From Novice to Expert – Understanding the Learning Journey

Robert Goatham – Calleam Consulting Ltd

As a writer on the subject of project failure it's impossible to avoid being drawn back to the issue of training. Effective training is a critical element in most human endeavours and the failure to provide appropriate training is a contributing factor in the majority of the failures I review. Although the vast majority of the people who participate in these failures have the raw intelligence needed to succeed, a lack of effective training stifles the ability of the organization to leverage that intelligence into an effective project delivery capability.

Few would argue that expertise is a critical component in the success of any project. A team with the required expertise is far more likely to succeed than a team that lacks expertise. Despite the obvious correlation, the nature of expertise and the processes by which expertise develops are issues that are poorly understood by most organizations. As a result, many organizations struggle to develop the advanced skills needed to successfully deliver today's complex projects.

Perhaps because expertise is hard to quantify, in practical situations (such as when hiring) many organizations simply equate expertise to the number of year's experience a person has or their level of training. Despite this fairly simplistic perspective, when challenged most people recognize that expertise is something more than that. While experience and training are the raw materials from which expertise is formed, there is a further process by which experience and knowledge are distilled into the invisible commodity we call expertise. That step is often poorly understood and in many practical situations individuals are left to manage that portion of the learning process on their own. Some people have the natural ability to make that leap by themselves, but others find it more difficult to do. The net result is that many organizations are left with a wide variance in the capabilities of their team and studies commonly report a ten to one variance in the performance (in terms of both productivity and quality) of individuals performing the same role within the same organization [1, 2].

As well as being a significant source of problems for organizations, the difference in individual capabilities raises some important questions. How does expertise develop? Are the differences between individuals purely inherent or could changes in the way people are trained and managed bring everyone up to the same level (or at least closer to the same level)? Beyond experience and training what other elements affect a person's ability to develop expertise?

Expertise Defined

The starting point for answering those types of questions lies in understanding what expertise is. Although it's a commonly used expression, few are able to give a detailed breakdown of what it really means. Dr Gary Klein is a world leader on the issue. Klein specializes in the field of naturalistic decision making (making decisions in real world conditions rather than in pre-conceived experiments) and the processes by which individuals become experts. Klein's findings show that an expert is a person who has, within their domain of expertise [3];

1) High levels of situational awareness and the ability to see the big picture
2) The ability to distill patterns out of complex events and the ability to recognize those patterns when faced with similar situations in the future

3) A detailed understanding of the inner workings that drive events and an understanding of the complex, entwined relationships between cause and effect
4) The ability to envisage the chain of events that led to a situation and the ability to forecast what will happen in the future
5) The ability to see events that did not happen (that might have been expected) and other violations of expectancies
6) The capacity to see differences between situations that are too small for a novice to detect
7) The capacity to see opportunities and improvisations that would not occur to a novice
8) The ability to generate many ideas for how to solve a problem or approach a situation

Collectively, these types of abilities are referred to as "higher order thinking skills" [4]. Higher order thinking skills are the skills that support critical thinking, analysis and problem solving and they represent the high end of the learning food chain.

The Stages of Learning

A well known model for understanding the progression of learning that leads to the development of higher order thinking skills was developed by Benjamin Bloom in 1956 [4] and subsequently updated by a team led by Anderson and Krathwohl in 2001 [5]. The model is commonly referred to as Bloom's taxonomy and although it was originally intended as a classification system to help educators establish learning objectives, the model also provides a useful representation of the different levels passed through as a person develops a skill. The model can be summarized as follows;

1. Remembering – Memorization of basic facts, terminology and concepts
2. Understanding – The extraction of meaning from the basic knowledge
3. Applying – The ability to see how to apply learned material to real world situations
4. Analyzing – Being able to see internal structures, components and cause and effect
5. Evaluating – Being able to make subjective value assessments and judgments
6. Creating – The ability to create new ideas or strategies based on observations and insights

Watching children learn to play soccer is a good illustration of what happens as a learner passes through the different stages. Young children start by learning the game's key elements and basic rules. Once the child has gained that basic understanding they start applying their knowledge by playing a game.

