PhUSE 2019

Paper PD02

10 Useful Skills You May Have Learnt Through Being a Programmer – Other Than Programming

Paul Grimsey, Roche Products Ltd, Welwyn Garden City, UK

ABSTRACT

Modern programmers need to have a number of different skills in order to function effectively in the modern working environment. Demands such as: working in multi-disciplinary teams; working remotely; changing timelines and/or changing requirements, require the programmer to be adept not only at coding but also in these other, non-technical aspects. It is possible that the programmer may not be aware that they have these skills, as they are not easily taught but rather honed over time and through experience. This paper will describe what the author considers to be some of the most useful of these skills, that can be used outside of the programming arena and applied to everyday life.

INTRODUCTION

We all have to use a variety of different skills in our daily interactions. Our programming skills, whilst being essential to doing our job, only make up part of what we do during the day. There are many other abilities we have that we can use inside, or outside of our day to day jobs and these may vary from person to person, or job to job. It may be that we have certain skills but have never given them much thought, or are even unaware that they maybe a skill. For example, have you ever thought about how you do a certain task that you are good at? Maybe how you do it is natural and you do not give it any thought at all, or maybe have intellectualised your thought processes and are able to describe it all in detail. If you've never thought about this before, then I'd encourage you to give this some thought as you read through the paper. The "skills" explored below are written from my experiences and reflections and are not intended to describe how things should be done. Instead, these have been written to provide examples of how work-based skills could be translated into the outside world and these may enable the reader to also reflect on how they perform similar tasks or actions. The order in which the following information has been presented is based purely on how I was wish to present it, rather than its relative importance.

1) PROBLEM SOLVING

> *" … the awesome splendour of the universe is much easier to deal with if you think of it as a series of small chunks."* – *Terry Pratchett*

For me, problem solving is about breaking down a big problem into smaller, more manageable tasks. Each of these smaller tasks can then be solved in turn, or further broken down as required. When performing a coding task, problem solving can be used throughout: from the overall programming flow; debugging, right down to determining what is the correct syntax for a particular function / procedure you may seldom use. The programming flow could be worked out by sketching different ideas onto paper and seeing very quickly how everything could fit together. The syntax issues could be found from text books, on-line resources, or by asking colleagues. For me, a task only becomes a problem when:

- I do not know where to start, in order to complete the task
- There are many ways to complete the task but I am not sure which one is best
- I don't have what I think I need in order to complete the task

Let's take a look at some strategies we can use to solve problems regarding the creation of a piece of code and then look at how this could be applied more widely.

UNDERSTAND THE PROBLEM

This sounds simple but how many times have you seen someone go straight into coding, without working out what they have actually been asked to do, when they don't understand the problem? Before starting to solve the problem, do you have a clearly defined goal, or outcome? If a set of specifications, or other type of written request exists then the outcome is clearly defined. If the task is of a more exploratory nature, then the goal or outcome may be less obvious. In these cases, it could be worth asking the person who gave you the task what question they are trying answer. Although I don't like the phrase "what does success look like?", here it is absolutely perfect, as by answering this question, you will know when the task is complete.

WHAT RESOURCES ARE AVAILABLE AT THE START

When we understand the problem, we know where we need to end but where do we start? A good place to start would be to work out what inputs / information we need to start coding. Once these have been established we may find that they are not in the expected format and we need to do more work to get them to where we expect. However, we may find that much of the work has already been done for us, in which case the problem may not be as complex as first thought.

STEP BY STEP

We now know the start point and we know the end point. The next part is to work out the bits in the middle. If we filled in the gaps starting from the beginning, there is a danger of getting bogged down in lots of detail and losing sight of the overall direction. Instead, if we know the ideal scenarios along the way, these can be added in to our plan and then we can work out how to connect everything. A good analogy would be navigating a journey using a map and you only know the start and end points. If you have had time to plan your journey you might pick up places of interest that you would like to visit. The order that you pick the places of interest does not matter but the order that you visit them probably does. How you decide to get from place to place is up to you - it could be the fastest, most efficient way, or it could be the way that is easiest, or least dangerous. If you start a journey with only a start and end point, then it is much more likely that you'll get distracted by a place of interest (and this is absolutely fine if you are a tourist on holiday). If we think of our programming flow like this journey, then: the start point is our inputs; the destination is our finished product; the places of interest are the sections of our code that we propose to create. How we get from place to place may be straightforward based upon our knowledge and experience. If this is not obvious, then we need to break down a section further within our planning.

NOT JUST A PIECE OF CODE

The last 3 sections have related to coding, but they could also apply to the whole study or project. It is just that you now have more components / potential problems to deal with. I find that this way of approaching problems works well both inside and outside of work. In addition to the steps described, I also like to think of how the problem could be solved with unlimited resources (e.g. time, money, people) and this can stop a solution being limited by what we think is available to us. Whilst we may not be able to get all of the resources required, we may be able to get more than we initially thought. This wouldn't happen if we only considered what we currently have.

