I had previously loaded Apps on there with my own account. It very nicely asked us if we wanted to transfer them to fMelodies. Great, except that didn't seem to work first time either (that may have been where the sync cancel came in, if I remember correctly). I think I had to unplug it yet again.

We finally got there, but it was a slog. Remembering James Bach's sessions at the ACCU 2010 conference, I realized that I was having to think like a tester. I needed to form a mental model of how Banana had written this software. In my case not to test, of course, but simply to use the darn thing. For a novice user it was really a nightmare, and for an advanced user, annoying.

And then there is the the other software company whom I shall call Bighard (hee hee ;^). I created a presentation using their StrengthSpike software at work, saved it and put the laptop into hibernation. On the train, I restart the laptop to refresh my memory. I notice a mistake and want to edit the thing, except I can't, because it's in protected view. OK, I click on Enable Editing (knowing I can trust the source). It then gives me a message box saying it can't find the file, and won't enable editing for me. This is true, because I am no longer on the network, but I don't see why that should stop me. OK, so I now start to think laterally; I know, I'll save it as another file for the time being. Saving is disabled in protected mode! And guess what? Yes, the same silly problem stops me again. The only thing I can think of is to use the clipboard to copy the whole content to a new file, which messes up my formatting somehow, as I don't have the right template. Aaarrrggghhh.

We dragged and dropped, synced, did it again, and again. What should have been intuitive was not working very well

This user story is ridiculous. A product with so many person years behind it and designed for everyday use simply should be better. We know from good books on software development that this kind of 'error' is to be expected and should be catered for in program logic. Some programmer somewhere made the decision to simply output a message box if something went wrong. I hope he or she reads this and has a very bad conscience for a time or two, and more importantly can learn from it. And if you are the product manager responsible for this, then it is part of your job to work out what to do! Yes, it's nitty gritty detail, but it is important to get right.

I am sure there are countless stories like this every day, where millions of person-hours are squandered. Companies are blissfully unaware that their users are frustrated, trapped. All you product managers and software developers out there, please try to raise your standards. Listen to and watch your users. They are not stupid, they just don't deal with software for a living! ■

SIMON SEBRIGHT

Simon Sebright has been developing software or managing it for over 15 years, suffering from it even longer. Currently he is battling with the beast they call SharePoint.

Inspirational (P)articles

Frances Love introduces Katie Friesen from DevChix.

Katie Friesen from DevChix tells us how she got into programming for another inspiration particle. DevChix is 'an international group of female programmers working to make the tech community a better place for everyone.' (http://www.devchix.com/)

In 1996 my family got a modem and hooked up to the hot new World Wide Web. One issue of American Girl Magazine, which I subscribed to along with most of my friends, contained an article on making web pages. I was enthralled. I set up a homepage and started learning HTML.

When I was selected for a local science and technology magnet high school I enrolled in the accelerated programming class offered to freshmen and got a solid dose of C++. I will never forget the sense of accomplishment I felt looking at the terminal producing the output of my first program.

Because of the technology sector bubble of the late 1990s I was able to find an internship working for a local company that allowed me to do bug fixes on their applications. During the school year I took every programming class my high school offered, getting brief overviews of computer graphics, artificial intelligence, and computer architecture. I did a year-long senior project in PHP.

At the small liberal arts college that I chose for my undergraduate work I was one of three students in my class to major in computer science and for two years the only woman in the entire major. I was very glad for the training I received and the encouragement from peers and professors as I launched into my career as a software engineer.

Write for us!

C Vu and Overload rely on article contributions from members. That's you! Without articles there are no magazines. We need articles at all levels of software development experience; you don't have to write about rocket science or brain surgery.

What do you have to contribute?

- What are you doing right now?
- What technology are you using?
- What did you just explain to someone?
- What techniques and idioms are you using?

For further information, contact the editors: cvu@accu.org or overload@accu.org

Goodbye from the Conference Chair

Giovanni Asproni takes his last bow.

The 2011 event was my last one in the conference committee, after six years – with the last four spent has the Conference Chair – it was time for me to step down. In hindsight I couldn't do it at a better time – in 2011 we, for the first time, sold out a couple of weeks before the event, allowing me to leave on a high :-)

During the years the conference has changed a lot – it has a bigger attendance, a wider subject coverage and a greater international visibility. In the last four years we also experienced an huge increase in the number of proposals which caused the acceptance rate to drop below 50% and forcing me and the committee to make some very hard choices. I attribute this increase to the wider subject coverage and the greater visibility.

There is one thing that hasn't changed at all. We managed to keep the quality of the conference intact. In fact, the reputation is so high and the atmosphere so unique that all the 'big names' that have been there once are always willing to come back – e.g., Robert Martin, who was supposed to be one of the keynote speakers in 2011, but had to pull out because of personal reasons, wrote in his blog 'This is one [is] of the few conferences I go to just for the fun of it, so I'm kinda bummed' (http://cleancoder.posterous.com/the-last-programming-language).

Alas, I cannot take all the credit for all those achievements. The foundations for them have been built during the years by the previous committees and conference chairs. During my tenure it has always been a team effort. We are indebted to all the people who have served on the conference committee, Julie Archer and her colleagues at Archer Yates (who took care of all the logistic aspects), my network of unofficial advisers (in particular Kevlin Henney and Allan Kelly, who were always available to provide help and suggestions) and, last but not least, the speakers and the attendees who always made every one a memorable conference.

The Conference Chair has now been taken by Jon Jagger. I'm sure he will be able to make the conference even better and I'm looking forward to ACCU 2012 next April – I will be more relaxed and more willing to stay late at the bar...

Code Critique Competition 70

Set and collated by Roger Orr. A book prize is awarded for the best entry.