At this point children have only been exposed to the taxonomy's first two levels and anyone who's watched young children play soccer for the first time will have seen a raw example of what can happen when learning is restricted to that level. Rather than being a game of passing the ball between carefully positioned players who gain strategic advantage by reading ahead of the game, you get a single pack of children all enthusiastically running after the ball at the same time.

As players mature the process of developing the higher order thinking skills needed to play soccer at a more advanced level begins. In level four (analyzing) young players begin to understand the game's strategies and structurally why different players are assigned different positions on the pitch. As players develop still further their ability to see and evaluate the constantly changing dynamics of a game allows them to better judge where they need to be on the pitch in order to anticipate where the ball may be heading (level 5 - evaluating) and finally, players start to form their own ideas about strategy and what to do next (level 6 - creating). By age eight you can already hear team members yelling at other players what they ought to be doing!

Levels 4 through 6 are an ongoing iterative process that involves the synthesis of both training and experience. The more practice a person has and the better the quality of the training they receive from their coach, the more soccer players are able to develop their skills.

A Template for Learning

Although Bloom's taxonomy provides a picture of the stages through which skills are developed, it does not explain the process by which skills development occurs. Although the psychological process of learning remains an area of research, Klein's work provides some practical insights on the issue. By studying highly skilled workers in their natural environments and the process through which they developed their skill, Klein has identified a set of conditions needed in order for higher level thinking skills to develop. While the conditions themselves don't explain the inner workings of the brain, they do hint at the processes by which learning occurs. The conditions Klein identified are [3];

1. Repetitive exposure to common patterns of events
2. Timely feedback between a decision being made and the decision's outcome becoming manifest
3. A clear relationship between an individual decision and its outcome
4. Time to reflect on performance

Again learning to play soccer provides a good illustration of these processes in action. Much of the coaching young soccer players receive is done using repetitive drills that provide the student with exposure to the repeated patterns of events that may occur in a real game. The coach provides timely feedback on performance and because each exercise focuses on an individual skill there is a clear relationship between the actions the learner took and the outcomes. Most teams also participate in local leagues that allow the skills to be used in real world situations. The score at the end of the 90 minute game provides clear feedback and most coaches are more than willing to provide feedback as the game progresses! To cap things off, many games are followed by a debrief that allows the individuals to reflect on their performance.

Although not every young player will go on to bend a ball around a defensive wall the way David Beckham can, the average standard of play for those who have been through a proper coaching program is far higher than for groups who have not. I learned to play soccer without proper coaching and certainly the standard in the junior leagues in the area I live in is far higher than the level my friends and I managed to attain kicking a ball around on our own.

The Training Gap

Obviously, training "knowledge workers" is a far more complex challenge than teaching someone how to play soccer. In IT projects, roles such as Project Manager, Business Analyst, Architect and Developer involve significant levels of complexity and subjectivity that go beyond the confines of the soccer pitch or the relatively simple set of rules by which soccer is played. Despite the additional challenge, the levels defined by Bloom and the conditions identified by Klein still represent the basic architecture around which knowledge workers develop higher order thinking skills.

As with learning to play soccer, the starting point lies in attaining basic understanding and in the technology sector most introductory training naturally focuses on the lower levels of Bloom's taxonomy. After a certain level of experience has been achieved many people move on to complete one or more of the currently popular certification programs. While some of these programs attempt to delve into the higher order thinking skills, in practice by nature of the way the learning is structured (text book learning, self study, multiple choice exams, etc) they are often unable to make serious advances in this area.