WHEN ALL ELSE FAILS, ASK

Sometimes it is hard to keep perspective when you are in the middle of a problem and it can be difficult to see a way out. Fortunately, there is help. If you work in a team, then a fresh pair of eyes can occasionally see things you miss. Even if they can't help with your direct problem, they may be able to offer a different strategy, which may be simpler than your original one. Often, once I start talking through a problem with someone, I suddenly realise the solution myself. If there is not a real person to ask, then why not try finding a solution through a search engine. The chances are that someone has already been through the same thing, asked the question and posted a resolution online.

TAKING A BREAK

Quite often when I am not working, I come up with the solution to a work-related problem, or vice versa. If it is becoming too much then take a break, have a walk, or do something to get your mind off of the issue. Most of my thoughts regarding the writing this paper did not come when I sat down at the computer specifically to write. For example, most of this short section was written in my head when I was at home, taking out the rubbish for recycling.

2) COPING WITH CHANGE

> *"Life is a series of natural and spontaneous changes. Don't resist them; that only creates sorrow. Let reality be reality. Let things flow naturally forward in whatever way they like."*
> *- Lao Tzu*[1]

Whilst programmers like the certainty that comes within coding, they must also be able to cope with an ever-changing world. When working in an IT environment, there are so many things that can change. Here are some of the things that you may be familiar with:

WHAT YOU DO

Coding tasks are ideally given to us in the form of a written specification, which forms the foundation of what we are about to do. At any point during the task, these instructions could be changed for a number of different reasons e.g. the requestor realises a mistake; the requestor changes their mind; there is new project information. New information may completely change the direction of a project. This could mean your task is no longer needed, or your task may become even more important and be under increased scrutiny, leading to even more changes.

HOW YOU DO YOUR JOB

New versions of languages may be available for you to upgrade to. This can be a really good change in that the

latest update provides new functionality that makes your life much easier. If you are lucky, then all of those annoying bugs you've had to deal with for so long suddenly go away. One of the downsides to a new language version may be backwards compatibility with codes from previous versions and possibly any third-party packages you are using. Outside of your programming language, any other part of the computer / IT equipment you use is fair game for change. Operating systems may be upgraded on a company level and it can take some time to work out where things have moved to. Operating systems can update on their own, for no apparent reason and stop you doing anything productive while this happens. Applications also come, go and get upgraded. Usually by the time I become familiar with one application, getting to a proficient level of efficiency, the application gets replaced. My learning cycle then starts again until the subsequent upgrading / retiring of that application. Hardware also gets upgraded / replaced, or may even break, forcing a change if the piece of kit you were currently using is no longer available.

THE LOCATION OF YOUR JOB

There may be a number of reasons for having to work in different locations from time to time. If you work for a large company, then you may be asked to "hot desk". Hot desking means that a given desk space is not the realm of a single person but can be used by multiple employees during different time periods. This can actually be a very positive change, as you may get to interact with people you wouldn't normally interact with. On the downside, it can take some time to readjust your chair and rearrange your things in the morning before you are able to start doing anything productive. I do like to keep my desk for as long as possible, to save making the morning adjustments but changing my desk occasionally helps to give me a new perspective and stops things becoming boring.

WHO ASKED YOU TO DO IT

When working on large projects, it is common for teams to change; this is normal. With a change of people can come a change of ideas. This is especially true if the new people are senior and have something to prove. I have witnessed this a number of times and sometimes the change of direction is exactly what is needed for the team but if the changes brought about are severe, then it can also completely disrupt the team.

TIMELINES

In my experience of working on clinical studies, the thing most likely to change is the timelines. Project plans get arranged far in advance of the work needing to be completed. As the timeline approaches, we get to see how accurate those assumptions really are. Even if all of the assumptions are correct and everything appears to be on track, there will inevitably be unforeseen events which can cause delays to our part of the process. The greater the number of steps in the data getting to us from the source, the greater the likelihood of delays occurring. One way I try to overcome this is to ask for more time to complete the tasks than I actually need. I do this to absorb any downstream delays and still deliver to the agreed deadline.

CHANGE IS EVERYWHERE

With all of the change we get exposed to in the work environment, it is a wonder we are able to get anything productive done at all. But we do. This goes to show how resilient we can be to change. Combine this with the changes we see in the outside world and the changes in our own personal lives, then I'm sure you will agree that we do get exposed to a huge amount of change. Where does the resilience come from? I don't know for sure but I'm certainly more resilient to change if I: try and put the changes into perspective; use breaks to help "mentally reset"; or talk to colleagues.