Please note that participation in this competition is open to all members, whether novice or expert. Readers are also encouraged to comment on published entries, and to supply their own possible code samples for the competition (in any common programming language) to scc@accu.org.

Last issue's code

I've written a streaming helper for dates – seems to work OK – and a manipulator so I can stream today's date, plus a convert from string function. Please can you review my code?

Listing 1 is the code, and Listing 2 is an example of its use.

Critiques

There weren't any critiques this time … I can only deduce that ACCU members are not interested in critiquing badly written date routines.

Commentary

Let's look at this code under three separate headings.

Usability

The header file is described as 'a streaming helper for dates' and while it does provide a function for streaming out **`time_t`** values it does not (directly) help with streaming them in: the function convert creates a **`time_t`** from a **`char const *`**. While this may well be a useful function, the header probably ought to provide an input operator such as:

```
std::istream& operator>>(std::istream& os,
  date & input);
```

This would go along with a constructor taking a non-const reference to the **`time_t`** that would be populated by the input – an example of the usage pattern would be:

```
time_t tv;
std::cin >> date(tv);
```

I'm also not persuaded that the class is correctly named: **`date`** leads me (at any rate) to expect a class that has general date functionality rather than simply an input/output manipulator. Perhaps **`date_manip`** or **`date_io`** would be a clearer name?

Implementation

The actual details of the date conversion functions are a little complicated and I would want to do some checking to verify they are in fact correct. In my opinion the days of writing your own date manipulation functions are long past (except in some specialised cases) – it is a classic example of something that you should look for in a standard library. The C runtime includes various functions such as **`mktime`** and **`strftime`** and the C++

Listing 1

```
#include <iostream>
#include <time.h>
#pragma once
// date class
class date
{
  time_t const& tv;

public:
  date(time_t const& tv) : tv(tv) {}
   // print self to stream as YYYY-MM-DD
  void printOn(std::ostream& os) const;
   // convert YYYY-MM-DD to time_t
  static time_t convert(char const *);
};

void date::printOn(std::ostream& os) const
{
  int day = tv / 86400;
   // base on Mar 1968 (makes leap years easy)
  day += 365 + 366 - 60;
  int year = day / 365;
  day -= year * 365;
  day -= year/4;

  if (day < 0)
  {
    day += 365;
    year--;
  }

  int month = 0;
  int daycounts[]
    = {31,30,31,30,31,31,30,31,30,31,31,28};

  while (day >= daycounts[month])
  {
    day -= daycounts[month++];
  }

   // base on Jan
  month += 2;
  year += month/12;
  month %= 12;

  os << year+1968 << "-";
  os << (month<9?"0":"") << month+1 << "-";
  os << (day<9?"0":"") << day+1;
}
```

ROGER ORR
Roger has been programming for over 20 years, most recently in C++ and Java for various investment banks in Canary Wharf and the City. He joined ACCU in 1999 and the BSI C++ panel in 2002. He may be contacted at rogero@howzatt.demon.co.uk

Listing 1 (cont'd)

```
time_t date::convert(char const *date)
{
  int yr,mn,dy;
  if (sscanf(date,"%i-%i-%i",
      &yr, &mn, &dy) < 3)
    return -1;

    // base on Mar 1968
  yr -= 1968;
  mn -= 3;
  if (mn<0)
  {
    yr--;
    mn += 12;
  }

  dy--; // 1-based
  int daycounts[]
    = {31,30,31,30,31,31,30,31,30,31,31,28};

  while (mn)
    dy += daycounts[mn--];
  dy += yr * 365;
  dy += yr / 4;
    // rebase to 1970
  dy -= 365 + 366 - 60;
  return 86400 * dy;
}

// stream date
std::ostream& operator<<(std::ostream& os,
  date const& rhs)
{
  rhs.printOn(os);
  return os;
}

// stream today's date
std::ostream& strdate(std::ostream& os)
{
  return os << date(time(0));
}
```

runtime contains **time_get** and **time_put**. Using either set is likely to be more robust than writing your own date handling routines: this is an area that seems to have a large number of pitfalls for the unwary. For an example of **time_get**, for instance, see cplusplus.com [1].

The code is not very flexible, especially for input where different users and different places have a range of conventions for date formats. Again, this is normally much easier to support when you use standard functions rather than when you are trying to roll your own solutions.

Additionally the code is assuming that the UTC date is what is required: I suspect that many users might prefer to handle dates in local time or at least have that as an option.

Listing 2

```
#include "strdate.h"
int main()
{
  std::cout << date(86400) << std::endl;
  std::cout << date(86400*31) << std::endl;
  std::cout << strdate << std::endl;
  std::cout
    << date(date::convert("2000-01-01"))
    << std::endl;
}
```

Other problems

As it stands the header file can't be included in more than one compilation unit since it includes implementation of methods without marking them as inline. An alternative solution is to move away from a header-only solution but providing an implementation file containing the method definitions. While this can be slightly less convenient it can have significant advantages in build time as the header is much more lightweight, less code goes into each object file and the linker has less work to do.

The header includes **<iostream>** but does not need to: **<ostream>** would be enough. Including unnecessary headers is to be avoided as a rule (and readers of accu-general may recall Peter Sommerlad announcing on 1st June a beta version of a tool 'Includator' to help with automating this). Additionally **<iostream>** brings a static initialiser into every file that includes it which can cause various performance problems when executing the program – see, for example, http://llvm.org/docs/CodingStandards.html#ll_iostream where including **<iostream>** in library code is actually forbidden!

References

[1] http://www.cplusplus.com/reference/std/locale/time_get/get_date

The winner of CC 68

It could have been you … why not write your critique for this issue now?