While the traditional training and certification programs provide knowledge workers with the knowledge and understanding levels of Bloom's taxonomy, the higher order thinking skills represent a considerable challenge. IT projects are in many ways the antithesis of Klein's four conditions for developing higher order thinking skills. The inherent character of a technology project is such that;

1) They often extend over considerable periods of time, thereby diluting the immediacy of the feedback (even a few days or weeks can be enough for dilution to occur)
2) Extended project durations mean that team members may have only experienced a handful of prior projects, thereby reducing the level of repetitive exposure
3) Projects are all unique and generally involve high levels of complexity. The combination of factors mean that the identification of common patterns is often a difficult task
4) The large number of individual decisions made in a project , the high number of interactions between those decisions and the complex cause and effect chains arising from the interactions between the decisions increases the difficulty of tying outcomes back to individual decisions
5) A project's final outcomes are often abstract in nature and far removed from the individuals who worked on the project, again reducing the effectiveness of the feedback loop

In addition, few organizations have in place effective coaching programs that are able to provide the ongoing support and feedback necessary to allow higher order thinking skills to develop in an orderly fashion. The net result is that in many organizations the process of developing higher order thinking skills is achieved through the school of hard knocks. That's a long painful process and not only is it costly for the organization, the cost on the personal level can also be extreme. I personally know of several people who have been placed on medical leave by their doctors because of the highly stressful situations that arise as projects get into trouble.

A Case Study in Advanced Learning

So do the factors outlined above mean that developing higher order thinking skills within the context of a project based environment is an unachievable goal? Clearly not, many people have navigated the minefield on their own and some organizations have achieved more wide-ranging success. The pressing question is how do we broaden those successes so that they become attainable to a larger audience?

The answer to that question lies in a number of elements and perhaps the easiest way to explore the question is to contrast examples of efforts that have worked with some that have failed. The starting point for such as discussion comes from looking at the way organizations provide formal training. Although much of the traditional training targets the lower levels of Bloom's taxonomy carefully structured programs can be developed that directly focus on the higher order thinking skills.

I observed one good example in an organization I worked with many years ago. The department in question had a team of approximately 30 developers who were responsible for the bespoke development of software to support a large multinational commercial business. The developers had a range of backgrounds (some had computer science degrees and others had moved into the field after working in other capacities throughout the organization). Experience levels ranged from as little as 3 months up to 15 years and included a large contingent of newly hired junior developers.

Over the years the organization had provided the developers with initial training that taught them the syntax of the programming language they were using, but beyond that that had been no additional training in how to structure code effectively, how to make code less bug prone, strategies for simplifying the programs, how to write code that was maintainable or strategies for promoting code reuse. Interestingly, even those who had been through formal computer science degrees had never been trained in such considerations either.

In essence the developers had been through levels 1 to 3 of Bloom's taxonomy, but had never had any formal guidance in the higher levels. As a result (and as is typical in many other roles in the IT sector) each individual had developed their own thinking on issues such as what the difference between good and bad practices were. The results were as you might predict; productivity and quality variances between individuals were high, teams were having difficulty sticking to project schedules due to the wide ranging variances and because of continual quality problems, wastage from rework was high.

To tackle the problems the technical manager responsible for the department commissioned the development of an in-house training program directly aimed at improving the quality of the code the team produced. The training was based on the book "Code Complete" by Steve McConnell [6] (a much recommended read for those doing development related work) which lays out the techniques, strategies and practices used to create high quality code. As such, the book directly addresses the higher order thinking skills needed in order to become an "expert" developer.

The training material was created by translating the strategies in the book directly into code fragments that were meaningful within the context of the organization. Each key idea from the book was

communicated as part of the training and real code from the organization's code base was used to demonstrate how code could be structured in different ways to improve readability, reduce the amount of code needed in order to achieve a given result and improve overall quality. The restructured code fragments (as well as the steps take to do the restructuring) were built into a set of four, half day workshops that were conducted over the space of a few months.

The program is interesting because the organization maintained a set of metrics that allowed the effect of the training to be measured;

1. Comparing the 1 year period prior to the training with the 1 year period following training, the amount of effort the department spent fixing code defects that had made it through to production environment dropped by 52% (and even then, much of the effort spent fixing defects in the year after training went on problems that predated the training)
2. Productivity improved by 30% (measured in average source lines of code written per person)
3. In addition, two specific projects (one completed prior to training and one done after training) were compared in detail (both contained approximately the same amount of code and were of similar functional complexity). The first project suffered 8 production failures in the 6 months after production release while the second had no such failures
4. The percentage of newly created code found to be in overly complex routines (measured using McCabe's Cyclomatic complexity measure) dropped from 36% to less than 3%. In plain English the code was simpler and that in turn would imply reduced maintenance costs in subsequent years. The simpler code is also likely a key contributor to the drop in defect rates

In part the program was successful because it directly contrasted good and bad practice. Those contrasts made people think about the advantages or disadvantages of different approaches (the analysing and evaluating levels of Bloom's taxonomy).