FIGHT, OR UNDERSTAND?

Depending upon the type and change and where it has come from, you may decide that it is too much and you need to do something. This could be that you see the change being detrimental to the department / group, or the change may be too much for you to take on. In this case you should speak up against it (make sure your argument has been though out first!). If the change is really big and I have been unable to have my concerns heard, then I try to get onto a working group for that initiative. By doing this, even if I have not agreed with what's happening, I can better understand the reasons for the change and maybe affect it slightly.

CHANGE IN ACTION

Alongside the working environment, competing in sports are a good way to see a lot of the above aspects in action. You may experience: a change in venue; an ad-hoc change of rules; equipment breaking; injuries of team members requiring a change in the team; a change in game strategy based upon how the team / opponent is playing. The next time you are playing a game, or competing, see how you deal with these changes. Do you take the changes in your stride and change strategy accordingly, or do you get overwhelmed? Maybe this is an unfair comparison, as change is certainly expected in sports and you would enter a competition prepared.

3) RECEIVING AND PROVIDING FEEDBACK

> *"Criticism is something we can avoid easily by saying nothing, doing nothing and being nothing" - Aristotle*

Feedback is extremely important as it can help us to improve what we do and it can also help to motivate us to continue to perform well. Throughout our entire working day, we are subject to feedback, which can come in many different ways. Here are some examples of different types of feedback and how they are delivered to us:

FEEDBACK FROM MANAGER / PROJECT LEAD

Generally, this type of feedback would be given verbally and often happens in a meeting environment. Most likely it would be to do with your approach to a study / project. This type of feedback should be given diplomatically and often feels more like being given advice, rather than feedback. At its best and if provided well, you could come out of the meeting with a completely different strategy and be absolutely certain that this new idea came from you, rather than your manager or lead. However, don't be afraid to challenge the feedback if you think it is appropriate to do so.

FEEDBACK FROM COLLEAGUES

An example of this would be when a colleague checks your work as part of a Quality Control (QC) process. Generally, this would be in a written format (unless you have completely misunderstood the task at hand where they might ask you to make another attempt at the work, before they start the QC again). This type of feedback should state the facts plainly and without emotion. If appropriate, your colleague may suggest an alternative approach to the one you have used.

FEEDBACK FROM MACHINES

As a programmer, you will be very familiar with this type of feedback. As code is being developed or tested, it is virtually inevitable that you will receive some sort of message relating to a syntax error, a warning, or the fact that a required resource cannot be found. Admittedly, some of these messages may be the result of your defensive programming strategy but for the sake of argument, let's still keep them in this category.

HOW DO YOU FEEL ABOUT THE FEEDBACK?

The first type of feedback is a lot easier to deal with than the second. At worst, the example of QC comments can make people very defensive and so should be handled in an agreeable way. The last type of feedback can feel more aggressive and can lead to you feeling frustrated - have you ever wanted to throw your laptop off the top of a high building? How many times have you not agreed with the computer that something you are doing is wrong and then proceed to repeat the operation a few times more? If you nodded knowingly to the last point, then you may find it amusing that Einstein defined insanity as doing the same thing over and over again and expecting different results.

FEEDBACK OUTSIDE OF WORK

Outside of the workplace, we are subject to similar types of feedback: the verbal feedback may come from family / friends; written feedback from mail (electronic or real) / social media; feedback from machines could come from an even wider variety of places, as technology is used almost everywhere - even the microwave lets me know when I have pressed the wrong button. In addition to those examples, you may notice that feedback is also widely provided on busy roads and virtually anywhere else where there are more people than space. Hopefully the feedback you observe in these situations has not been directed at you.

HOW WOULD I FEEL ABOUT THE FEEDBACK?

As well as receiving feedback, we may also have to provide it. Examples of this could be: annual feedback of colleagues, QC of work, email correspondence with external business partners, etc. If you are working in developing functions, macros, scripts or applications that are used by other people, you will have to write the messages relating to error handling. In all of the above situations, I find it good to put myself on the other side of the situation e.g. if I was to receive the feedback, email, or error handling message, how would I feel? If you've ever seen someone take feedback badly, then you'll know how damaging this can be, especially if they take it personally.

