

National College of Ireland
MSc Learning Technologies
2005/2006

Answering Challenges Enhances Learning

Investigation of the performance improvement of
software developers resulting from the deployment of the
Inner Workings Developer™ practice-based learning
system

Clement McGann

03234614

clement.mcgann@icsmember.ie

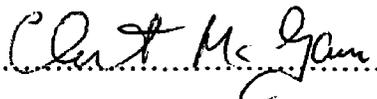
August 2006



National
College of
Ireland

*The college for a
learning society*

I hereby certify that this material, which I now submit for assessment of the programme of study leading to the award of Master of Science in Learning Technologies is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: 

Clement McGann

Date: August 11, 2006.

Student Number: 03234614

Table of Contents

1	Abstract	5
2	Introduction	6
2.1	Background	6
2.2	Overview	6
2.3	Approach	7
3	Literature Review.....	8
3.1	Software Quality.....	9
3.2	There are problems with the quality of software.	12
3.3	How to Attain Quality	19
3.4	Education as a solution.....	27
3.5	Reflection	33
4	Method.....	39
4.1	Discussion on Methods	39
4.2	Field Observation	39
4.3	Field Study	40
5	Field Observation	41
5.1	Relevance of the field observation	41
5.2	Audience.....	41
5.3	Description of the Course	42
5.4	How typical was the course?.....	43
5.5	Structure of the Course	44
5.6	Quiz.....	44
5.7	Motivation.....	45
5.8	Feedback	46
5.9	Conclusions from the field observation	47
5.10	Other Observations from the Field Observation.....	48
6	InnerWorkings Developer™.....	51
6.1	Field Observation	51
6.2	What is InnerWorkings Developer™?	51
6.3	InnerWorkings Developer™ in Operation.....	52
6.4	InnerWorkings Developer Learning Issues.....	53

6.5	Claims of InnerWorkings Developer™	54
7	Field Study	55
7.1	Purpose of the field study	55
7.2	Participants in the field study	55
7.3	Organisation of the Field Study participants	56
7.4	How valid is this field study? Or: How qualified are the participants?	58
7.5	Was learning enhanced?	65
7.6	Field Study Conclusion	75
8	Future Perspectives	75
9	References	76
10	Appendices	85
A.	Answers to Field Observation	86
B.	Interim report on the field study from the TSB group	89
C.	Answers to Field Study Questionnaire	91
D	Field Study Responses	109
E	The ISO 9126 Standard – Definition of Quality	111

1 Abstract

This paper considers the current deficiencies in software quality. Software quality is defined, with reference to the ISO 9126 standard. The effect of the deficiencies is illustrated. Some of the reasons are explained.

Current approaches to address the issue of software quality are given, with special reference to sound engineering approaches such as the Capability Maturity Model. Quality Assurance reduces the effect of lower quality software. The role of standards and the use of Fagan's inspection method are mentioned. Opinions differ as to whether the quality goal is achievable.

Current solutions are based in the engineering tradition. There are alternative approaches to software writing espoused by Richard Gabriel who speaks of the artistic nature of programming. He calls for the empowerment of the 'mob'; that is the programmers

Education in the Computer Sciences is neglected. To succeed it needs to address the higher cognitive levels. Reflective learning is discussed, the need to transform knowledge from 'knowing' to 'doing'; from 'declarative' to 'procedural'; to actually solve problems and write code.

Means of how to motivate students are discussed; in particular how students respond to challenges.

The field observation is of a programming class in industry. Some aspects of the course are reviewed. The hypothesis that 'answering challenges enhances learning' is formed.

The field study tests this hypothesis while trialling InnerWorkings Developer™. The views of the volunteers confirmed the hypothesis and were positive to the benefits of InnerWorkings Developer™.

2 Introduction

2.1 Background

The motivation for this study stems from concern for the quality of computer software. This quality is disappointing. There have been and continue to be efforts to address this issue. Although remarkable achievements are often claimed, the problem persists. Many of these panaceas focus on the management of and the practice of software production. Although substantial attention has been paid to educational issues, this paper asserts that its potential is underestimated, and it is therefore under-resourced.

