

Prudence and Safety

Brain Overload

As we said right at the beginning, mapping and packing are different. The difference can be seen in the organisation of the workplace, and the behaviour of the people within it. A packer organisation sees all work as consisting of a mechanical series of actions, that are performed at a particular place. It's not that cynical managers believe that putting everyone in open plan offices, thus stopping them concentrating and hence impacting the product doesn't matter because their goals are short term - it's that they don't believe that there is any such thing as concentration (as mappers know it) going on in the first place!

Some work environments can be so packer-orientated that they render mapping impossible. There will be constant interruptions, preventing a mapper attaining flow. Meetings will be structured as a series of 'posture statements' from individuals that are scored to identify winners and losers without regard to the comparative relevance of any particular sound bite. In this situation, a mapper whose considered thought may require two, or even three sentences to enunciate, will simply be seen as a pathetic loser.

Worse, the very ineffectiveness of the packer cognitive strategy leads its users to become very defensive when the etiquette that conceals all the packers' lack of understanding is breached. If a small proportion of the staff are mappers (or even a large proportion if they don't know what is going on), acrimonious situations can emerge.

The key point in this picture is that there is no point in the mappers trying to persuade their packer colleagues of the value of a rigorous and complete approach with ever more careful arguments - the problem is that the packers aren't ready to accept any sort of detailed reasoning in the first place! So a mapper can work themselves sick trying to reason with people who just aren't listening.

You really can work yourself sick when doing mapping, and it is very important to watch this and avoid it. The first thing to do is recognise situations where an intense mapper approach is appropriate.

All mapping can be seen as a search, and the thing about searching is that one does not know where the sought thing is. Therefore one must usually undertake to continue the search until the object is found. This is much easier to do if one can have confidence that the object of the search actually exists! Otherwise, one must impose some sort of artificial termination such as a time limit. This is where the 'mapper faith' we mentioned at the beginning comes in - mappers discover over and over again that the natural world is always simpler than it looks, providing that one looks at it in the right way. Sometimes great hidden complexity must be uncovered and explored on the way, but the simplicity, the necessary sufficiency of the 'quality plateau' will be revealed in the end. In all situations found in the natural world, a mapper investment will be worthwhile, because the deeper the hidden view, the more worthwhile (powerful) it will be. Situations that do not involve the 'natural world' in this sense (after all, everything in the universe is 'natural') are those where consciousness has acted to create a local area of irrationality. In other words, where you are playing against another mind, which by accident or design is setting out to confuse you, by setting out to show you only parts of the whole system (which is rational) so that what you see appears as complete but irrational. So packers, by using the same language as mappers when talking about thinking, but meaning something different, appear to behave irrationally. When one adds the mapper/packer communication barrier into the picture, we're back in the natural world, which includes the mind that is the opponent, and rationality is restored.

There is a radio panel game called *Mornington Crescent*, whose rules are for some reason never actually published. If one listens to the play and assumes that there is a rational system of rules that is being adhered to, one will go mad. There are no such rules. The real game consists of the adroitness with which the players make it appear that the rules do exist, and that the game has a particular character.

So if there is another mind in the situation under consideration, its potential perversity must always be considered to ensure that the situation remains natural. As computer programmers, this might seem to leave us in a paralysis of paranoia,

because many of the problems we deal with involve users. But things are not this bad because if a business activity has existed for some time, it is going to be a consistent natural phenomenon that is amenable to mapper analysis, even if none of the human players actually understands what is really going on. Remember however, that short lived business activities, such as types of transactions offered by merchant banks for short periods only in a perpetually changing market, may not actually be sustainable, and thus must be treated as the products of perverse minds. This doesn't mean that the transactions aren't automatable - just that the only thing to do is code up the madness in your 4GL or whatever RAD tool you are using just like the risk managers ask you to do it, and let them worry about renormalising their own behaviour with respect to their equally perverse peers. Sometimes, organisations that do this sort of thing ask mappers to look at the whole situation and see if they can find any logic if the boundary is wide enough. These jobs can be extremely interesting and rewarding.