DIPLOMACY IN ACTION

Providing feedback can very often be an exercise in diplomacy. You may have a number of useful phrases at your disposal to help guide a colleague when you see their idea has a low chance of success e.g. "have you thought about ….?", or "here's an approach that worked for another project team ….". If the person is still intent on following the current course but you can see it ending in disaster, then you can always ask them to follow their logic, or work out how to deal with certain situations: "what would happen if….?", "how would you deal with….?", "can you see any weaknesses with this approach? ". This can be a good way of doing things, as it will allow them to come to the conclusion by themselves and they may even think that they came up with the idea in the first place (as may have been done to us – see "FEEBACK FROM MANAGER / PROJECT LEAD"). As part of my life outside of work, I help to assess students in their karate grading examinations. During the examination we make an assessment and then, based upon certain criteria the student will either pass, or they will fail. The result is given to the student face to face and it is much easier to tell someone that they have passed, rather than tell them that they have failed. Failing students is not a pleasant thing to do and I've seen many approaches to do this. The most common I've seen this done is by the examiner telling the student in an apologetic way "I'm sorry but you have failed". One of the best ways I have seen this done is: "I'm sorry but we have been unable to pass you, on this occasion". This phrase is much kinder the first one.

4) DESENSITISATION

From my time in Data Management and Statistical Programming, I have cleaned, processed and reported a large number of adverse events from many different study types and phases. At first, I was exposed to many terms that were new to me and I didn't know the meaning of them. I used to look these new terms up in a medical dictionary and try to understand what they meant. Being adverse events, these terms were nearly always something unpleasant. When I took the time to think about that event occurring to an actual person, it did make me feel sad (but also very lucky to be fit and healthy). The terms that affected me the most were from the paediatric studies. Over the years, I learned to disassociate the term from the actual meaning. In effect the term just becomes a word and I was reporting these terms as words and not thinking their meaning. This allowed me to compartmentalise what was happening in terms of a task. I still react to some terms but usually these are the more extreme ones. In effect, I have been desensitised to written terms and view them as only this: written terms.

OBJECTIVELY, NOT EMOTIONALLY

So, what use is this? Well, if I spent time looking at every adverse event and understanding what it meant, it would make the job harder emotionally and slow down my ability to get the work completed. I find I can now process the information objectively by placing myself outside of the meaning of the terms. On speaking to those people I know who work within the emergency services, they tell me something similar. In effect, they are able to remove themselves from the situation, in order to process what is happening. I'm in no way saying that my job is similar to the daily experience of the emergency services. What is interesting is that the information is processed in a similar way.

SOMETIMES THE EMOTION WINS

Whilst I try to be objective as much as possible, I know this cannot always be the case. There have been certain situations when me and my colleagues have seen, or been involved in traumatic events. If the work situation mirrors these experiences, then the compartmentalisation may be impossible. One example of this could be to do with a severe illness that you or a family member is going through and you have been asked to work on a clinical study which deals with this. Whilst the emotion can affect your ability to do your job properly, it may also be used to self-motivate. For example, when I am working on a study which is showing a great benefit to patients I try and think of this during times of stress on the project. Visualising the difference that I am helping to create helps motivates me to carry on, even when the pressure is high and the energy levels are low.

THEY WANT YOUR ATTENTION

I find that outside of work, I use this processing or filtering ability anywhere there is an abundance attention grabbing text. A good example is social media, where a great deal of content is placed online, in some cases purely to get a reaction. There is also plenty of additional content from other types of media and these can be overly sensational. If I see something written which makes me react but has text only, I try and only believe it when I see something else to back it up, or there are complimentary images.

5) COMMUNICATION

> *"The limits to our languages are the boundaries to our worlds" - adapted from a quote by Ludwig Wittgenstein*

Traditionally, the programmer is seen as someone who prefers to tap instructions into the computer, rather than spend time interacting with the outside world. This may still be true in some cases but most of the programmers I know have to be very good at communicating, not just with the computer but also to real people at different levels and with different backgrounds. We may be asked to present our work in front of colleagues, or to provide updates at project team meetings. We may not all find communication easy but it can be an essential part of our job.

WRITTEN COMMUNICATION

Written communication happens in a variety of ways and is key to the programmer performing their job efficiently. You could argue that writing the script itself is communicating with the computer and if this is not correct, the program will not run correctly (see "FEEDBACK FROM MACHINES"). Within the code, we have opportunities for written communication that can help understanding the script. For example, a well written program header will provide the reason or purpose for the code, as well as things the user may need to be aware of when running it. Within the code itself there should hopefully be some comments and these help sections to be easily identifiable, or explain why things have been done a certain way. Personally, I like to add lots of comments but this is mainly to help my memory when I get asked to make updates to the code at a later date. Outside of coding, we may be asked to create specification documents and these need to be good enough for someone to code from, without us being there to answer questions. Other forms of written communication could include communicating with colleagues, or stakeholders on email, or instant messaging.

