Transfer Narratives

(All have been anonymised. In the anonymisation, gender has been preserved. Additions in square brackets are editorial.)

Frank

Idea – difficult for students to relate to text-based programs since grown up with rich GUIs. Not sure where source for this was. We use BlueJ so graphical interaction there from start (Perhaps, thinking of it, it's an idea of Barnes & Kolling).
Practical incorporation – in past years did not introduce material on GUI construction on basis that more interested in core OO concepts. Applied idea & made GUI based programs focus of last 3 weeks. To get there, also introduced preceding 3 weeks of text-based I/O programs. So actually idea evolved into making "whole programs" & satisfaction gained from that – using BlueJ we'd never fussed about stand alone. The "whole program" idea turned out to be very productive. It seemed to give final stage of course more life & students really enjoyed results. We also passed idea in direction of executable jar files. So students could create programs that ran by double clicking, just like all the other programs on the computer.

Benjamin

When looking at assessment in the Commons I re-appraised our second semester students and noticed the long tail of weak students were dragged through all of the content during the semester and coming out with 30%-40%. They have minimal understanding of the learning outcomes. So why not have them learn the first 30%-40% of the unit well? At least they would have the underpinning conceptual framework for later.
Adopting a book for the entire year helped here. With assessments after each chapter. Students work through the chapter before doing the assessment … and it takes as long as it takes.
Problems … slow students were still doing the assessment for the first chapter (out of 6) in week 8 out of 13! There was a sudden scramble to finish what they could. These students are still coming out with 30%-40% but are happier (how do I measure that?). part of the success is down to individualising the assessments. Each student has a company allocated and the assessment is written in generic terms.
Unforeseen problems … I'm radical and revolutionary because I do not cover all the learning outcomes. But these students understood very little of the learning outcomes anyway. At least this way (I hope) they understand 80%-90% of half of the required content.
Students repeating have been very positive when comparing the different modes of presentation (good).
Some weak students complained they did not do the work because the did not know when assessments **needed** to be done (bad).
Having this approach has difficulty when students with poor learning skills or who have been brought up with acquired helplessness are let loose and given responsibility for their learning.

Joseph

The idea: Walter's gradebook concept – where the coursework consisted of a series/sequence of exercises and students could complete these in any order up to the end of the semester

Why do I like? Fits with a) my understanding of itp learning as *stepwise* and b) enabling students to have control and go at their pace – if they fall behind on one week that's OK, as long as they catch up. This compares with regular drop-dead deadlines, where a missed exercise is "lost" to their final grade on the coursework and so there's no point in going back.

Incorporated: Each week introduces 3 new exercises. Completion of an exercise merits a tick. Students must attain 2/3rds of all ticks on offer in a semester by a couple of weeks after semester end, in order to gain course completion criterion. Support staff developed an on-line tick management system.

Did it do what I expected? It certainly introduced flexibility and did allow students to work at their own pace. Of course, this means that less well-motivated students could leave it all to the last minute—entirely against my intention—and many did this.

There are a number of downsides to this particular implementation

- Cannot issue model answers until session end
- Management of ticks is time-consuming
- Students leave work until last minute
- Tutors can't directly discuss the exercises

Keep it? Change? Still believe that ability to catch up is important – crucial – for students who have fallen behind … There must be an incentive to do this. However, there **must** also be an incentive to complete work in a timely way – because it is **very important** to keep teaching/feedback sessions in step with the exercise work as much as possible.

1 option – reduce the time interval from 2 1–semester blocks to, say, 6 4–week blocks? Or require that a tick **average** is maintained (but then this is hard to see at a glance).

The ***key*** is that students are all brought up to a common point, regularly, so that they are not left behind – disenfranchisement leads swiftly to demoralisation.

(A colleague in Engineering has a great design in this light for Engineering Maths from a few years ago—must try it!)

Elizabeth

The idea came from my visit to Emma at N university on my observation visit. I first watched her give a lecture on arrays in VB-NET. This was interesting as she did it a bit differently to me. But after that, a subgroup (tutorial) of the class had their next session with her. It was a programming class but it was in a classroom! I had never experienced classroom sessions in my time as a lecturer for teaching programming. Always the model was lecture—lab, both when I was a student and a lecturer. So I observed this one-hour class and saw Emma using the whiteboard, explaining further the concepts covered in the previous lecture, answering questions and interacting with a small group (about 10) students. The students had their textbooks out and a single A4 sheet of programming exercises they were working on for that session and were