Beginner's Mind

Although it's tempting to attach all of the credit for the improvements to the training program itself, there is a more complex story behind the story. Understanding the deeper story casts more light on other processes that came into play.

As you might expect, in general, people who have been doing a job for 15 years are less open to learning new ways of doing things than those who have 3 months of experience. Certainly the organization found that some people were more open to the training than others. On the one side were those with the least experience and those who had a natural openness to new ideas. At the opposite end, a few of the more experienced staff had the attitude that there really was nothing more for them to learn and so why participate. Although most still attended the workshops, one person flatly rejected the training and declared vociferously "it's my way or the highway".

Although the "my way or the highway" attitude is fortunately in the minority it represents the apex of a larger problem organizations have in developing higher order thinking skills. The underlying problem is one of structuring the organization and its training methods so that learning is an ongoing activity rather than something that only happens in the period immediately after a person moves into a new role.

The problem was summarized succinctly by the Zen priest Shunryu Suzuki who said "*in the beginner's mind there are many possibilities, in the expert's mind there are few*" [7]. Although Suzuki used the word "expert" in a different way from Klein (in Klein's definition "real experts" are able to generate lots of options to solve a problem), the point Suzuki makes underscores one of the great challenges organizations have in developing higher order thinking skills.

Suzuki's insight was based on the observation that when we begin learning something we approach it very differently than we do once we feel we have reached proficiency. In Zen thinking, the "beginner's mind" represents the state of being open to learning that occurs when we first start learning something new. Because we're not weighed down by prior experiences or preconceived notions, we're open to new ideas and looking at things from different angles. As we become proficient at something (or more to the point, as we judge ourselves to have become proficient) we transition to a more closed mode of thinking in which we are less observant and less receptive to new ideas.

Klein's work has actually identified a set of cognitive processes that may, in part, lie behind this phenomenon. Klein found that the mechanism used by beginners to make decisions differs from that used by those who have more experience. The predominant decision making mode used by beginners is a type called "mental simulation". In mental simulation the brain makes careful observations of the environment, thinks of ideas for how to respond and envisages (simulates) how a given course of action will turn out. Based on the different options and the internal simulation of their possible outcomes, the brain selects what it determines to be the best option. Mental simulation involves thinking (sometimes conscious and sometimes semi-conscious) and certainly when beginning to learn a new skill, the mental simulation mode makes us more open to suggestions and guidance from external parties who can help us identify options and evaluate how a given course of action might turn out.

As our level of experience increases Klein found that mental simulation decision making gets relegated to situations the brains deems to be atypical. For more familiar situations, the brain naturally switches to a faster method of decision making called "recognition primed decision making" (RPD). RPD is the subconscious process behind the intuitive thinking we use for the vast majority of our day to day decisions. Using RPD, the brain bypasses the relatively time consuming observation and simulation steps used in mental simulation and instead relies on rapid recognition and the brain's stored patterns of past experiences to decide how to respond.

Learning to drive is a good illustration of the two processes at work. As a first time driver, we have to concentrate on every individual decision it takes to operate the vehicle, but as we become more accomplished we no longer need to concentrate as hard. In fact it's not uncommon to arrive somewhere without being explicitly conscious of the details of how we got there (such as what traffic lights we had to stop at or what every other vehicle around us was doing). Ask a person who has just completed their first driving lesson. Because the beginner uses the conscious mind more than the expert, new drivers are often highly aware of those types of detail.