CHECK WHAT HAS BEEN WRITTEN BEFORE IT IS SENT

Working for a multi-national company has really helped how I view my written communication. When writing an email, or instant message I find that it is good to check it first. When checking through it, I try and put myself in the position of being a non-native English speaker: are my sentences too long? Do I put too many things into a single sentence? Could I say things in a simpler way? Is what I am writing too wordy and benefit from removing some words (without losing the meaning)? If after a read-through, what I have written makes no sense to me, it is very unlikely it will make sense to anyone else. This helps my written communication become more concise and hopefully easier to understand. Also, my brain tends to work faster than I can type and this tends to mean that mistakes can occur. There have been a number of times when reading through an email, I've come across a "have", or "do" and missed out the proceeding "not". Reading through emails before sending them is also a really good idea, especially if you are writing an email under stress, or angry. It is not recommended that you write the email under either condition but sometimes we have no choice (I'd suggest if you have to do this then keep the "to" field blank whilst making the initial draft). In this case, the very nature of writing what you feel can help to calm you down before rewriting and removing anything which you think may cause offense. Re-reading written communication before posting on social media is also essential, as the real world can be a very unforgiving place.

COMMUNICATION WITH OTHER STAKEHOLDERS

A lot of the time at work we use words, phrases, or acronyms that mean nothing to anyone who works outside of our own small sphere of expertise. If we look at our project team, which is made up of other individuals from different parts of our company, we may find that this group has its own terminology too. Newcomers may have to quickly grasp the language used by the team, if they are to have any hope of understanding what is going on (let alone take the minutes of the meeting). Even within the team, if someone from one discipline uses terms that are unfamiliar to the other team members, this can cause confusion and require backtracking to explain what has just been said. It could be that an acronym from one department used in a meeting has a completely different meaning to members of other departments within the team. Have you seen any of these things happen? Are you guilty of using your own group's terms but you know that not everyone will understand what you mean? I am. Where possible I try and use non-technical words and keep things as simple as possible but it is a hard balance to achieve. Throughout this paper I have talked about a "project lead", or a "project meeting", hopefully this choice of terminology is fairly understandable. If I had used one of the many 3-4 letter acronyms I've seen used at different companies for different project lead / meeting types, then maybe you might not have initially understood me. Buzzwords can also create a layer of vagueness that needs to be cut through in order to get to the underlying meaning (if any). Many of these buzzword terms or phrases could be very easily replaced with simple words, where the meaning is easy to understand by everyone. How often have you sat in a meeting where someone has talked or presented for a number of minutes, using lots of buzzwords which have blinded you to what they are trying to say? Buzzwords do seem to be contagious, as they are very easy to pick up and use. We'll never be able to stop the spread of this but we can try and limit our own usage. If you do notice yourself using these terms, why not try saying what you actually mean instead?

MEETING PEOPLE HELPS UNDERSTAND THE WRITTEN COMMUNICATION

Sometimes we can interpret written communication (such as an email) in ways that were not intended. For me, I think this mostly happens when I do not know the person who is sending the email. When I know the person who is sending the email, I can visualise them speaking to me and understand why certain words, phrases, or humour are used. This is much harder to do for someone you do not know in person. If I am likely to be working with someone quite frequently on a project, I think that it is a good idea to meet them in person. If they work in the same building as you then there is little excuse not to arrange a meet-up. If they are not in your building, country, or time-zone then a video conference would be the next best thing. This helps greatly when having to communicate over email, or in meetings, as you will have a better understanding of what that person means when they write or say something.

JARGON OUTSIDE OF WORK

Jargon does not just appear within work; it also appears outside of work too. If you've ever taken a car to the garage, called out a plumber, applied for any sort of financial service, or ordered a coffee, the chances are you will have heard terms that are not standard (I tend to just ask for a normal coffee, at the standard size). It may be that you see someone who is an expert communicator and has realised you are not from their world, easing you gently into their ways and sayings. You may have someone who is quite new and doesn't know how to translate from their technical language to the lay person. You may be very unlucky and find that someone is using words to deliberately deceive you. How often do you go along with something where you do not understand what is being said to you, in case you offend them? If terms are being used and you need to understand them in order to make informed decisions that will affect you, it makes sense to ask that these are clarified before you proceed.

6) COPING WITH PRESSURE

For me, stress at work, or at home comes from a number of things:

- Time pressures

- Too many things to do
- Not knowing the best way to perform an urgent task

Whilst we can usually deal with one of these things at a time, more often than not when I am under pressure it is due to more than one of these being applied at the same time. An example of the first 2 points could be that there are a number of different projects all wanting something ready in the next few days. This could be stressful if each different project has a different project lead telling you that their tasks are the most urgent. On top of this you could get an urgent request for a project you are covering for someone else. This request might need you to look into something you were not previously aware of, find where it is located, find the creation program and decipher the programming logic to make edits. This could be an example of the last point.

IS THE DEADLINE MOVABLE?