Having identified situations where we should either abandon mapping or redefine the problem, we are left with the problem of how long understanding will take to come. Experience with mapping does build up a bizarre kind of intuition about intuition, which can often give a good sense of this. Don't pronounce estimates until you have convinced yourself privately that you have built up enough experience to do this. Write down your own private estimate before you embark on a mapping job, and see if you are right when you're done. Even with decades of experience, you can still be wrong. If the job was understood, there would be a COTS product, wouldn't there?

Mappers often contemplate problems for many years before they crack them - the investigation culminating in this course took either thirty years or six years, depending on where the boundary is set! It is important that while there is a limit to the intensity one can bring to bear on a problem, there is no limit to the duration of a state of alertness with respect to a problem, save the lifespan of the mapper. There is no disgrace in recognising a long haul problem and reducing the intensity of one's contemplation. This attitude produces one of the most hilarious examples of the mapper/packer communication barrier. To a packer, a project consists of a series of actions to be performed. The speed with which the actions are performed indicates the efficiency of the worker. Working on a problem, then appearing to leave it alone, is evidence of shambolic disorganisation on the part of the worker. Hence packers are able to be extremely patronising about mappers' `curious projects', while recognising and simultaneously dismissing the remarkable creative results that mappers regularly deliver, because they clearly weren't attained `correctly', although the packers have no suggestions as to what the `correct' approach might be. Poor things. Education has a lot to answer for.

With the rules of engagement laid down, we must next address taking care of one's self while doing intense mapping. Basic physical health can be compromised in two ways. During intense engagement some mappers find that nutritional and exercise needs are left unattended. Be proud of what you are and what you can do, and make sure there is plenty of fresh fruit and a full freezer before entering the state. Then eating is easy. Every mapper seems able to work well while taking a solitary walk, so take walks.

The second way to compromise your health is by overfilling your brain. The following advice is not to be taken literally, as we have no neurological basis for it, but this is what it feels like to some mappers who have experienced the difficulty. If the problem is a big one, the complexity of it, that has to be held in the mind in one go before collapse can occur, seems to fill more and more of the mapper's brain, and it starts to occupy the brain parts that hold the mapper's body image. When this happens, physical fitness can collapse in days, and a limber, trim person can turn into a stiff couch potato very quickly. We aren't saying don't do this - it really is up to you. But if you stop within a week of getting suddenly slobby, and regain your body image by working physically and getting some feedback, you can retrieve your fitness as quickly as you lost it. Swap your body image out for too long though, and it gets much harder.

Remember what we said about not wasting energy by repeating unproductive cycles of thought.

Remember that things are also happening in the outside world. Your personal relationships need maintaining, and while some of those close to you will recognise the state and wait for you to reappear, others will need some contact. Practice working in background, by using the `plate spinning' technique we described earlier. With time, you will find that you can vary the amount of your mind you give to contemplation, and the amount you leave free for holding witty conversations. If you're at a stage where you need to commit a lot of your mind, you don't want to stop because it will take a week to retrieve the partial picture you have, and there's a function you're expected to attend, you can always just go in happy idiot mode. You know you're paying no attention to the prattle around you, but incredibly, the prattlers rarely do!

Pay no attention to the opinions of packers as to your unhealthy ways. Such comments are nothing to do with the genuine health considerations we have discussed in this section. To packers, `thinking too much' is a disorder in itself!

Brain Overrun

Mapping is an intense, absorbing emotional experience. Every problem collapse is exhilarating. The trouble is, we hit the peak of intensity and exhilaration, and then the damn thing is cracked, and there's nothing to engage with any more. This can lead to a danger of depression at the end of a project, because the fun has stopped. It can also lead to a mind racing around and around, skipping around what is now a very simple structure in an extreme example of going around and around an unproductive cycle of thoughts.

All this stuff is bad for you. If you have been working within a team, do talk down sessions. Give yourself something useful to talk about by analysing how you approached the problem, and what you have learned about the class of problems from the specific one you just tackled. What have you learned about your platform? Is it cool or bletherous? During talk-down sessions, remember mood control. The idea is to wind down, not up. Replace the pleasure of cracking a problem with a celebration of your success.

If you are not within a team, try to get into a totally separate activity involving others as soon as possible. There is always a problem in that when you are engaged, you don't think about your social calendar, and as soon as you're done, you can get into a funk too quickly to sort yourself out. So either have the wit to invite some friends around in advance, or go blind visiting.