Code Critique 70

(Submissions to scc@accu.org by Aug1st)

I've written a simple arithmetic expression evaluator – it works left to right but does supports brackets, a bit like old fashioned calculators. But when I try to test it I get this output:

```
cc70>test "1+2" "1/3 * 3"
"1+2" = 1.78744e-307
"1/3 * 3" = 1.78744e-307
```

Can you explain what's wrong (and also comment generally on the implementation)?

- The test program is in Listing 3
- The header (expr.h) is in Listing 4
- The implementation (expr.cpp) is in Listing 5.

You can also get the current problem from the accu-general mail list (next entry is posted around the last issue's deadline) or from the ACCU website (http://www.accu.org/journals/). This particularly helps overseas members who typically get the magazine much later than members in the UK and Europe.

Listing 3 (cont'd)

```
#include "expr.h"

#include <iostream>
#include <sstream>

// evaluate and print supplied expression
int test(std::string const &s)
{
  try
  {
    expr e(std::istringstream(s).ignore(0));
    std::cout << "\"" << s << "\" = "
      << e.value() << std::endl;
    return 0;
  }
  catch (std::exception & ex)
  {
    std::cerr << "\"" << s << "\" failed: "
      << ex.what() << std::endl;
    return 1;
  }
}int main(int argc, char **argv)
{
  int ret(0);
  for (int idx = 1; idx != argc; ++idx)
  {
    ret += test(argv[idx]);
  }
  return ret;
}
```

Listing 4

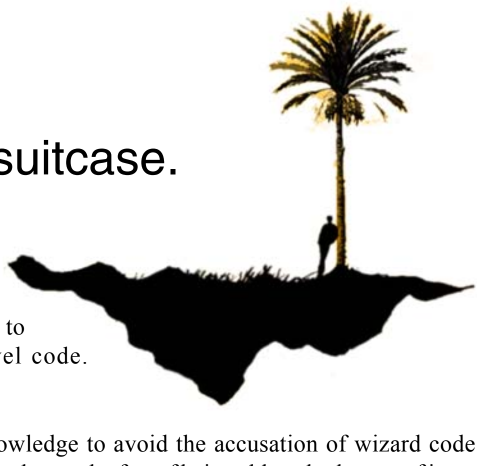
```
#include <istream>

/* Left to right expression parser */
class expr
{

public:
  /* Parse a stream */
  expr(std::istream &is);
  /* Get the value */
  double const & value() const { return val; }

private:
  /* A term in the expression */
  class term
  {
  public:
    term(std::istream &is);
    term(double v);
    operator double() const { return val; }
    void operator+=(term const& rhs);
    void operator-=(term const& rhs);
    void operator*=(term const& rhs);
    void operator/=(term const& rhs);
    void operator%=(term const& rhs);
  private:
    double val;
  };
  term val;
};
```

Listing 5

```
#include "expr.h"

#include <functional>
#include <iostream>
#include <stdexcept>
#include <string>

expr::expr(std::istream & is) : val(is)
{
  char op;
  while (is >> op && op != ')')
  {
    if (op == '+') val += is;
    if (op == '-') val -= is;
    if (op == '*') val *= is;
    if (op == '/') val /= is;
    if (op == '%') val %= is;
  }
}

// Read a number or a bracketed sub-expression
expr::term::term(std::istream & is)
{
  char op;
  is >> op;
  if (op == '(')
    val = expr(is).val;
  else if (!(is.unget() >> val))
  {
    std::string error("Bad parse at: ");
    throw std::runtime_error(error + op);
  }
}

// ctor from double
expr::term::term(double v) : val(v) {}

void expr::term::operator+=(term const& rhs)
{
  val += rhs;
}

void expr::term::operator-=(term const& rhs)
{
  val -= rhs;
}

void expr::term::operator*=(term const& rhs)
{
  val *= rhs;
}

void expr::term::operator/=(term const& rhs)
{
  val /= rhs;
}

void expr::term::operator%=(term const& rhs)
{
  val = std::modulus<long>()(val, rhs);
}
```

Desert Island Books

James Byatt shares the contents of his suitcase.

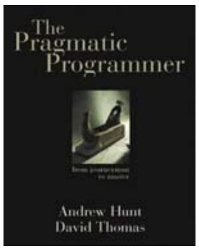


The imagination and effort that people are kind enough to put into this column never ceases to amaze me, along with how keen people are to write for it. The series has grown further than than I could have imagined and it has certainly helped me to get to know what makes certain ACCUers tick. My Subversion repository tells me that I wrote the first column in January 2008, since then 21 people have described their Desert Island Books and of all of the people I've asked in that time only 2 have declined. From Kevlin Henney onwards (Kevlin was the first after me) the imagination has grown and made me see just how flat my original was. Maybe I'll have to have another go.

I've only really started to get to know James Byatt properly in the last few months. The only embarrassing story I have involving him was when he interviewed me, over the phone, and I didn't even know it was him! Luckily I did ok. On another occasion, over a very nice Mexican meal one lunchtime, we managed to completely bore Alan Stokes by talking about music and one Swedish Progressive Death Metal band in particular. I am exceptionally happy to present the most in-depth Desert Island Books yet.

James Byatt

I fear that by the time I have packed this chest for my desert island sojourn much will have been revealed about myself that somewhat hurts my credentials as a software engineer! I don't read a great deal of technical books; I heavily distil the few I finish (apologies to the unopened copy of *Pro Spring* that I won in a raffle); one who prefers generality to the point of endangering important specifics (what's this 'standard' thing I keep hearing mention of?); one who defers the creation of real content with mindless semi-colon delimited lists.