Specific motivation for this study arose in the context of a programming course, in industry, and considerations on how it could be improved. This is discussed in 'field observation' on page 41.

This study was enabled by the availability of a specific educational product InnerWorkings Developer; this is described on page 51. The study is described in 'Field Study' on page 55.

2.2 Overview

This paper considers software quality to be important; it considers it to be lacking. Various approaches to address the issue of software quality are surveyed.

In considering education, as one of the solutions, the issue of reflective learning is raised. Reflective learning, in this context, means transforming learning from knowledge of facts to the ability to apply that knowledge; moving learning from 'knowing' to 'doing'; from 'declarative knowledge' to 'procedural knowledge'.

There are various mechanisms for stimulating this transformation. It is proposed that 'answering challenges', actually writing code in reply to a quiz question, is such a mechanism.

2.3 Approach

A novice programming class in industry is considered. From this 'field observation' the hypothesis 'answering challenges enhances learning' and subsidiary hypothesises were formed. To confirm these hypothesises a 'field study' was undertaken.

InnerWorkings Developer™ is a learning tool which relies on issuing challenges. A group of experienced programmers, in industry, used this product. The 'field study' is an analysis of their experience using InnerWorkings Developer™. Based on their reports, this paper concludes that 'answering challenges enhances learning'.

3 Literature Review

The Literature review discusses software quality. It considers that software quality is an important issue. There is a definition of software quality. Then the current shortcomings in software quality are illustrated with some 'horror stories'. Some reasons for these defects are mentioned

Having determined that software quality is important, but lacking, current steps to rectify the situation are outlined. Standards are discussed along with their difficulties and contradictions. Accepting that errors are inevitable, we alleviate the situation by identifying and reducing errors. Quality Assurance is discussed. Nonetheless quality, the objective to 'get it right – first time', is a laudable aspiration. Ways to achieve it using methodologies and education are introduced.

This paper then focuses on education, its deficiencies and advantages; in particular reflective learning is discussed. This is in preparation for the research question; the hypothesis that 'answering challenges enhances learning'.

Following the literature review, this paper continues with a 'field observation', an observation of a programming class in industry, which introduces the hypothesis. Then this paper concludes with the 'field study' to confirm the hypothesis.

3.1 Software Quality

3.1.1 Software Quality is an important issue.

In a ceremony at the White House on Monday, 14th March 2005, President Bush presented the National Medal of Technology to one of the leading visionaries on software quality, Watts Humphrey, Fellow of the Software Engineering Institute of Carnegie Mellon University, for applying the principles of engineering and science to software development. Humphrey and Carnegie Mellon's Software Engineering Institute are trying to lead the industry toward defect-free software

The National Medal of Technology is the highest honour awarded by the president for *“technology innovators and rewards contributions to the nation's economic, environmental, and social well-being”* (Bush, 2005). Bush said: *“Your work is making our country more competitive, more hopeful, and more prosperous.”*

In a survey, published in Information Week of 300 business-technology managers, 49% specified ‘improving software quality’ as priority. (Information Week 2005)

This paper, while lauding Humphries’s application of engineering principals, doubts that defect-free will be achieved using this ‘engineering’ approach, advocates also promoting the craft of programming, in particular moving programming education from the theoretical to the practical.

3.1.2 What is “software quality”?

Just what do we mean by “Software Quality”? One’s first thought is that it is software which is free of error, if, indeed, such exists.

However there are other necessary attributes, such as the ability to change or adapt and the economic imperatives

Change is inevitable; *“To live is to change, and to be perfect is to have changed often”* (Newman, 1845) therefore the code must be maintainable. But change can introduce error.

Economic imperatives cannot be ignored. They require the software to be written efficiently and to execute efficiently. So we would add the cost of software production and its performance to our list. But compromising to these necessary requirements, can introduce error.

Our list of what constitutes 'quality software' can grow to the extent that it is self-defeating. It can reach the stage where 'Worse is Better' (Gabriel, 1990). This work challenges many of our ideas on quality by forcing an acknowledgement that the adoption of some quality attributes results in the denial of others.