If the worst comes to the worst, and you are stuck on your own with a solved problem, get it over with as soon as possible. Get your trophies, your listings, your diagrams, your product, get utterly loaded on the poison of your choice, and spend an evening gloating. It might seem highly self-indulgent to do this, but it recognises and deals with a genuine emotional cutoff that is related to bereavement. Just don't overdo it - one evening, then go get a life!

Overwork

Don't confuse genuine mapper intensity with packer work binge displays. Remember that mapping is all about leverage, and get things worked out in your head so that what you actually do in the physical world is limited to necessary and sufficient right action.

Bear in mind your personal layered process, and evaluate your response to the situation by asking if the plan it currently represents is appropriate. That way, being at work or at home is a practical and objective issue, not a moral one.

Cultural Interface Management

Be aware of the need to manage the interface between the mapper values necessary to produce software, and the packer values that usually surround your project in its commercial environment.

Do not get involved in discussions without establishing the ground rules for rational thought. Claim the space to make a structured case.

When there are choices to be made, don't try to lay out all the logic as you might want it yourself. Remember that your need to be informed of all the facts so that you can make your own decision is not shared by packers. Don't try to explain why the optimal solution is correct, either. This just incites the packer need to score political points (the only way to survive in the infinite chaos) by arguing with you. Instead, teflon yourself by working through several options with costs and benefits, and restrict yourself to making sure that everyone understands the options. Packers can do this - it is how they buy washing machines and Double Whammo Choco Burgers.

Declare ambiguities and manage them. A useful buzzword is to call this exercise a 'Risk Parade'. Identify the unknowns and post them publicly with an estimate of their becoming a problem. Update the Risk Parade when things change. Present these data either formally or informally, but make sure everyone knows where they are.

Be willing to use the phrase 'I don't know'. This act of simple honesty can deflate no end of packer pomposity and

coercion, while leaving you with a clearer understanding of where you are.

All these techniques work by addressing a basic problem. Packers want to remove complexity by shifting blame onto others. Unless you act to prevent it, you can find yourself 'responsible' for the difference between the real world and the packers' wish-fulfillment fantasies. By acting to place the realities in a publicly visible place, but without foisting them onto any one individual, you actually help restore your whole environment to sanity while saving your own butt.

Individual Responsibility and Leadership

Mappers are used to sharing and aligning their mental models. They can then easily refer to aspects of those models in casual language to increase mutual knowledge. They also put emphasis on doing things optimally, and seem to be more comfortable with the win/win model of co-operation rather than win/lose.

All these factors lead to a general tendency that emerges when mappers get together to share problems and solutions and educate each other. This co-operative tendency is an important part of the hacker culture.

The simple fact is the techniques of mapping, particularly mapping in a given domain, are a craft art. Whatever we do to quickly upload new languages and notations to programmer's brains in formal courses is only ever the icing on the cake. The real training happens on the job, as experienced people show newcomers techniques they may find useful. The newcomers themselves then evaluate what to use and how to use it, in the light of the state of the discipline as they enter it. This is one way that our field evolves quickly.

We can either work with this fact and foster it, to take control of our own development, or we can ignore it and play with 'skills summaries' that list programming languages. We propose that a sensible way to take control has already evolved as a natural result of the problem types. Our industry abounds with formal but arbitrary categorisations, but the one we are about to offer is informal but real, and already exists, needing only to be openly considered in the workplace.

Traditionally, those starting to learn artisan skills are referred to as apprentices. They are entrusted with real work from day one, but always under the close supervision of a more experienced worker. When it is evident that the close supervision is no longer necessary, the worker is recognised as a journeyman, who can be trusted to do a good job and guide the apprentices he may need to help him.