Just because we have been asked to do something, it doesn't mean that it is important. How many times have you put in the extra hours and given up work on other projects, or given up personal time only to find that the work hasn't been used? What would happen if the deadline was missed? Is it an arbitrary deadline or if it is missed are there consequences for the team, department, or company? Working out which deadlines are true deadlines and which are not can be helpful in managing the workload.

NEGIOTIATE BY GIVE AND TAKE

What if the deadline is immovable but the amount of work is too great to get done in time? Try and find out what work is the most important and prioritise this. After all, your requestor can't look at all of your work at exactly the same time, can they (I know there are lots of exceptions to this but hopefully you get my point)? By delivering work using a step-wise approach you may be able to buy some time and help reduce some of the pressure. A better solution, if practical, would be to negotiate delivering the most important work early and then everything else, in a stepwise way until completion. This way some work is completed and in return you may be allowed to get a deadline extension for some of the later work.

IS THE PRESSURE COMING FROM YOU?

Sometimes I see that when people outwardly showing that they are under pressure, it is because they are putting too much pressure on themselves. I can be guilty of this too. It can feel like there is no way out and this is a major cause of stress. As with problem solving, talking to people is also a good way to help relieve some of the pressure. Whilst you may not be able to solve all of your problems through talking to people, the very act of talking to someone can help your resilience. It can also let people know how you are feeling. If you work in a team and feel completely overwhelmed there may be some tasks that can be taken on by others. If you are leading a team and feel overwhelmed, then maybe you could delegate some work to your team members. Sometimes when I am in the middle of a pressured situation and I can't see a way out, this is a good time to take a walk. As in the "TAKING A BREAK" section, getting your mind off of what is happening can help with mental resilience. Constant pressure is not good for our health or wellbeing. I tend to find that when work gets busy or family pressures pile up, the first thing that tends to be lost is time for hobbies. This is not a good thing, as hobbies provide stress relief by being a much needed outlet and help us become more resilient to everyday pressures.

7) CODING CAN BE FUN

If most of your time at work, or at least a large portion of it is actual coding, then this should be fun. Why spend a lot of your time doing something that you don't enjoy? This doesn't mean that if things aren't always fun then you should be looking for a new opportunity. Everyone's career has peaks and troughs and learning how to deal with the troughs is very important for future development.

WHY DO YOU ENJOY IT, OR NOT ENJOY IT?

If you are enjoying what you do, then have you thought about why this is? Do you enjoy the challenge, is it the people, or do you get given a good variety of tasks to do? Hopefully it is not just about the money. If you are not finding coding fun anymore, then why do you think this is? Is the work not varied enough, or maybe you are not being challenged sufficiently? Maybe the answer has nothing to do with coding but something else entirely. Working out the answer to these questions can help you to make some positive changes to what you do.

CAN YOU PUT THE FUN BACK IN?

Not everything we do in life though is enjoyable. For the things that we have to do that we don't enjoy, it helps to find some way to make the experience more enjoyable. If you are finding the coding not enjoyable anymore, maybe learning some new skills that you could apply to the job could help. Maybe a change in project is what is needed, or it may be as simple as sitting at a new desk for a while (see "THE LOCATION OF YOUR JOB"). If you are having fun then self-motivation is much easier.

8) CONTINGENCY PLANNING

> *"If you take enough precautions, you never need to take precautions."*
> – Terry Pratchett

For me, this is a very important part of problem solving or planning. How many times have you got part of the way into a strategy, only to have something unexpected happen, or go wrong which means that you have to re-evaluate your approach? A contingency plan (also referred to as a "plan b") is put in place for when things do not proceed as expected. By doing this, we have already thought about our alternative course of action upfront and this saves time and hopefully stress, as by the time we need a contingency plan, we may be under all sorts of pressures. Contingency planning can be an essential part of business planning and could include a number of different steps e.g. assessing the risks, implementation of the contingency plan, testing of the contingency plan, improvement of the contingency plan. I tend to not go this far in my day to day activities but instead think more of the risk and then what would I do in that particular situation, if something goes wrong.

WHAT CAN GO WRONG?

My way of approaching contingency planning involves looking into what could go wrong and then working out what options are available. As programmers, we should be pretty good at thinking about what could go wrong, as we routinely include sections in our code for data checking and error handling, etc. This encourages us to think about what things can go wrong in our input objects and what a user can do to stop our script running in the intended way. Once our code has picked up that something has not turned out as expected it may provide a helpful message to the user about why the script has stopped. It might also say what to do next, or it may just stop the rest of the program from running, if the unexpected issue is critical to its successful completion. If we did the same amount of contingency planning (or more accurately, working out what can go wrong) in real / working life as we do in coding, then we may appear a little paranoid. If we thought of everything that could go wrong and made a plan for it, we'd not get very much done at all!