There are two issues with 'Quality'; first: a definition is required; second: we need a measurement. There are inadequate answers to both questions. That is not to say that these answers do not exist, but that these answers continue to be refined. The ISO 9126 standard, discussed below, and described in the Appendix, on page 91, provides both a definition and a measurement of software quality

3.1.3 Software Quality: A definition

What is Software Quality? An early, oft quoted, definition, identified these attributes: (McCall, 1977)

- Correctness
- Reliability
- Efficiency
- Integrity
- Usability
- Maintainability
- Flexibility
- Testability
- Portability
- Reusability
- Interoperability

3.1.3.1 ISO 9126 Standard

The ISO 9126 Standard has a definition of quality, entitled 'Software Engineering – Product Quality'. In addition to its 'quality model', ISO 9126 propose three metrics for measuring quality: External, Internal and 'in use' metrics (ISO).

The ISO 9126 definition is summarised in the Appendix, on page 91.

Shortly after ISO 9126 was published, it attracted comment and amendments were being proposed. Other definitions were proposed. Most of the ISO 9126 definitions do not lend themselves to direct measurement. Several are, of their nature, more subjective than objective.

Other standards were proposed, such as the IEEE Standard 1061-1998 'IEEE Standard for a Software Quality Metrics Methodology' (IEEE, 1993)

The efforts of many contributed to the 'Eagles' report (Eagles 1996). This did propose extensions to the ISO standard

3.2 There are problems with the quality of software.

It has become something of a catch phrase to “*blame the computer*” for failures. It was back in 1978 that the Farmers’ Almanac famously declared: “*To err is human, but to really foul things up requires a computer.*” Computer systems do come into their fair share of criticism. We may seek to reject such criticism as unfair, but perhaps we need to be more realistic and consider the facts. One of the differences between computer programmers and other trades is that others can more easily find excuses for their errors, while an error in programming, once the system fails, leaves little room for denial. There is an old saying in journalism to the effect that: “*Doctors bury their mistakes, lawyers jail their mistakes and journalists publish their mistakes for all the world to see*” (Richards 2002). What of programmers? They say: “*It’s not a bug it’s a feature!*” (Lubar 1995), or as Rich Kulawiec said: “*Any sufficiently advanced bug is indistinguishable from a feature*”

It is said: “*If debugging is the art of removing bugs, then programming must be the art of inserting them.*” Ultimately we cannot escape responsibility for faulty code. “*It is a painful thing, to look at your own trouble and know that you yourself and no one else has made it*” (Sophocles, 440BC). It is, therefore no surprise to read that 51% IT workers cite job stress as a problem, which is ten percentage points higher than the overall figures. (Sosbe 2006)

3.2.1 Horror stories

The purpose of this section is to illustrate, with extreme examples the defects in software quality.

3.2.1.1 PPARS

A recent local 'horror story' was the PPARS payroll system for the health services. In April 2005 their 'National Test Centre' was celebrating that "... *there were no problems only solutions ...*" (PPARS 2005). We now know that in June, the then CEO of St James's Hospital, wrote "...*the Hospital is now not willing to continue with an arrangement which clearly threatens its basic functioning,*" (Kenny, 2005) In August, the Irish Medical Times reported PPARS as a disaster, which cost €231 million and could cost €500 million (IMT 2005) The political fallout continues. Many excuses have been proffered for this error. However we are left with the story of a payroll system, which failed to accurately calculate wages. The Midland Health Board carried out a test on a sample number of employee pay slips to test the systems accuracy; 43% of the sample had one or more errors on their pay slip. (Kelly 2005) In one notorious incident, a health service employee was overpaid by €1m as part of an electronic funds transfer error. And of course we don't know about the overpayments that were not reported.

3.2.1.2 FISP

The PPARS fiasco was quickly followed by FISP, the central financial management system for the Health Service, which wasted €30 million (Reid 2005). A further three million Euro was spent on a health portal, which failed to be delivered (Irish Health 2005). All that was delivered is an empty web-site at www.fisp.ie.