The problem for organizations is that in situations where a person has received insufficient guidance through the upper levels of Bloom's taxonomy the transition to RPD style decision making can occur too early. RPD decision making is at its most effective when it's backed up by a sufficiently rich set of internalized mental patterns and a sufficiently rich mental model that links the patterns together. Although a person can use RPD without having attained the advanced insights of a true expert, the lack

of underlying understanding increases the chances of suboptimal decision making, higher errors rates and all the associated problems that come with those mistakes.

Once the transition from the beginner state to the experienced state has been made many people find it difficult to reopen the door to learning. My wife is only too painfully aware of this. Although she makes lots of suggestions for how I could improve my driving, I tend not to listen. The old curmudgeon with the "my way or the highway" type attitude represents the height of the problem.

Clearly one of the big challenges for organization is keeping the door to learning open. Only by doing so are organizations able to work towards a team in which expertise is epitomized by Klein type experts (able to generate lots of options for how to respond to a situation) rather than the less informed type noted by Suzuki (there is only one way to do things and it's my way or the highway).

Indirect Paths to Learning

Returning to our example of the training program that sought to develop the skills of the programming team, within a year of the program completing everyone involved (including the curmudgeon who refused to participate) had measurably improved their abilities. Given that some people weren't interested and one person didn't even participate, that's a surprising result. It's an observation however that hints at some of the indirect learning mechanisms that occur within an organization.

Although formal training is the front lines of education, learning is something that happens in other ways as well. Under the right circumstances we can learn from each other (peer to peer), we can learn from role models and we can learn from those who provide us leadership. In our subject organization, likely all three processes came into play. Understanding the sequence in which changes occurred within the development team indicates the internal workings of some of those processes.

Not surprisingly the first group to actively adopt the improved set of practices were those who were the most junior. Having the openness of the "beginner's mind" they lacked the baggage that takes hold when we perceive ourselves to be proficient at something. That openness was also demonstrated by the fact that it was this group who were most likely to seek out additional one-on-one coaching from the senior developer who was leading the program.

The second group to make noticeable changes was the team leaders who were overseeing the more junior resources. Although this group did not solicit additional one-on-one coaching they became exposed to more real life examples of high quality code because they were responsible for reviewing the work of the junior resources who reported to them. By being exposed to more examples of well written code and possibly through something of a peer pressure mechanism, significant improvements in code quality started to take hold within this group as well.

Not unexpectedly, the final group to change were those who were the least interested and in particular the curmudgeon who was directly hostile to the idea of change. Doing minor maintenance work rather than project related work, several of these individuals worked in isolation from the rest of the team and hence were not responsible for reviewing the work of the junior developers. Despite their lack of direct exposure, the complete department was working on the same application so again even these diehards were able to see examples of the better code in the shared libraries and hence were able to start picking up some of the ideas through osmosis.

One of the underlying elements in the sequence of events outlined above comes down to building up a body of samples. They say a picture is worth a thousand words, but as a corollary I would say that a sample is worth two thousand and a set of good examples is worth ten thousand more. When a person is surrounded by good examples, it gives them a frame of reference against which to work. Although the workshops were the starting point, the program really took hold once a sufficiently diverse set of examples became available. From that point forward the improved practices became institutionalized and a part of the group's everyday practice.

Role Models

Beyond the physical examples that contributed to the success of the training program, to some participants (primarily the junior staff), the consultant who ran the program became a role model as well. Role models can be one of the most powerful learning tools an organization can have. Many individuals who attain the highest levels of achievement in their field credit role models as having inspired them. Coaches training young sports teams leverage that affect by using instructional videos created by leading professionals. The videos provide examples of how to do things right and provide role models towards which the younger players can aspire.

Not only do role models provide learners with a frame of reference, they also provide a benchmark against which expectations can be set. This is true for both the performance expectations set by the organization's management team as well as the expectations we set of ourselves. When a person aspires to a role model, the role model provides a benchmark against which that person judges their own proficiency. Because role models typically have higher performance, the level at which we deem ourselves to have attained proficiency often gets set at a higher level than it would in the absence of the role model. By setting the bar at a higher level, the door to the beginner's mind is left open for a longer period of time. That additional time and the presence of the role model provides the context within which higher order thinking skills are better able to take hold. As an example, consider the development of a newly graduated engineer. If they joined a team in which the most senior person only had 3 months of experience, the chances are that they would advance to the level at which they judged themselves to be proficient much sooner than they would if they worked alongside the team of experts that put man on the moon.