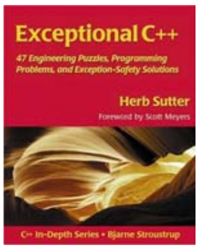
My first choice is also the first technical book I read. It aided my transition into work from idle, drunken studenthood no end. It is *The Pragmatic Programmer*. Until that point, most of the programming I'd done had been php-based web applications and, frighteningly, quite a lot of VB (and VBA!).

The book immediately turned on several lights: the need for source control and continuous integration, the near mindless drive for automation needed to keep a project moving as it grows, the sheer volume of nitpickery required to keep a large software project in order.

One concept I still have trouble with is 'wizard code'; drawing the line of where my knowledge can safely end is very difficult. In different projects I've discovered issues rooted in everything from application level to operating system level code. There seems almost no way of holding together all of that knowledge to avoid the accusation of wizard code usage; I've withdrawn to the nearby fort of being able to look most of it up.

Sadly, much of the book's message had yet to really sway the majority of the team I was in, and a swathe of manual process evolved in order to bridge the gaps. Making changes was frequently programming by coincidence almost as much as I had done on my own. In hindsight, I can see we were extremely lucky to have an extremely dedicated (and patient) QA team holding things together.

I wilfully shoved the book in the general direction of a couple of other graduate joiners in the hope that they might also glean some of the knowledge from it and make my life easier. One of them showed dangerous signs of having digested it utterly, finding several places where I had done evil (™) things. I counter-claimed that it was simply wizard code.



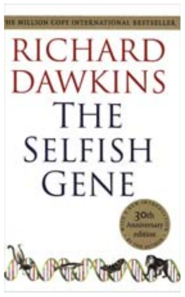
My second choice is Herb Sutter's *Exceptional C++*. A copy was pushed at me as I began to contribute to a large C++ codebase, probably because I wasn't showing a suitable amount of fear given the scale of the task (and the amount of legacy code) this project would be facing.

Here the lessons were sharper. Three, in particular, stick out. The first (and probably best) is the example that asks: 'how many execution paths are there through this trivial looking piece of code?' Answer: a frightening number. I'd been exposed to 'modern' C++, but up until reading that, the motivation behind it had remained unclear to me. The scope of my ignorance was (partially!) revealed.

The second was the LSP – a tremendously general tool for reasoning about object oriented programming. I have used it to rail against the misuse of concrete inheritance ever since. I already disliked it, but it was extremely satisfying to find that there was theory to back up my opinion. You can't beat research that confirms your own personal prejudices.

The third was the pImpl idiom, and, more generally, the power of terse header files. A whole new world of encapsulation suddenly appeared; I became Gollum-like, keeping all but the slightest hint of implementation hidden away, safe from the prying eyes of hobbits.

A second look at the legacy libraries we'd be relying on now filled me with dread; exactly as the lender of the book had intended, I suspect.



My third book steers me safely out of the frying pan of software, and into the fire of ethology. It is *The Selfish Gene*, Richard Dawkins' introduction to modern evolutionary theory. Regardless of what you may think of Dawkins' more recent publications working towards the ultimate description of all religion as bunk, this, his first book, remains a powerful description of the process of evolution.

If you do happen to sit on the 'Dawkins is a prannet' side of the fence, I still highly recommend you read it, as in the preface he

What's it all about?

Desert Island Disks is one of Radio 4's most popular and enduring programmes. The format is simple: each week a guest is invited to choose the eight records they would take with them to a desert island (http://www.bbc.co.uk/radio4/factual/desertislanddiscs.shtml).

The format of 'Desert Island Books' is *slightly* different from the Radio 4 show. You choose about five books, one of which must be a novel, and up to two albums. Some people even throw in the odd film. Quite a few ACCUers have chosen their Desert Island Books to date and there are plenty more to go.

The rules aren't too strict but the programming books must have made a big impact on your programming life or be ones that you would take to a desert island. The inclusion of a novel and a couple of albums helps us to learn a little more about you. The ACCU has some amazing personalities and Desert Island Books has proved we only scratch the surface most of the time.

Each issue of CVu will have someone different. If you would like to share your Desert Island Books please email me: paul.grenyer@gmail.com.

Next issue: Rachel Davies.

reveals a frustration that people still regard this as his best book, where he prefers *The Extended Phenotype*. Perhaps you may wish to cause him some small unknown annoyance by reading this book and preferring it, as I do.

For the phenomenally lazy, reading the first two or three chapters may be enough. Here, Dawkins constructs the idea of 'replicators': simple molecules that, in the right conditions, can create copies of themselves from natural resources in their environment. In such a scenario, which replicators become most prevalent? Those that can copy themselves most accurately ('fidelity'), and at the highest rate ('fecundity').

It is clear (to me at least) that this process is at the core of evolution. I might go as far as to say this *is* evolution, but that's a dangerous statement to make! Given this powerful analogy though, we naturally now ask: what is replicating?

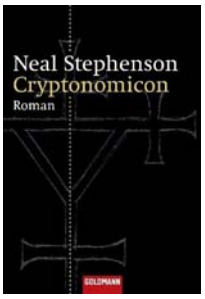
The answer is not the individual, nor the species; it is the gene. There's an obvious aside here: what is a gene? Some complain the definition given in TSG is infuriatingly circular where others laud it as elegant. You're going to have to think (© Richard Harris), I'm afraid. What does this mean for us as humans? Do we have free will, or are we merely 'lumbering robots', controlled by our genetic code? If the latter, can we explain the existence of altruism and co-operation (or are they themselves illusions)? The book starts down the path to answering some of these questions, and provides excellent examples of reasoning from a gene centric point of view.