3.2.1.3 Budget overruns

Pulse, the Garda record-keeping system was unable to scale up from its pilot implementation and cost an extra €20 million, overrunning its budget by 50%, to implement.

The Comptroller and Auditor General reported that a new IT system for the Irish Blood Transfusion Service overran its budget by 115%.

A railway signalling system, the Iarnród Éireann Mini-CTC signalling system, overran its budget by 150% (Silicon Republic, 2003)

3.2.1.4 Financial Services

The Irish League of Credit Unions had to write off €34 million when the ISIS project failed. The system was intended to facilitate electronic funds transfer and support ATMs. (Averbuch 2004)

Dr Joe McDonagh, a senior lecturer in business studies at Trinity College Dublin said: *“The €150m lost on the health system — you don’t have to look far in this country to see figures higher than that being lost in failed projects in large financial services firms.”* (Kennedy 2005)

3.2.1.5 Department of Works and Pensions,

The UK Government is no stranger to IT failures – last year a failed IT upgrade paralysed the UK’s Department of Works and Pensions, causing 80,000 civil servants to resort to writing out giro cheques to some 800,000 pensioners. (Lettice 2004)

3.2.1.6 Pentium floating-point error of 1994

Perhaps the most widespread bug was the Pentium floating-point error of 1994 (Lawrence 1994). It was a simple omission. The Pentium computer chip’s division algorithm relied on a table, from which five entries were inadvertently omitted. (Pratt 2005)

3.2.1.7 Therac-25

Therac-25, a computer-controlled radiation therapy machine, had *“several types of serious errors in its design, any one of which would be obvious to an undergraduate in computer science or engineering, led to the deaths of six patients”*. (Bernecky 2004)

3.2.1.8 Electronic Voting

The Irish government spent €52 million on an electronic voting system which cannot be used (Reid 2006). The Commission on Electronic Voting published its findings in July 2006. It reported *“Design weaknesses, including an error in the implementation of the count rules that could compromise the accuracy of an election, have been identified”* (Commission on Electronic Voting 2006). It may seem shocking, but while, most of the time the C-language computer program counted votes cast correctly, in the words of the commission: *“However there were a small number of cases that were counted incorrectly”*.

3.2.1.9 Summary

“Surely there have been enough software project failures to acknowledge the need for Generally Accepted Software Engineering Practices”. (Morgan 2005). This paper will go further and call an improved approach to education.

The objective in identifying these ‘horror stories’ is not to deny that a disciplined approach can identify and address defects, but to illustrate the consequences badly written software.

3.2.2 Reasons for Low Quality

3.2.2.1 Pressure to deliver

Developers are being pressurised to deliver, and deliver quickly, quality is not necessarily something that will be at the forefront of their minds. We all know that doing something quickly does not always equal doing something well. They will test their programs to make sure that they work, but few developers, in such circumstances have time to think about conditions that could lead to their code or functionality failing.

3.2.2.2 Knowledge

Clients seek software to perform specific functions, freeing them for more lucrative pursuits. Sometimes software professionals don't have quite the right skills or background to understand the business requirements or apply the right tools to model and produce the corresponding systems. (Morgan 2005)

3.2.2.3 Data Quality

This paper addresses issues of deficiencies in software quality. That is not to maintain that there are no other issues, such as problems of Data Quality. One of the causes of deficiencies in data quality is poor software quality.

3.2.2.4 The extent of poor quality

Experiences vary, however I would expect to encounter eight errors in every thousand lines of code. However, from the literature, that figure appears conservative.

3.2.2.5 Bugs

Many and various reasons have been and will be advanced to explain these faults. Whatever one says of, for instance: PPARS, we are still left with a system, which fails to calculate wages. We are left with 'bugs'.

Bugs have been with us from the beginning. Thomas Edison coined the term 'bug' in 1889 (Pall Mall Gazette 1889). In 1945 (Hopper 1945) Admiral Grace Hopper introduced the term to computing. (Hopper 1981)

Maurice Wilkes, the creator of the first stored-program computer, discovered debugging in 1949 *"As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs"* (Campbell-Kelly 1998)

In that same year, Alan Turing asked the question “*How can one check a routine in the sense of making sure that it is right?*” He provided an illustrated answer “*In order that the man who checks may not have too difficult a task the programmer should make a number of definite assertions which can be checked individually, and from which the correctness of the whole program easily follows.*” (Turing 1949) Clearly, he considered it possible.