In retrospect, my very first professional job demonstrated the point. I started as a junior programmer and shortly after I joined the organization was raided by a competitor who was rapidly expanding. Because the competitor was offering more money, many of the more experienced staffs were lured away. The net result was that a year after completing basic training, I and the other trainees who joined with me were not far off being the most experienced in the department. Despite the team's sincerest efforts the results were predictable. Much of the work we produced had problems and I remember one particularly grim week where six of the seven pieces of work we released into the production system had to be deactivated immediately after release because they were disrupting business operations.

Following that incident, the manager responsible for the department recognized the problem and arranged for some experienced staff from another department to shore up our capabilities. In part because the door to the beginner's mind had already started to shut, the newly assigned technical leader had significant difficulty convincing some that different practices needed to be adopted in order

to avoid future problems. Although you can't re-live history, I suspect that had strong role models been present throughout the formative periods, the problems we had would have been greatly reduced.

Many organizations may feel that such problems don't apply to them because they already have role models in place. Certainly most organizations do have people who are doing commendable work that could be used as a reference point. The problem is that simply having capable people is not enough. Role models only contribute to organizational learning in situations where the role model's work products and decisions are clearly visible. If you have a capable person but no one ever sees the direct outcome of their work, they are unlikely to be of value in terms of training.

Leadership

As much as it was a lack of role models, the problem my team had in that first organization was a lack of leadership. In addition to supervising our team, our manager was also working on a task force charged with establishing a commercial partnership with another organization. Task force activities dominated his time and as a result, not only did we lack role models, but for much of the time we lacked any leadership at all.

The net result was that in a management capacity he had little time to do anything other than assign work to the team and then firefight when things went wrong. One wise manager I know calls that the DAFT approach (Delegate and Firefight). The word "daft" is an English colloquialism meaning dumb or stupid. The manager who coined the expression did so after watching several of his colleagues struggle for years. His point was that the fires within an organization are ignited by weakness in the capabilities of the team. Because the DAFT approach lacks an educational component that could break the cycle, managers using the DAFT approach are trapped in a continual cycle of failure.

In contrast to the DAFT approach, some leaders view education as a part of the management role. One manager I worked with typified the thinking. He would hold regular lunch and learns and frequently circulated observations or lessons learned to the team. In addition, rather than waiting for annual performance appraisals he would provide ongoing feedback. Although not everyone read everything that came round, the manager felt that keeping the learning process alive was part of his job. The resulting actions the manager took created a learning environment that was reflected in the behaviours across the team.

Where managers take less interest in the learning process, even the value of a well structured training program can be lost. Following the success of the programming skills development training outlined above, the Vice President overseeing the department decided that the program should be rolled out to two other development groups. Each group assigned a coordinator to run the program within their department. Although the new groups had similar makeup to the original group and the coordinators followed the same steps, the program failed to achieve any measurable changes. In large part those failures came down to differences in the leadership styles in use in the different groups. Having commissioned the program, the manager overseeing the first group was a strong supporter of the program. The managers in the subsequent groups showed little interest. In part the "not invented here" dynamic may also have played a role in the subsequent failures.

Coaching

Leaders who do perform a hands-on educational role are essentially providing the same service as a coach. The input they provide spreads the learning experience into the upper levels of Bloom's taxonomy and the advice they give provides the immediacy of feedback that is critical in the development of Klein type expertise. In addition, a leader's position of authority can help prop the door to the beginner's mind open thereby allowing richer and deeper levels of understanding to form.

Most organizations recognize the potential value coaching can bring and rather than having managers act as coaches they establish peer-to-peer programs. In many cases such an approach is necessary because the managers themselves lack the technical skills in the domain in which they are working (this is especially true where technology has changed very quickly). While establishing such a program is a valuable step in the right direction, in many of the examples I see there are fundamental problems with the approach being taken. Those problems often undermine the value the program has to offer.