The preface mentions that many readers find the level of reductionism advocated by the text frightening; I find the reverse! It is no small comfort to have an enormous mystery (viz: biodiversity, human behaviour) reduced to simple, scientific explanations. Why are we here? Book four puts it better than I ever could...

> ...Godfrey was, by birthright, a stupendous badass, albeit in the somewhat narrow technical sense that he could trace his ancestry back up a long line of slightly less highly evolved stupendous badasses to that first self-replicating gizmo – which, given the number and variety of its descendants, might justifiably be described as the most stupendous badass of all time. Everyone and everything that wasn't a stupendous badass was dead.

(Full quote here: http://samuelhansen.com/post/3049394857)

This is Neal Stephenson's *Cryptonomicon*. Like #3, I've gone through several copies of it – I think I've bought it at least four times, the other three having been shoved at people foolish enough to: a) ask me about books, b) be near a copy of *Cryptonomicon*, c) have not read *Cryptonomicon*. I was delighted to find my current copy in Oxfam for £1!



The book is split across two time periods:

The first, set during the second world war, tracks the adventures of two main protagonists: Godfrey Waterhouse (a mathematician) and Bobby Shaftoe, a U.S marine, both of whom end up involved in Detachment 2702, an Allied group breaking Axis crypto systems while simultaneously creating evidence to persuade the enemy that their codes are intact.

The second, set in the modern epoch, tracks the progress of a team of entrepreneurs setting up a data haven in the fictional south east Asian sultanate of Kinakuta (perhaps I should attempt the same on this desert island!). Our main protagonist here is Randall Waterhouse (grandson of aforementioned Godfrey), programmer extraordinaire.

Cryptonomicon is, for me, a big geeky comfort blanket. The two 'Waterhouse' protagonists roughly mirror my two halves of mathematician and hacker, although I wouldn't claim anything like their heroic levels in either field, and my claim of being the former grows weaker by the day. The worlds in which they walk, the ways in which they talk, even the style of the writing; these all resonate enormously with who I am. I often end up reading *Cryptonomicon* when I'm finding life a bit of a challenge as a result.

There ends of the papery part of the media chest – I struggle to pick a fifth that has had the same magnitude of effect on me. There are plenty that came close. *Effective Java* would probably have made it if I hadn't read #2 beforehand. Ben Goldacre's *Bad Science* has to be near the top of the pile as well. In fiction I was immensely grateful for the existence of *Cryptonomicon*. I would otherwise have faced a torturous choice between one of the other Stephenson books, probably *Quicksilver*, or maybe *The Diamond Age*, a Pratchett, probably *Night Watch*, or *Thief of Time*, or Charles Stross's *Accelerando*. So I will instead cheat, and pick the pair of talks given by Simon Peyton-Jones at ACCU '08, titled 'Caging the effects monster' and 'A taste of Haskell'.

I'd heard good things about Haskell but knew very little about it, similarly I hadn't really grokked the real ideas underpinning Functional Programming. I was definitely on the immutable data boat, and apparently everything in Haskell was immutable: 'what an awesome concept', I thought, without really considering the consequences.

It turned out, frighteningly, to be even more awesome than that. The thing that stays with me is the sheer amount of information contained in each function signature, because in the absence of IO, the scope of what the function can access is right there, in the types referenced in that signature! Sometime later someone showed me the Curry-Howard isomorphism, at which point my brain exploded.

At the time of the talks I was seriously pining for purity (having just finished a degree where I had avoided anything including the word 'applied' or 'modelling' for nearly three years), so I was especially vulnerable to such ideas; especially when delivered by the charisma machine that is SPJ, and even when, as was revealed at the end, xMonad's entire-window-manager-in-sixty-lines-of-uber-Haskell needed a boatload of C code in order for the Haskell parts to interface with X.

The media chest is nearly full! Space remains only for one album of music. For me, this is the hardest choice; I own at least as many CDs as I do books; just in order to write this section I've had to isolate myself from any possibility of listening to the shortlist I drew up, because, frankly, they are all too distracting.

After much rumination, I'm going to impulsively pick Opeth's *Blackwater Park*; my gateway drug into progressive music. Frighteningly the full genre of this album, and Opeth in general, is probably 'progressive death metal'. However, I wince whenever I type that, because it's so utterly ridiculous. [PG: I think it's an excellent and descriptive genre title. Wait until you get into Progressive Black metal!] This album has probably had more effect on what I listen to now than any other.

It was considerably more technically complex than anything I'd come across before, the sheer number and quality of musical ideas per song was mind-blowing, and the variety of places they'd stolen from was again, huge.

Years later, I struggle to listen to 'standard' rock and metal, and popular music in general, which sticks to a fairly rigid structure of verse, bridge and chorus, with, if you're lucky, an instrumental part stuck in where there was space, it simply doesn't do enough to hold my interest. This is *Blackwater Park*'s fault. There are obvious exceptions here, many bands to never depart from that structure have recorded some excellent music, like, um...none of the examples that I thought of, who I only considered listing because they are less inventive than my average liked album.

Now, *Blackwater Park* is neither the most progressive nor the most technical music I now listen to, and it probably isn't the most expressive either. This, too, is *Blackwater Park*'s fault. Even having found albums that push the boundaries more, I still enjoy *Blackwater Park* because it brings together all three of those dimensions so successfully, and it's pleasing that it's standing the test of time. On the island, however, I might have to wait for a thunderstorm in order to play it in the right atmosphere.

Mentored Developers Update

Paul Grenyer outlines the latest projects.

It's an exciting time for the ACCU Mentored Developers! The Growing Object Orientated Software Guided By Tests (GOOS) project has completed and we've planned and started the next two projects.