PLANNING IS SIMILAR TO CONDITIONAL CODING

I like to think of contingency planning as a way to not stop my journey down a given path. It becomes a different branch, or choice of branches to take if the first-choice path is not an option. In this respect it is very much like conditional coding. For example, we may say IF everything is as we think, THEN let's do our first choice ELSE IF our first choice doesn't work THEN let's use our contingency plan, OTHERWISE if our contingency plan isn't an option either, let's ask for help. Feel free to substitute the capitalised text for syntax from your language of choice.

SURVING THE RUSH HOUR

One really good real-world example of this is travel of any type. This certainly becomes more important when time pressures are included. Personally, I don't like driving and find that getting to and from work can be a little stressful, especially when everyone else is trying the same thing. Most mornings I use my tried and tested route to work, which although not the most direct route is the quickest route based upon the average conditions. Traffic closures and roadworks can cause unexpected delays and queuing, which is not really where I want to be. Having some knowledge of the local area I have a number of alternative routes which I can divert to in case something unexpected happens. Due to having thought about this beforehand, I can change my plan without having to think too much and this helps keep my stress down as much as possible. The most sensible thing though would be to have a satnav with active traffic information on it (like my wife has) but then that wouldn't have made such a good example.

9) IT'S OK TO MAKE MISTAKES

> *"If we knew what we were doing, it would not really be called research, would it?"*
> *- Albert Einstein*

For any task you are given, there is a possibility that you may make a mistake. Factors affecting this could be: our experience; the amount of time we have to do the task; the amount of other work we also need to get out at the same time. Processes can be employed to catch mistakes before the task is completed but it is not always possible to catch everything (have you ever used a piece of software that contained a bug?). There are some places where it is acceptable to make mistakes and some places where it is not. This section intends to show that in certain situations, it is ok to make mistakes and in fact this is a natural part of the learning process.

ACCEPTABLE MISTAKES

During working for different companies I have seen that they accept mistakes up to a certain level. Once that level has been reached, action is taken but up until this point the mistake is deemed acceptable. Whilst working in Data Management for a large Pharma company, we had to compare the information on the paper CRFs (yes, they used to be hand-written on paper) and compare this to what the Data Entry person had input into the database. Although there were processes to minimise the amount of data entry errors, they could never be fully eliminated. When

running the comparison, we would randomly select a number of CRFs and check every page. At the end of the checking, we would determine the error rate and if this was less than 1 error in 1000 data points, it was deemed acceptable. Another example came from my time working in a small shop. The shop realised that mistakes would be made and that these could apply to money taken through the till and also accidental damage to the stock. Like the first example, if everything fell under acceptable limits then no action was taken. We may see something similar in our QC processes. For example, we may notice mistakes in the code, or outputs that do not exactly match what was provided in the requirements. During the course of our QC we will determine the magnitude of this mistake and if it is has no impact to the final outcome, it could be deemed acceptable.

A PLACE TO MAKE MISTAKES

Whilst we are likely to make mistakes when we are doing something new, we are given places to make these mistakes, which are insulated from the outside world. When coding, we would have some sort of development environment which allows us to create, test and debug our programs before they are let loose in the "main" environment and handed on to our customers. New applications, or processes are also created and tested within working groups of the company. These groups are responsible for the successful implementation of the application, or process and may have to go through many iterations before something robust is ready.

BLAME CULTURE

I've witnessed work departments and families that look to determine the blame as soon as a mistake is identified. From what I've seen, this tends to encourage mistakes to become hidden and the culture of blame also leads to a defensive mind-set, where people are more interested in covering themselves than in doing the work. I find that this leads to an unhealthy environment and certainly not one that is pleasant to work in. Sometimes, covering mistakes can take more work than owning up in the first place. A good story relating to this is about Tom Watson Jr, CEO of IBM[2]. A young executive at the company had made some mistakes costing the company money. When the young executive went to the office and expected to be fired, Tom Watson replied: "Not at all young man, we have just spent a couple of million dollars educating you."

CUTTING IT FINE

If I try anything new, I am pretty certain that I won't get it right the first time. In these cases, I try and test out what I am doing on something that doesn't matter, before carrying out the originally intended job. An example of this would be to do with DIY (or home improvements). If I get a new tool, I'll try it out on a spare bit of metal, wood, or plastic to see how it reacts. I'll learn the best ways to use the new tool on something unimportant and make my mistakes here. Once I am confident, I'll attempt the actual job.

MAKING MISTAKES TO PUSH THE LIMITS

We have to make mistakes as part of the learning process. In fact, if they are not encouraged, how do you know where your limits are? I've known karate squad training to include turning drills where the students are encouraged to turn on the spot as fast as they can, until they go so fast they lose balance. This exercise encourages the limits to be found and increased but can only be achieved by them making a mistake (losing balance), learning from it and correcting it. The limit can then be increased over time. This would not be possible without allowing the students to make the mistake in the first place.