working alone or in groups. It was a lively and productive session. Their next class would be where they coded the designs they had been working on in this session.
I realised almost straight away that this was something I wanted to implement back in my module. The benefits were that students were more likely, in a smaller setting, to ask questions, it was easier to give remedial help, facilitate group working, and even get to know **names**. Whereas in a lecture (and even to a large extent in a lab class) these things were very much harder. Also, I hope that by forcing students away from the keyboard, they would have to think about program design and testing more thoroughly than they did at the PC.
Back at my place, I put in a request for timetable changes. Currently each student had 2 lectures and 1½ hour lab, and with 4 groups that was 8 hours/week tutor time. My proposal was to have 1 lecture + 1 seminar + 1 lab, x 4 which took contact hours to 12/week so management was not keen but eventually agreed to it. I had dropped the 1½ lab to 1 hour, but it was still an overspend to implement it. So how did it go?
The benefits were that I got to know my tutorial groups much better this year. I made a point of learning all their names early on and using them whenever I could for practice, which they thought funny. It put our group/relationship on a good footing from the start. Students found it helpful to have the extra help in designing programs, and they didn't waste time once in the lab, or struggled to know what to type. Everyone got on with their work straight away. What helped was that some of the exercises were going to be selected (by each student) to go into their personal portfolio. I think it gave students more confidence, especially the weaker ones.
In semester 2 it didn't work as well. I think it was because I didn't think out the tasks in the classroom so clearly, and needed a bigger variety of tasks. Some students said they wanted more lab time, too. On the whole I think it was a very good innovation and I will keep it and improve it for next year.

Daniel

There have been several changes to the itp module and my delivery practice that have occurred, and the majority of these arose simply from conversation and thinking time across the period [of the *Commons*] and so cannot be attributed to a single event.
One that can is the use of Post It notes to elicit unstructured, genuine feedback from students on an ad-hoc, as needed, basis. The idea came from Frank and I could immediately see how I could use this in my lecture/lab context. Like many good ideas, it was simple to issue to students, simple for them to use (for those that had pens!) and easy to interpret. It has the particular benefit of asking for as much or as little as a student was prepared to give, at a frequency determined by me.
I used it several times in years 1 & 2 (particularly in the first term) and made sure I fed back something on the comments in the following week.
Comments ranged from insightful and helpful stories/views from students filling both sides of the note to diagrams of dubious content. I suppose the latter at least revealed the students were willing to share (anything).
I am, in early June, going to share this approach with colleagues in our Teaching and Learning Committee meeting on Formative Feedback. (They believe my contribution is on formative feedback to (rather than from) students!). I am intending to show some examples.
I will continue to use this approach in years 1 and 2. I have no desire to use this with year 4 as the culture I need to instil in the class is inconsistent with this method.

It is definitely more useful than more formal feedback and whilst it is not academically clever, it is a smart solution.

Ernest

Ever fecund (I think that's what they're saying about me) I'll do three. (There were more)

Practice 1 Herbert + reading	Using Python/writing an adventure. Python came from Herbert mentioning it and my feeling obliged to remove my ignorance of it. The adventure bit came from perusing the ACM library for papers on teaching programming (monkey island …) – which perusal had started as a result of the *Commons.* I needed a re-vamped ITP course, I put the two together; it worked; I'll take it further for my own pleasure.
Practice 2 Group discussion reading	Minute Papers. This came from a general discussion re: feedback on student understanding; I can't remember who I stole it from. Basically the students write down one thing they didn't understand in a lecture – anonymously – and the "notes" are collected … the "winner" is explained again. I later learnt these are minute papers. They worked. Well. Also I founds that often I couldn't have predicted what they would choose. If I'd been carrying on I would have recorded my prediction against the students' choice.
Practice 3 Elmer/Walter	This came from a small group discussion with Elmer and Walter where we talked about starting the course again in week 3 (and 5) (and 7). In week 3 I split the practical into 2 1-hour slots. The first hour was the material for week 3, attendance compulsory. The second hour was a seminar on week 1, attendance voluntary, very informal. Worked well – especially as some of the "stronger" students stayed and did some mentoring. It carried on like that.

George

This year more laboratory time was allocated to students learning software development; an increase from 90 minutes per student last year to 2 hours this year (although in some way this was nominal due to time not being strictly enforced).
The idea was based on experience at the *Commons*, where the previous allocation seemed insufficient. In some ways it was made possible due to lower student numbers (not increase staffing or teaching hour allocation). As such, incorporation was trouble free.
The successfulness of the approach was largely counteracted for two reasons. First, the introduction of a weekly assessed logbook removed 1 (of 2.5 allocated) lectures. Second, a prolonged teaching visit to China removed 1 lecturer from much of the delivery. Essentially this meant less tutor accessibility than before.
The change is certainly worth keeping and it is clear that other constrains would have left even less teaching contact time available for students. However, more staffing is also needed, or more use of online resources, such as Moodle, by students, needs to be encouraged.