Well, that just about wraps it up for GOOS

Sunday 30th May 2011 was officially the last day of one of the most successful ACCU Mentored Developers projects to date. We didn't slip with any of the chapters! I have learned a great deal and it's always good to have another completed book under my belt. On behalf of the group, and personally, I would like to thank Steve Freeman and Nat Pryce for their contributions. This project would not have been what it was without them. My personal thanks also go out to Mike Baker who stepped in on a couple of occasions, most critically when I had an unexpected spell in hospital. A review of the project will follow in the next CVu. If you would like to review the project, the email archives are available here:

http://lists.accu.org/mailman/listinfo/accu-mentored-growing

Book reading project

The original list of suggested books for the next project was:

- *Clean Code* by Uncle Bob
- *Refactoring* by Martin Fowler
- *Working Effectively With Legacy Code* by Michael Feathers
- *Continuous Delivery* by Jez Humble and David Farley
- *Domain Driven Design* by Eric Evans

Although all received a good number of votes, the most popular two books were *Working Effectively With Legacy Code* and *Domain Driven Design*. We employed an alternative vote system, where those who originally voted for *Clean Code*, *Refactoring* or *Continuous Delivery* voted for *Working With Legacy Code* or *Domain Driven Design*. *Working With Legacy Code* came out slightly ahead, and I'm delighted too that author Michael Feathers has agreed to join the project. If you'd like to get involved in the project, which involves the group reading the book at the pace of about a chapter a week and each member posting a review in turn, sign up to the main Mentored Developers list: http://lists.accu.org/mailman/listinfo/accu-mentored-developers

Grails project

A group has already been put together for another project based around the Grails framework. Working from *Grails In Action*, our mentors are Russel 'Groovy' Winder and *Grails In Action* co-author Peter Ledbrook. As well as reading, there will be a practical programming element to the project (you can't learn a language or framework without programming in it) and the possibility of a follow on project to develop a real world application. If you would like to follow the project please sign up to the list at http://vps.gnomedia.net/mailman/listinfo/accu-mentored-grails

PAUL GRENYER

An active ACCU member since 2000, Paul is the founder of the Mentored Developers. Having worked in industries as diverse as direct mail, mobile phones and finance, Paul now works for a small company in Norwich writing Java. He can be contacted at paul.grenyer@gmail.com

Desert Island Books (continued)

It looks as if there might be room for one more thing in the chest. Indeed, yes, a little refactoring will yield the space I need to slot in one further CD; not an album this time, but the PC Game, *Deus Ex*. An unusual choice, perhaps, but I think the mere existence of *Deus Ex* destroys several idiotic conjectures otherwise sensible people have spouted on the dangers of computer games stupefying society (I'm looking at you, Susan Greenfield). I should explain why. I will start at the end.

Three concrete endings confine how you finish the game. Up until then, your strategies are essentially uncountable (within a reasonably linear plot). Several extremely dedicated people have completed *Deus Ex* with the absolute minimum of kills. There are two points where avoiding death is impossible if you wish to progress. By the time you get to these points though, it doesn't matter, because:

You'll be completely immersed in the game. The quality of the scriptwriting, plot and character development will have convinced you that these deaths aren't just necessary, but deserved. The sheer attention to detail put into creating the dystopian future the game is set in is irresistible.

Suspicious players will find subtle hints at the real motivations and goals of the people and organizations they work with (and for). You'll frequently break into apartments and find out more about the owner (and, often some foreshadowing of coming plot) by what books they're reading, as well as the contents of their inbox. Datacubes within the game contain excerpts of, to name a few, *The Man Who Was Thursday* (G.K. Chesterton) and Sun Tzu's "*The Art of War*".

The whole game is as rich in social commentary as a good sci-fi novel; in particular, one conversation with a deliciously sinister AI named Morpheus yields some absolute gems (You can, should and must watch it on youtube, here: http://www.youtube.com/watch?v=COwfIhvRtNw). Each of the endings is paired with an appropriate quotation; one is from Milton's *Paradise Lost*; another, Voltaire; the third, Khalil Gilbran. I am yet to find another game so committed to being so flagrantly cerebral.

Your character grows his skills (don't pick swimming) not just by practice, but by nano-augmentation. You travel the world via stealth helicopter, visiting New York, Hong Kong, Paris and Area 51 before you're done; spending as much time on side quests and exploring as you do making real progress through the game.

Ten years on, PC Gamer has given *Deus Ex* the accolade of 'best game in history' two years in a row, in a remarkable instance of 'being right on the internet'. It is written that every time someone mentions *Deus Ex*, someone reinstalls it. As I seal my media chest, I fear I may spend most of my time on this sunny desert island putting on a trenchcoat and fighting conspiracies in this masterpiece of a game. There is a working computer on this island, right?

Standards Report: C++0x

Roger Orr brings us up to date with the latest news.

It has been a long time coming but we finally have a new C++ standard going for official ratification by ISO: the formal ballot should have started on 6th June.

This was the most important formal motion that was agreed at the most recent ISO standards meeting (held in Madrid 21–26 March). Three members of the BSI C++ panel attended the week: Alisdair Meredith (who many of you will know from ACCU conferences), James Dennett and me. We also had a couple of days from Mark Batty of Cambridge University who has particular expertise in the new C++ memory model. There were in total around 50 people present for the week and eight different countries were formally represented.

The event was hosted by Telefonica I+D, who not only sponsored the event but additionally underwrote the cost of lunch for all the delegates each day (which meant we all ate together in the hotel, giving us a chance to discuss 'hot topics' further over lunch).

The three major individual items of particular interest to the UK panel were discussed in Madrid.