Many competent workers enjoy the activities of a journeyman, practicing their craft, and remain as journeymen for the remainder of their careers. They prefer that another person, perhaps of a different temperament, takes responsibility for the success of projects. Such a person cannot be created by nomination. Either the large amount of journeyman skill, drive and insight into the nature of the craft are there, or they are not. While the development of a master craftsman can proceed with the guidance of others, it is the new master which must find his own voice. The subsequent regard is fully, and properly accorded to the student, not the teachers. To become recognised as a master craftsman, a journeyman must produce a masterpiece. In this, the craftsman demonstrates his (or her - this is olde language ability to create a workpiece of exemplar quality. In olden times, when the work was with material goods, masterpieces were somewhat overdone, because the new master would wish to demonstrate a range of skills, and would probably never produce anything as baroque again. The later stuff would be more directed towards an actual purpose, and so better fitted to its task. Thus the masterpiece was actually the lowest level of masterly work, not the highest as common usage would suggest! Today, a masterpiece is a whole system delivered at the quality plateau, and the only difference is that we abhor unnecessary bells and whistles. The masterpiece is still the first system, and all subsequent ones should be better, as it is the experience of all good programmers that we are always learning. One of the reasons that it is easier for programmers to learn from each other is that we are all aware that whether we are at any moment teachers or students, we will both be in the other position pretty soon.

There are a couple of consequences of recognising the craft model. Firstly, it produces maximal development and maximal productivity at the same time. The master craftsman controlling a project must ensure that each team member is challenged within their capabilities, but at a stretch. There is no shortage of jobs for competent programmers, so finding the challenges is not a problem. This requires the worker to make an effort, which not only pays direct dividends in the care that is taken, but also ensures that resources are actually used at the margin of efficiency, which is what the accountants want to achieve, but cannot within a procedural model that copies a packer repetitive industry. No two cathedrals or systems are alike.

Another consideration that all programmers already know but which is worth repeating in a win/lose packer society is that the fear of teaching one's self out of a job that worries so many professionals just does not apply to us. We are at the beginning of a new cultural Age. Look about you and see if you can see any way society could be made more intelligent. Have you ever tried to buy a house? There will be no shortage of work for programmers for a very long time, and if there ever is, well the robots will be doing everything and we'll just program cute graphics for running at our saturnalias.

The False Goal of Deskilling

This section will explicitly make a point that has been mentioned several times in this course, because it is an essential distinction between the mapper and packer views of the workplace.

The packer worldview is not natural - it has to be trained into a child instead of the development of the child's natural exploratory faculties. It was probably a low-cost form of minimal education and maximal organisation, from the beginning of the Agrarian Age to the end of the Industrial Age. In it, humans perform repetitive tasks in the material world.

The mapper worldview utilises and develops the natural human mental faculties for exploration of ideas, and is the unique preserve of humans in the Information Age.

Programming is a mapper activity. If we really had to repeat the writing of the same program over and over again, some bright programmer will produce a COTS product, and that is the programmer.

The traditional packer view towards any job is to assume that it is therefore repeated, demeaning labour, and figure out how to do it as simply as possible, optimise this, and if not actually automate it, deskill it to reduce labour costs.

To assert that the packer strategy that works for bales of hay and cogs works for all jobs, just so long as many people are engaged in them, denies the movement from a brute material economy, where a human is a poor substitute for a horse or a steam engine, or even a numerically controlled machine, to an information economy where menial labour is less necessary than understanding or creativity.

The film *Saturday Night and Sunday Morning* begins with a mechanical engineering worker having a boring time mass producing components on a machine tool. 'Nine hundred and ninety-eight... nine hundred and ninety bloody nine...' he complains. Grim though its appearance was, this was actually an age of remarkable enlightenment compared to modern times. In those days, the managers paid workers by the workpiece produced, devolving real power and incentive to optimise to the worker. In a software context, we should not attempt to control every aspect of the typing in of 1,000 lines of identical code every day, we should be asking why the worker hasn't written a macro.

The deskilling concept is right through our society, and this makes it pernicious. Muddled thinking can slide it by you, but any argument that uses it is bogus. Never forget this.

Bearing in mind the impossibility of deskilling programs, we can examine a couple of myths. Even in the realm of material mass production it is hard to compare like with like. Sure, we can compare this month's production of motor cars with last months, and see if we are doing better. But last year? We were using three different kinds of light cluster units then, and only offered five colours. Ten years ago? Every aspect of the technology, from anti-lock brakes to engine management systems to air bags to traffic announcements to fuel composition has changed. Real insight is needed to even tell if we are richer than our ancestors! Academic attempts to do wage/price comparisons over long ranges fall back on the time investment required to buy a housebrick and a loaf, because those are just about the only things you can just go and buy across many centuries! So what on earth can we make of the wonderful exponential 'productivity improvement' curves associated with each new 'breakthrough technology' that will enable you to staff your project with orangutans and get ten MLOCS out of them per second. What on earth can these curves be comparing in our ever-changing field? One must conclude that there is an awful lot of rubbish being talked of very statistically shabby work going on.