10) YOUR OWN EXAMPLE

> *"Experience is the best teacher" - Latin proverb*

Up to this point, the paper has been about my thoughts and experiences. This section is about you. Hopefully throughout reading this paper, you have been thinking about your own experiences, what you have learned and how you deal with certain situations on a day to day basis. You may completely agree with everything written here. You may have found something new here that you really like and would like to try it out for yourself. Most probably though, you will have thought of different ways you do things to the ways presented here. You may even disagree with my choices for the previous 9 "Useful Skills You May Have Learnt Through Being a Programmer" (there were more I could have chosen but these were the ones I wanted to talk about). I'd like you to now think of your own example, what is it that you have learnt through programming that you use in everyday life that has not mentioned here? You may not have given it much thought up until now, or it might be your secret weapon in dealing with all manner of stresses inside and outside of work. If you would like to share your own example, then I'd be happy to receive this (see "CONTACT INFORMATION" below), or if you want to keep this to yourself, then that is absolutely fine.

LAST THOUGHTS

Hopefully you will agree with me that the programmer gets exposed to many different things and needs a wide variety of skills to deal with them. By now you may have added your own thoughts and examples to those written here. You

may even see that the general concepts behind what has been presented so far applies not just to programmers but could also be adapted for other industries too.
Although the paper is written as though these skills are learnt inside the workplace and then applied outside of work, in reality this may not be true. It may be more likely that the skill was actually learnt outside of work in the first place. For the sake of simplicity, I have concentrated on writing the paper as though the skill is developed at work and then applied outside of work. I hope this has not caused you any confusion whilst reading my thoughts.

My very last thought on this came to me nearing the completion of this work. Although I had not intended for this to happen, I had noticed that possibly all of the 9 defined points above had been used in the writing of this paper. By now, you could probably guess how these were applied but here they are anyway:

1. **Problem Solving** – My end point was that a paper had to be finalised within a given timeframe. My start point were ideas for "skills". I then broke the paper down into the main sections to get the overall structure. Each section was then added to with notes or bullet points for key information I wanted to include. For anything I wasn't sure of, or had to research further, placeholders were created so that I could come back to those at a later date. Finally, the detail was added in, after I was happy with the overall feel.
2. **Feedback** (Receiving) - a number of review rounds allowed me to obtain constructive feedback in order to improve the content and flow.
3. **Coping with Change** – the review brought about comments for improvement, or even huge changes to sections. I was expecting to have to change the aspects of the paper and so this was not a surprise. In just about every case the proposed change made sense and I think it made the paper better.
4. **Making Mistakes** – this is why the paper was sent for review!
5. **Contingency Planning** – in case of extreme feedback during review round, I had backup "skills" in case anything I had written needed replacing. Whilst writing, the file was backed up and hard copies printed at various times. This would allow work to continue in case of a technical failure, or being away from a computer (for presenting, I usually carry a backup copy of the slides on a memory stick + a hard copy)
6. **Make it Fun** – I have chosen to write about something that interests me, this has helped a lot in enjoying the writing process. In writing the paper I have also tried to see the humour in the different aspects mentioned but without trying to make the paper funny.
7. **Coping with Pressure** – the main pressure from this task comes in the form of timelines. I set myself generous timelines for completion of key tasks in order to cope with any additional work / family pressures that have cropped up.
8. **Communication** – hopefully, if you have got this far then I have done a successful job in communicating my thoughts. I've tried to minimise the amount of jargon and sayings. Please use the contact information below, if you disagree.
9. **Desensitisation** – I can't see any obvious examples of this in writing the paper, though if I identified any then I wouldn't have been desensitised.

REFERENCES

[1] Although the quote is generally attributed to Lao Tzu, it does not appear in the Tao te Ching
[2] https://the-happy-manager.com/articles/characteristic-of-leadership/ - there are many versions of this story.
[3] https://www.iainabernethy.co.uk/content/10-things-martial-arts-should-have-taught-you-about-life-podcast

ACKNOWLEDGMENTS

- My wife Jess for inspiring me with the initial idea for the paper, after helping create a skills based CV for a complete change of career
- Iain Abernethy's podcast "10 things the martial arts should have taught you about life"[3] for providing me with the structure for the paper
- Sensei Gavin Paul for his experience and examples
- Vincent Buchheit, Tim Barnett, Carolyn Fisher and Paul Dick

Thank you also to all of the people that have provided me with their thoughts, experiences and ideas in writing this paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Paul Grimsey
Roche Products Ltd
6 Falcon Way, Shire Park
Welwyn Garden City / AL7 1TW
+44 (0) 1707 36 5871
paul.grimsey@roche.com