Admittedly, discussing chip design, it was recently observed (Magnusson 2006) that “*It is nearly 60 years later, and debugging embedded software has not changed all that much from how it was done on Wilkes’s EDSAC.*”

Figures from the Software Engineering Institute at Carnegie Mellon University estimate that every 1,000 lines of programming code contains between 100 and 150 errors. That means up to 15% of code contains bugs. (Sarin 2005)

3.2.3 Poor Quality – Summary

There are two approaches to addressing problems of software quality. The first, attempts to ensure that the software is written to a high standard in the first place. The second accepts that errors are inevitable and uses testing and error detection techniques to identify and eliminate these errors, later. These approaches are not mutually exclusive. Indeed, both paths are usually followed. However more emphasises can be placed on one or on the other.

3.2.4 Is Poor Quality Inevitable?

Dr. Fred Brooks has been labelled as pessimist by some, sceptic by others and realist by others. In ‘No Silver Bullet’, he argued that the difficulties are inevitable, arising from software’s inescapable essence. (Brooks 1987) He described this ‘essence’ as “*complexity, conformity, changeability, and invisibility.*”

Dr Brad Cox presented a contrary view when he asked: “*What if there’s a Silver Bullet and the competition gets it first?*” (Cox 1992)

Software quality was poor, in the past. In 1970, Professor A.J.Perlis said *"I think it is inevitable that people program, and will continue to program poorly. Training will not substantially improve matters."* ... *"We have to learn to live with it"*. Software quality is poor, in the present. Under the heading "IT Execs to Vendors: Your Software Stinks", Information weekly reported: *Representing billions of dollars in annual technology spending, IT leaders from British Petroleum, Lockheed Martin, Unilever, and Kaiser Permanente made it clear that the software industry needs a new business model, better quality control, and closer product development ties with customers. "The quality of software I'm getting from you people is abysmal," David Watson, Kaiser's chief technology officer, told an audience of several hundred software industry executives.* (Kontzer 2005)

Most approaches, to date, hold that the application of sound engineering principals will impose order on chaos and rectify this situation. This will be discussed, in the context of the Capability Maturity Model, on page 26. The advocates of engineering solutions call *"Surely there have been enough software project failures to acknowledge the need for Generally Accepted Software Engineering Practices"*. (Morgan 2005).

Edsger Dijkstra, the father of structured programming, posed the real question when he asked whether the programmer is a "Craftsman or Scientist" (Dijkstra 1975). The programmer needs to be both. Most approaches concentrate on the 'scientist', although the term 'engineer' is usually applied. Perhaps we have neglected the craftsman. This theme is now being recognised. Last year, June 11 2005, the ACM recognised Richard Gabriel with the 'Newell Award'. The citation reads: *"For innovations not only on fundamental issues in programming languages and software design but also on the interaction between computer science and other disciplines, notably architecture and poetry"* (ACM, 2005).

3.3 How to Attain Quality

These 'horror stories' illustrate the shortcomings in many computer systems. There are various initiatives to address this issue, such as: Licensing, Standards, Testing, Design Methodologies, Tools, Education and others. These can be broadly classified into two: 'getting it right first time' and 'fixing it later'. While we aspire to the former, we adhere to the latter.

As well as reviewing ways to attain quality, we can consider why we fail to succeed.

There are various approaches to achieve quality. For instance, these and other experiences have led to calls for the licensing engineers. However, that may not be either practical or in the best interest of the industry or the public (Knight & Leveson 2002).

Whether licensed or not, there is a general consensus that standards are required, but do standards, of themselves, improve quality? The use of standards to achieve quality is discussed next. Then 'quality assurance', formally known as 'testing' will be discussed.