Range-based for statements and ADL (N3257)

The new range-based for (syntax to support simple iteration over an entire container) has been implemented in a couple of compilers and people have now begun to use it in existing code bases. However there were some unexpected errors, for example with some versions of boost (www.boost.org), an ambiguity is caused when user-provided definitions of begin or end are available. For example:

```
namespace n {
 struct X { ... };
 template<typename T> void begin(T& t) { ... }
}
std::vector<n::X> v;

for (auto i : v) // ambiguity in range-based for
```

After a late night discussion and a plenary discussion we added a look-up for member names **begin** and **end**, before looking for free functions, to resolve the ambiguity in the common cases such as that above.

noexcept in the standard library (N3248)

Recent work has raised some issues with wholesale adoption of the new **noexcept** keyword for the standard library. (This keyword has similar semantics to the existing **throw()** but makes less onerous requirements on the runtime if an exception does in fact get thrown.) After discussion we agreed on a conservative set of functions in the standard library as candidates for marking with the noexcept keyword. This is a safer option as it is much easier to add the keyword later to more functions than to try and remove it from existing ones.

Race condition in exception_ptr

The **exception_ptr** class was added to the standard principally to support passing exception objects between threads and the draft standard allows the exception to be either copied or reference-counted. The BSI panel (and Anthony Williams in particular) dislike the reference count option as this introduces a data race into the new standard and we recommended enforcing a copy. However a majority of implementers stated that this was not implementable without breaking backwards compatibility on their ABI (so far as they were aware at the present).

Anthony Williams (another name known to ACCU members) joined the discussion via Skype but a straw poll on the issue resulted in a very heavy majority against making this change. While still hoping to get this changed in the future the UK members didn't think the issue was big enough to delay the delivery of the standard.

So what's in the new standard?

There are a number of new features of the language and the library in the new standard and everybody's list of the most important one is different.

There is a 'C++0x FAQ' at http://www2.research.att.com/~bs/C++0xFAQ.html maintained by Bjarne Stroustrup and this page also includes links to other web sites.

For what it's worth my own short list of key new features includes:

- the **auto** keyword

  The compiler infers the correct type for you, so you don't have to type things like: **std::vector<std::string>>::const_iterator**.
- range based for (like the example above)

  Provides a simple way to express iterating over an entire collection – works very well in conjunction with auto
- lambda expressions

  These allow anonymous functions/function objects to be defined easily.
- the memory model, atomic variables and basic threading support

  Obviously people are doing a lot of multi-threaded programming in C++ already, but the work gives a standard (and portable) way to do this.
- variadic templates

  This allows templates with variable numbers of arguments.

Some features of the new standard are already included in individual compilers – for example gcc 4.6 and MSVC 2010. (Scott Meyers maintains a summary of the feature availability of these two compilers at http://www.aristeia.com/C++0x/C++0xFeatureAvailability.htm). However, as the new standard is now in its final form not just a draft, compiler vendors should start to implement many more of the new features in future versions of their compilers.

I'm looking forward to making use of these features in my own code.

ROGER ORR

Roger has been programming for over 20 years, most recently in C++ and Java for various investment banks in Canary Wharf and the City. He joined ACCU in 1999 and the BSI C++ panel in 2002. He may be contacted at rogero@howzatt.demon.co.uk

Bookcase

The latest roundup of book reviews.

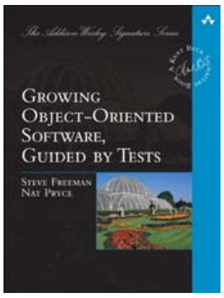
If you want to review a book, your first port of call should be the members section of the ACCU website, which contains a list of all of the books currently available. If there is something that you want to review, but can't find on there, just ask. It is possible that we can get hold of it.

After you've made your choice, email me and if the book checks out on my database, you can have it. I will instruct you from there. Remember though, if the book review is such a stinker as to be awarded the most un-glamourous 'not recommended' rating, you are entitled to another book completely free.

I must thank Blackwells and Computer Bookshop for their continued support in providing us with books.

Jez Higgins (jez@jezuk.co.uk)

Growing Object-Oriented Software, Guided by Tests



By Steve Freeman and Nat Pryce, published by Addison-Wesley, ISBN 978-0321503626

Reviewed by Paul Grenyer

I'll cut straight to the chase: This is a great book! In fact I'd put it second on my must read list for developers, behind *Test Driven Development* by Kent Beck. Testing is so important and while Unit Testing is becoming mainstream, many are failing to take the next next few steps to integration and system (end-to-end) testing. This book tells you why you should and shows, with practical examples, how to get started.

Growing Object Orientated Software Guided by Tests was the first place I read about the Walking Skeleton. Originally described by Alistair Cockburn, this is a technique I've been using for the last few years and didn't realise there was a name for. The Auction sniper example that covered by the middle chapters introduces not only testing techniques, but lots of useful and practical lessons about good design. The later chapters discuss improving your tests, including readability. The final two chapters cover testing threaded code and asynchronous code. Some of the ideas presented here were new to me and would have have been very useful refactoring exercises for some projects I used to work on.

If you want to develop higher quality, robust software, read and apply the lessons in the book.

Warning: The code examples in the Kindle version of this book are difficult to read and there are a few misprints compared to the paper version.

Web site highlight – The Free Software Foundation (FSF)

Reviewed by Ian Bruntlett

As you may already know, the FSF (http://www.fsf.org/) provides many of the utilities that Linux builds upon. They are also known to campaign against technologies that are threats to end user freedoms. They provide the GNU Operating System (http://www.gnu.org). They are working on an operating system kernel, Hurd but in the meantime their work is usually used on top of a Linux kernel, leading to such systems being referred to as GNU/Linux systems. GNU is a recursive acronym, meaning 'GNU's Not Unix'. Both Hurd and Linux are heavily influenced by Unix.