Chief among the problems is the fact that coaching is usually an add-on duty that gets piled on top of an already busy work load. Busy fighting the fires that are inherent because of weak fundamentals within the organizations, in many cases the coaches don't have time to provide adequate guidance to their trainees. As a quick and dirty metric I often ask those enrolled in these programs how much direct coaching time they have received. Across the organizations I visit, the average answer is a lowly 90 minutes, which is indicative of the challenges organizations have in establishing effective programs. Interestingly, that figure is usually a revelation to executives who assume that coaching is a central pillar of their training system.

Despite the difficulty I certainly urge organizations to try putting a coaching program in place. That advice however comes with the counsel that organizations need to do so with their eyes open and realise that coaching programs take resources, thoughtful planning and ongoing oversight if they are to succeed.

Higher, Higher Order Thinking Skills

Teaching someone the finer points of programming practice represents a significantly easier task than teaching some of the other skills used in delivering a technology project. While programming practices can be clearly captured and discussed based on physical examples of code, skills such as Project Management involve subjective issues that are less visible and less easily measured.

That difficulty is reflected in much of the current training for Project Managers. A significant portion of the commonly available project management training focuses on the relatively easy to train tools, techniques and methodologies of project management (the so called science of project management) rather than the more advanced topics and soft skills (the art of project management). Even advanced training often does little to bridge the gap and some so called advanced training does little more than delve into tools, techniques and methodologies in a deeper way than introductory training is able to do. In many cases that advanced training is of no value because the lack of "art" type training prevents the organization reaching a level where advanced tools, techniques and methodologies would be of use.

In part the focus on the lower levels of Bloom's taxonomy has been because of the difficulty codifying more advanced layers of Project Management knowledge. How do we express such information in a

way that it can be shared in a format suitable for training? While sports teams can set up realistic drills that emulate real play and can video complete games for detailed analysis, such methods are less viable in the context of something like Project Management.

The good news is that new tools are beginning to take hold that can help organizations address that problem. One particularly useful tool that is being adopted in a number of fields is the use of pattern languages. The concept of a pattern language was first proposed by Christopher Alexander, Professor emeritus Berkeley University, California [8]. His work in architectural design led him to the idea that successful designs often used repeated patterns. By understanding and documenting these patterns they can be used as a training vehicle to help others develop skills more quickly than if they had to experience the trial and error of discovering the patterns themselves. It's interesting to note that the identification of patterns is a key component of Klein's definition of expertise, so it's easy to see how the use of patterns as a learning tool can help bridge the gap to higher order thinking skills.

Pattern theory and the documentation of patterns have become mainstream ideas in a number of fields. The most well known is the design of IT architectures. The growth of the field and the value it brings has seen the concept applied not just to design, but to other areas as well (such as patterns of behavior, patterns of collaboration and decision making patterns).

To complement patterns of behaviors or events that lead to success, some groups are now also exploring so called "anti-patterns". Anti-patterns represent the patterns of destructive behavior or common problems that lead to project failure. The DAFT model itself could be considered an anti-pattern n the management domain and common on-line repositories such as Wikipedia have more extensive lists.

The field of identifying patterns that apply to Project Management is relatively new, however a body of knowledge is beginning to develop. Through the study of both successful and failed projects clear patterns can be distilled. The use of these patterns provides a platform from which individuals can be trained and also a vehicle to allow participants to reflect on their own experiences and isolate out the sequences of decisions and events that lead to certain outcomes (be they positive or negative).

A second field of study that holds great value is the field of systems dynamics. Systems dynamics is the study of the cause and effect relationships and feedback loops that exist within complex systems. Thanks to the interaction between people, time, budget and other factors, a project can be considered a complex system. By modeling the factors that affect project outcomes, a dynamic picture of the mechanics that drive events in a project can be built. Again such models directly support Klein's definition of expertise. By allowing us to understand the inner workings of a project and the chains of events that lead to certain outcomes, dynamic models can play a significant role in helping develop the insights needed to improve levels of expertise.