3.3.1 Standards of Quality

Although there were differences in emphasises among differing standards, there were not real contradictions. Indeed most depicted their standards as building on prior standards. However there were unresolved internal issues within these standards. Although the Eagles report discusses many of the attributes of quality, it lacks any comparison or priority of these attributes. This void was to lead to the 'rise of worse is better', discussed below.

A considerable body of literature has been written on standards, leading to the question 'Do Standards Improve Quality?' (Schneidewind 1996). Standards, of themselves, do not ensure quality. They provide a means to measure quality and provide the means to detect poor quality.

Standards can measure attributes such as Correctness, Efficiency and Maintainability

3.3.1.1.1 Correctness

Given any legal input, a program is considered **correct** if it produces the desired output. The ability of the program to handle illegal input (by displaying the appropriate error message), or **robustness**, is comparatively less important than its ability to perform the basic functions of the program.

3.3.1.1.2 Efficiency

A program is **efficient** if it performs its tasks without consuming too much processing time and memory space. How much is “too much” depends on the programming question.

3.3.1.1.3 Maintainability

A program is **maintainable** if the code is easily understandable, employing devices such as descriptive variable names, comments, indentation and modular programming.

3.3.1.2 Conflicting objectives

These laudable goals can be in contradiction. For example, to write a program to run most efficiently, it may therefore be more difficult to understand and therefore less maintainable.

The MIT/Stanford style of design attempts to address this by ranking the objectives. (Note that this only addresses issues of design rather than the complete code.) Their objectives are:

- **Simplicity** - the design must be simple, both in implementation and interface. It is more important for the interface to be simple than the implementation.
- **Correctness** - the design must be correct in all observable aspects. Incorrectness is simply not allowed.

- **Consistency** - the design must not be inconsistent. A design is allowed to be slightly less simple and less complete to avoid inconsistency. Consistency is as important as correctness.
- **Completeness** - the design must cover as many important situations as is practical. All reasonably expected cases must be covered. Simplicity is not allowed to overly reduce completeness.

This approach in ranking objectives is useful, as it therefore, resolves the conflicts.

3.3.1.3 “Worse is Better” classification.

Richard Gabriel has described this classification as “Worse is Better”. (Gabriel, 1990)

He extends the definition as:

- **Simplicity**-the design must be simple, both in implementation and interface. It is more important for the implementation to be simple than the interface. Simplicity is the most important consideration in a design.
- **Correctness**-the design must be correct in all observable aspects. It is slightly better to be simple than correct.
- **Consistency**-the design must not be overly inconsistent. Consistency can be sacrificed for simplicity in some cases, but it is better to drop those parts of the design that deal with less common circumstances than to introduce either implementational complexity or inconsistency.

- **Completeness**-the design must cover as many important situations as is practical. All reasonably expected cases should be covered. Completeness can be sacrificed in favour of any other quality. In fact, completeness must be sacrificed whenever implementation simplicity is jeopardized. Consistency can be sacrificed to achieve completeness if simplicity is retained; especially worthless is consistency of interface.

3.3.2 Quality Assurance

Testing, now known as ‘Quality Assurance’, is necessary to avoid poor quality. Most methods of ensuring software quality are variations on Fagan’s ‘software inspection’. (Fagan 1976)

There is a need to test and to test thoroughly. Now we refer to ‘testing’ as ‘quality assurance’. Quality Assurance is now a specialisation in its own right (Jedras 2004). There is an economic limit to testing. The duration of most beta programs is one to three months, although the period may be shorter (Shea 2006). We need, also, to improve the software which is produced.

According to the old adage “*When you are in a hole – stop digging*”. Regrettably in some cases – such as PPARS, mentioned above - Those who identified errors were met with denial, arrogance and dismissal. Those who questioned were vilified, threatened and bullied (INO 2005).

In a well-organised project, sound testing procedures are put in place. There are separate testing teams. Testing is recognised as a speciality in its own right. There is investment in automated testing tools.

3.3.3 Failure to Achieve Quality

3.3.3.1 Acceptance of Failure

In more recent years, the reality of software bugs is just accepted. *“Errors are unavoidable”* (German 2002). In former years efforts were only directed at ensuring that code was correctly written, in the first place. Not that those efforts are being neglected, rather tools and methods, which can identify defects, are supplementing them.