Ideally three contact lab hours per student should be allocated, based on discussions with other *Commons* members.

Elmer

The most direct example of transfer came when talking to Ernest about the incremental nature of programming. I spoke about someone I know who teaches communication, who organises the teaching around 6 topics, each of which is separate from the others. This allows students to continually start again, if they don't get on with a topic. Ernest said "wouldn't it be great if we could do that" I reorganised the unit into 4 blocks (á la Daniel), with each block starting with "what is programming about?" Each block then head off in a slightly different direction, but the dependencies between each block are minimised – at least less so than before. I will continue this again, as it was partially successful.
The integration of teaching and research, epitomised by many of the participants [of the *Commons*], also influenced me to explicitly collect data. Some of this will be useful, but I need to work on it.
In summary, I am reminded of a story in the Perishers cartoon. One of them (Marlon) sets out to be an inventor. After a day he has "invented" the wheel and the biro. When this is pointed out, his reply was "Yes, but I'm catching up!". I feel like Marlon.

Edward

About two years ago I decided to use the Standard Template Library (STL) in my intro programming course. I had attended a session @ SIGCSE where the idea was mooted and debated. I wasn't convinced, but decided to try it.
I dropped the idea after one attempt. The principal reasons were:
1. the STL provided a huge number of useful sources, but they were aimed @ experienced programmers and thus incomprehensible to novice programmers. For example, to add an item to a list one used the "push-back" operation. From a novice perspective the name is not just meaningless but actually problematic. What does "push-back" mean?
2. traversal involved iterators. Whilst not a major problem in themselves they are, in STL, equivalent to pointers and data access involves the * (C++) notation. The notation is completely alien to novices. Actually, it is probably alien to many "experienced" programmers
3. Most significantly, the STL disguised *too much*. One of the key aspects of learning is integration. Integration occurs when relationships between items are recognised. It is impossible to recognise relationships when one is at a relatively high level. Thus learning suffers
For example $e=mc^2$ is generally regarded as the most famous equation in the world. People who understand its implications do so because they understand the relationships it represents. If I write it as c=log(e/m)½ the it will still produce the same result but it sunders the neat encapsulation of the relationship between the entities.
Using the STL to build a linked list or dynamic array or trees etc. inhibits the students ability to experience and recognise the relationship between pointers, looping, structures and other aspects of CS and programming in general.

Daisy

1. One thing I changed was to ask for small feedback during my lectures

Where did I get this idea from? First of all, from Sally, as she mentioned once several ideas how this could be done, but then, more on a practical level, as she actually did during the meetings. Secondly, this was inspired by the handsets Joseph uses in his courses. OK – I do not have these handsets, but I try to ask my questions in such a way that they help clarifying things which might have been unclear before.

Did it work? I would say – yes. Though sometimes this takes more time as expected – so it needs to be planned well in the future. But also spontaneous things work.

And, maybe, that is also important, the students seem to like that.

One final thing – why this is important to me: I always thought my lab exercises (weekly, to be demonstrated and ticked off by me or an assistant) are the strength of my teaching. I can directly help in the labs & have a lot of feedback. Things are changing – I have not the time anymore to attend the labs – therefore the above introduced method.

2. A second thing I once spontaneously changed: somebody (Elizabeth) in the Commons said to me: if I did so many assignments and exams my department would say this course is over-assessed. This made me think, and finally I skipped the 50% exam at the end of term one. This has not only the effect to avoid over-assessment, but also to keep students interested until the end of term 2 (and the exam there).

3. The biggest change for me however was, beside confidence, to realise that not only thinking of *what* to teach is important, but *how* and *why* as well.

Walter

When I started the *Commons* I was giving 3 1-hour lectures each week & students had an optional 2-hour lab class for logbooks. Everybody at the *Commons* was lecturing less & having more practicals.

I still am allocated 3 slots/week but I make it clear that one is a student-led revision lecture. Everybody arrived for the first lecture but I made it clear that I would only answer questions. Attendance dropped from then on and the revision lectures lasted typically 20-25 minutes except when courseworks were assigned.

Lab classes were made compulsory and started with a lab-task which the student had to have signed off before working on their taskbooks. Lab tasks were dynamically written to reflect the material taught in the lectures. Because of the difficulty in doing this the lab tasks stopped about half way through.

In general I intend to increase this trend. I intend to be allocated only 2 lectures each week and make some of them revision lectures. There is enough time to do this. I intend to create and enforce lab tasks and possibly create some mpeg lectures to put online.