Patterns and dynamic models can be used in combination to help build simulations of a project. Such simulations allow hands on participation and experimentation that again helps reinforce messages. The combined use of simulations, patterns and dynamic models directly supports the natural processes by which people develop expertise and as such offers a realistic option for organizations in their efforts to deepen the levels of Project Management expertise within their ranks.

Conclusion

The project environment's natural lack of feedback mechanisms and the challenge in codifying more advanced layers of knowledge have for a long time been core problems that have prevented many organizations from developing higher order thinking skills on a consistent basis. The resulting gaps in the organization's skills development infrastructure exposes the organization to the rough and tumble school of hard knocks and as many organizations can attest, that's a schoolyard that is littered with the wreckage of failed and troubled projects.

Although different organizations suffer from the problem to different degrees, most of the managers I speak to recognize the problem (even if some will only do so in confidence). The inability to bring everyone to a consistent level results in many organizations developing what I call "A" team dependency. The "A" team represent the few whose innate ability has allowed them to develop the required higher order thinking skills on their own. Because the "A" team players are more skilled and more productive, management frequently places a disproportionate percentage of the work burden on their shoulders. While that strategy can achieve short term goals, for many reasons it's a poor strategy in the long run.

The high number of troubled projects in the IT sector and the number of organizations who are affected lead me to refer to the problem as a chronic crisis. While no disrespect is intended to those who are currently providing training (and there are a valiant few who are working in the realm of higher order thinking skills), I think as an industry we need to look carefully at the methods we use for developing skills in a project based environment. That requires us to codify more of our advanced knowledge, find ways to share that knowledge more effectively, improve the feedback mechanisms we use today and find ways to keep the door to the beginner's mind open (preferably forever).

I believe the raw ingredients are there to make this work and I'm certainly a believer that most of the people who get involved in the IT sector have the raw intellect needed to succeed. Rather than being a problem at the individual level, the problem comes down to the industry's inability to impart skills in a consistent way. It is only by looking carefully at the way learning is conducted that we will be able to overcome these problems.

While some organizations may shy away from the visible costs associated with providing more effective training, those costs need to be seen in perspective. The cost of failed project provides that perspective and while some organizations see failed projects as part of the cost of doing business, I would say there are two types of business cost. Failed projects are the cost of doing "dumb" business. Establishing an effective framework for learning on the other hand is part of the cost of doing "smart" business. Sadly, if any one would like one, I can provide a long list of organizations that have demonstrated just how high the cost of doing "dumb" business can be.

R. Goatham – Principal Calleam Consulting Ltd.
Robert@calleam.com
www.calleam.com

About Calleam Consulting: Calleam Consulting provides training and consulting services to the technology sector. Calleam specializes in the comparative analysis of both successful and failed projects. Using simulation, modeling and analysis of both successful and failed projects from the past,

Calleam helps organizations turn yesterday's hindsight into the foresight needed for tomorrow. For more information visit www.calleam.com

References

[1] Software Cost Estimation with Cocomo II – B. Boehm, et al – Addison Wesley, 2000
[2] Peopleware (productive projects and teams) 2nd edition – T. DeMarco, T. Lister – Dorset House Publishing, 1999
[3] Sources of Power (How people make decisions) – G. Klein – MIT Press, 1998
[4] Taxonomy of Educational Objectives: The Classification of Educational Goals; pp. 201-207; B. S. Bloom (Ed.) Susan Fauer Company, Inc. 1956
[5] A Taxonomy for Learning, Teaching, and Assessing — A Revision of Bloom's Taxonomy of Educational Objectives; Lorin W. Anderson, David R. Krathwohl, Peter W. Airasian, Kathleen A. Cruikshank, Richard E. Mayer, Paul R. Pintrich, James Raths and Merlin C. Wittrock (Eds.) – Addison Wesley Longman, Inc. 2001
[6] Code Complete – S. McConnell – Microsoft Press, 1993
[7] Zen Mind - Beginner's Mind – S. Suzuki – Weatherhill (New Ed edition), 1973
[8] Notes on the Synthesis of Form – C. Alexander – Harvard University Press, 1964