This realisation echoed the words of Maurice Wilkes, quoted earlier, who in 1949 said: *“As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered”* (Magnusson 2006). As Mark Halpin said in his memoirs *“fixing program bugs was our staple diet”* (Halpin 1992)

There is a curious change in the language used. Words such as “bugs”, “errors” and “flaws” are being replaced by “defects”, “weakness” and “vulnerability”. In this mindset, “errors” should not exist, but a “weakness” is something, which can be rectified.

To some extent this shift is a response to malicious Internet hackers. They seek to identify errors in software, so that they can exploit them. The response is to identify these errors before the hackers find them. A lot of money is spent on these exercises. In North America this exceeds a billion dollars annually (Garvey 2005)

3.3.4 Get right first time.

Efforts to address the issue of improved software quality can be classified into three approaches. They are: Process, Tools and Education, none of which can be taken in isolation. That is not to say that there aren't other approaches and variations on these themes

3.3.4.1 Process

This acceptance of defects is not universal. In a sense, the only reason why defects are accepted is that all of our efforts, to date, have failed to eradicate them. The earlier a defect is detected the less it costs to rectify it. Unfortunately, short-term objectives, often driven by marketing, results in lower quality being delivered sooner rather than superior quality being delivered later. It is therefore not surprising that we have so many “horror stories”.

A change of attitude is required. Sarah Saltzman argues that we need to take a longer-term view; we need to realise that making quality improvements at the development stage is much more cost effective than reacting to problems the quality assurance team picks up, or indeed fixing a bug once the application has been deployed. We *“need to foster a cultural change so that developers recognise that they will not only be measured on speed, but also on the consistency and quality of their code.”* (Saltzman 2005)

3.3.5 Can quality be achieved?

The ‘Worse is Better’ theory says that simplicity is paramount. There well may be a link between the number of defects in code and its comprehension (or simplicity), but that is yet to be established (Dunsmore and Roper 2000). It is difficult to measure the comprehension or simplicity of software because such comprehension is an internal process of humans (Uchida and Shima 2005).

Gabriel’s work concludes that we cannot impose rules on software creation, and it would be counter-productive to do so. *“It just won’t happen—it’s like those rockets: We simply do not know how to get massive software off the ground without crashing and endless fiddling. But we don’t accept that.”* He terms his solution as ‘Mob Software’ *“The way out of this predicament is this simple: Set up a fairly clear architectural direction, produce a decent first cut at some of the functionality, let loose the source code, and then turn it over to a mob.”* (Gabriel 2000).

This paper agrees with this thesis and advocates ‘empowering the mob’. There should be more emphasis on helping the programmer to acquire the necessary skills to write code, rather than imposing excessive regulation.

3.3.5.1 Methodologies to Achieve Quality

It is said that: “*a bad workman blames his tools*”. By methodologies we mean the tools used and the methods employed.

The various suppliers of compilers and operating systems now supply sophisticated IDEs (Integrated Development Environments) to support the writing software. These have improved greatly over recent years. Since these are pre-determined by the supplier, they will not be further discussed here. Other than, as will be said below, that it is better to learn using the same IDE as will be used in software production.

3.3.5.2 Process

“The quality of a product is largely governed by the quality of the process used to build it”. (Humphrey 1997)

Process or Methodology solutions represent the first attempts to address the issue of software quality. It was proposed that engineering principles be applied to software construction. Watts Humphrey was mentioned in the introduction to this work. He is trying to lead the industry toward defect-free software through the use of two methods--the Personal Software Process and the Team Software process--that use advanced engineering techniques. Humphrey also developed the basis for the Capability Maturity Model for Software, which became the generally accepted standard for assessing and improving software processes (Stahl 2005)