And what of the 'user friendly metaphors' that mean that the orangutans can now do anything they like, no skills required? We suggest that the true situation is that some sections of the market have been exploiting the myth that deskilling of complexity management is possible, and have been offering products that on superficial examination over a short time do in fact seem 'easy to use'. The trouble is, users actually have to do things like configure their IP addresses, firewalls, disks, scanners, printers, share drives, accounts and so on. At this point we discover that instead of a computer that requires no

skills because it pretends to be another piece of furniture such as a desktop, we have a computer that relevant computer skills don't work on, because after all, a desktop doesn't need to have its user accounts configured, so there are no such things as desktop user account configuration skills out there to be made use of. We eventually discover that even in domestic situations where all one might wish to do is pick a new IP without reloading the whole machine, shareware systems that admit that they are computers are more user friendly than the so-called 'user friendly' stuff.

Escape Roads

As professional programmers working in real commercial environments we often work under deadlines that we cannot guarantee to complete a real quality plateau solution within. An important part of the personal layered process, and the informal project management plan in sufficiently mature organisations is therefore the definition and continual re-definition of our contingency plans.

The most common kind of contingency plan is sadly based in dropping functionality. This is rarely an efficient way of recovering time, because most of the lower layer functionality usually still needs to be present to support the reduced application layer stuff, which should be a low-cost activity anyway if the lower levels are providing the right kind of application layer specialisation.

We suggest that the following approach is much more effective:

- First, get your basic layering right. Get the essence of each layer's API defined.
- Second, invoke Ken Thompson's dictum, 'When in doubt, use brute force.' Define a bloated, high-cost, inefficient, hard-coded and ugly way of providing the functionality within each layer. It doesn't matter that the whole system might well simply not work if it were actually implemented that way, because it won't be.
- Third, get on with providing each layer at the quality plateau. Revisit your crude technique from time to time to add the good bits that you do possess, and fill in the rest using possibly different crude techniques.
- Fourth, when the difficulties start, make an optimal decision in the light of your customer's short and medium term needs, your own risk parade, and the time available, as to which bits you will deliver crudely, and which bits will still be well done.

This approach has the enormous benefit that it enables you to do whatever is the best thing at the time. You cannot do any better for your customer than that.

When the layers can be implemented crudely, and if you have the code fragments you've written to test out you OS or specialist library API to hand, you can often actually implement the crude version very early on. This gives every programmer a common set of test stubs, significantly derisking the simultaneous construction of all the layers.

New Member Integration

Be kind to new members who are joining a new team. Like everything else in this course, we are not referring to a wishy-washy sanctimonious 'welcome wagon' ritual: we mean something very practical.

The team has a mental model of the job at hand. Share it with your newcomer. Make sure they understand what Situation Rehearsals are, and attend them. Explain the goal of the project, and then explain all of the external (customer facing) and internal (mental model) language in use on the project. Take them through the development environment including tools, configuration management, compilers and so on. Don't make them ask about each stage.

Don't ever make the mistake of carefully making sure that they have a desk, a chair, a workstation, but no account or anything to actually do. The worst thing when arriving on a new project is to find one's self sitting there like a lemon, with each minute stretching into longer subjective interval than the last.

A very sensible practical idea used very effectively at BT is to introduce a newcomer to an official 'nominated friend'. The

nominated friend is a peer who has been on the team for a while, and is explicitly introduced as the source of information, whom it is 's-1OK\s0 To Bother', about the kind of stuff a new team member needs to know. One of the best things about this approach is that being peers, the nominated friend will actually know the real answer to questions the newcomer will ask. Paper is usually in the brown cupboard, but the A3 stuff for the big diagrams is in the green cupboard downstairs.

This file last updated 10 November 1997

Copyright (c) Alan G Carter and Colston Sanger 1997

alan@melloworld.com

colston@shotters.dircon.co.uk