To support itself, the FSF has a shop (http://shop.fsf.org/). I've bought books from here in the past but have found that technical books tend to become obsolete pretty quickly compared to, say, a t-shirt (http://shop.fsf.org/category/gnu-gear/).

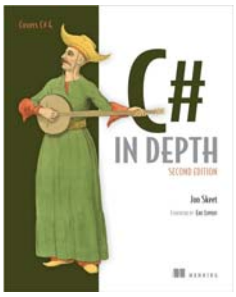
When learning to use GNU software, you are left with a decision: free download or purchase a paper copy? I have a handful of GNU books – however, when they are rendered obsolete (sic transit gloria mundi) I will replace them with (free) downloaded PDFs.

To get free GNU books, go to http://shop.fsf.org/category/books/ , click on the book you want and then click 'Download an electronic copy of this book'.

This isn't a comprehensive review of the FSF/GNU web sites but it covers what I find most relevant.

If you are using a Linux/Unix based system then there are two other sites you should know about. The first is O'Reilly (http://oreilly.com/) who publish a vast array of books. The second is Amazon UK – here in Northumberland specialist computer books are hard to get hold of and (compared to Amazon prices) very expensive.

C# In Depth, 2nd Edition



By Jon Skeet, published by Manning, ISBN 9781935182474

Reviewed by Steve Love

Highly recommended

To say that Jon Skeet is passionate about C# would be something of an understatement. The result of this is that he is prepared to investigate every feature, nuance, dark-corner and (importantly) implication of C# with patience and diligence, and then put all of his discoveries in a book. This book. In short, Jon Skeet knows C#. In ways many of the rest of us would perhaps find disturbing. Happily, the fact that Jon has looked into places we dare not, and written it down means we don't have to.

Jon's enthusiasm for C# positively boils over in the text,but that's not to say he's especially evangelistic about it; he's not afraid to highlight some of the warts present in any practical programming language (e.g. casts to and from enums), and doesn't shy away from pointing out one or two limitations of some featiures (e.g. not being able to use mathematical operators with generics). When the title says *In Depth*, it really means it.

A case in point is the book's treatment of Nullable Types. This is a superficially simple feature of C#2.0 with some syntactic sugar to make using it straightforward. However, there are pitfalls and details under the surface which may manifest themselves infrequently in

Bookshops

The following bookshops actively support ACCU (offering a post free service to UK members – if you ever have a problem with this, please let me know – I can only act on problems that you tell me about). We hope that you will give preference to them. If a bookshop in your area is willing to display ACCU publicity material or otherwise support ACCU, please let us know so they can be added to the list

- **Holborn Books Ltd** (020 7831 0022) www.holbornbooks.co.uk
- **Blackwell's Bookshop**, Oxford (01865 792792) blackwells.extra@blackwell.co.uk

View From The Chair

Hubert Matthews
chair@accu.org Members.fm

The AGM at the ACCU conference this year was a more lively affair than normal. There were a number of items of note. The first was financial with the publication of the accounts for 2009 and 2010, which made for unpleasant reading. I won't repeat all the details and explanations but sufficient to say that the large losses of 2009 have been reduced by half in 2010. These we hope to reduce still further in the current year. The other financial point was the increase in subscription rates. This was a necessary change and I thank the membership for their understanding and trust in this matter. The other item of note was that the discussion over having an elected conference chair was postponed until the next AGM because of a lack of time.

The fact that for a number of members the above paragraph will be the first time that they've found out what happened at the AGM raises the point that the ACCU doesn't have an effective or regular way of communicating with members. At the recent committee meeting we discussed this and decided that we want to improve matters. It is not desirable that only those members who attend the conference should be involved or should know what happened at their AGM – there are members who do not come to the conference because of distance, cost or problems getting time off work.

If the ACCU is to grow and attract more people we need to find ways of involving all of the members regardless of geography. We are therefore considering a regular electronic newsletter with ACCU news.

We also want to see if we can arrange some low-cost events (members' days) with a mixture of talks from sessions given at the spring conference alongside talks from first-timer speakers. Learning and improving are key values of the ACCU so this would be a great way of developing our members' presentation skills in a friendly and supportive environment. We might also be able to invite sixth-formers from local schools, as happened at one of the autumn security conferences at Bletchley.

Another idea was to hold these members' day conferences in different parts of the country to reach more members and to help kick start local ACCU groups. Broad participation is one way to engage current and prospective members – the ACCU is a community and we should seek to foster and nurture that feeling. Many software developers work in relative isolation, either physically through working in a small group or company or mentally through a lack of knowledge of current practices and techniques; the ACCU should seek to try to draw those people together and provide a long-term structure that is lacking for so many of them.

This is a long-term ambitious vision that will take a lot of time and effort to make happen but one that, I believe, is where the ACCU should be heading.

Bookcase (continued)

'ordinary' code, but when they strike, can cause hours of frustration in tracking their cause. Jon Skeet investigates Nullable Types in intricate detail, meaning that you can at least be forewarned of some of these problems and either avoid them, or recognise their symptoms when you're bitten by one.

The book progresses in an orderly fashion from C#1.0 to C#4.0 – current at the time of writing. The main thrust of each chapter is to show how each version of the language builds on its predecessor to make something easier, more natural or, in some cases, possible where it had previously not been. In particular the changes to delegate types in each version of the language are explored to show how to get the best from them in a given incarnation.

This is not really a book from which to teach yourself C#, and the back-cover even mentions the assumption that readers have a knowledge of C# basics. If you know C# well enough to write meaningful programs in it, then you should read this book. I defy you not to learn something you didn't know before!