3.3.5.2.1 Capability Maturity Model

This dissertation opened with a reference to the award given to Watts Humphrey. He is known as ‘the father of software quality’. He is formulated the ‘Capability Maturity Model’ or CMM. He later formulated ‘Personal Software Process’ or PSP (Humphrey 1997). Watts Humphrey stated: *“the software task should be treated as a process that can be controlled, measured, and improved”* (Humphrey 1994). He was not the first to advocate this approach, but the CMM did formalise it. Essentially, they sell ‘a kit’. This enables an organisation to measure how ‘mature’ its ‘capability’ is to create and maintain software. There are 18 ‘key processes’ to be measured. On each the organisation can be at five levels. The fifth, or highest, is ‘mature’. These ‘five levels of software maturity’ (Paulk, Curtis, Chrissis, and Weber 1993) are:

- Level 1 - The Initial Level
- Level 2 - The Repeatable Level
- Level 3 - The Defined Level
- Level 4 - The Managed Level
- Level 5 - The Optimizing Level

When this information is gathered, there is a ‘spice’, that is: *Software Process Improvement and Capability dEtermination*. This yields 35 processes organized in 5 categories. Each process can be performed at 6 different levels. It is the job of management, guided by consultants, to prioritise these. Individuals, in the organisation, are then tasked to raise the performance level of a process, for which they are responsible, to a higher level.

CMM is highly regarded. However it is not without its critics. Although it isn’t complex, as such, there is a lot of detail. The CMM can seem to be overly bureaucratic, promoting process over substance.

He wasn't the first to propose such an approach. In 1985 IBM held their 'Systems Management Institute' at various locations. They quoted from (Gibson and Nolan 1974) who spoke of the 'four stages of EDP growth'. (Stage four was 'maturity').

In 2002 the Carnegie Mellon Software Institute announced the 'sunset' of CMM. They now advocate CMMI-SE/SW

3.3.6 Quality - Summary

Software quality is important but it is as elusive as ever. We have tried standards, methodologies, various techniques and procedures. While we may be reducing the frequency and impact of errors, they remain. As Fred Brooks wrote "there is no silver bullet" (Brooks 1987)

The issue is fundamental. The commission on electronic voting tell us that a C-language computer program failed, on occasion, to correctly count votes cast. One wonders not only about their processes and procedures, but also about the calibre of the programmers. Computer Science graduates can secure their qualifications, yet many cannot construct a simple program (Jenkins 2001).

It is time to focus on their education and learning; time to recognise the "mob", in the context of 'mob software' (Gabriel, 2000)

3.4 Education as a solution

The difference in the productivity can be dramatic. Francis McKeagney, CEO of InnerWorkings reports that 2% of programmers were responsible for half the software produced. It

It was said many years ago, it is still true: (Stackman 1968)

“When a programmer is good,

He is very, very good,

But when he is bad,

He is horrid.”

Given that there is such a disparity, it should be profitable and possible to raise the ability of the laggards.

3.4.1 Deficiencies in Education

Although the industry spends money technical education and individuals spend time learning technology, little guidance is directed towards this education. A recent survey (CompTIA 2006) found that although individuals spent 11 hours a week and \$2,200 on education in the past year, and plan to increase that amount by another \$100 in the next year, only eight percent make these choices based on employer requirements or recommendations. This survey showed that most of the cost of IT education is paid by individuals rather than their employers. It is therefore no surprise that 47% of IT staff do not receive critical training (Swartz 2005)

IT professionals may pay for their own, possibly inappropriate, education, but what of the end users? Nearly 90 percent of the cost of software for end users is wasted because of a lack of training. (Gartenberg 2005)

Organisations spend, on average \$600 per year per employee on their development. But little of their training appears to be transferred to the job (Zenger, Folkman and Sherwin 2006). It seems that we can impart knowledge, but that the application of that knowledge fails to be implemented.

Even when employees are given training opportunities, it's not always clear that the training results in the expected outcome. According to psychologist Daniel Goleman, author of: *Working With Emotional Intelligence*, "*Estimates of the extent to which skills taught in company training programs carry over into day-to-day practice on the job are as low — and gloomy — as a mere 10%*" (Putrich 2005).

There is reason to believe that many of the problems with software development can be attributed to deficiencies in computer science degree programs and similar professional education. Graduates know the syntax of the language used and details of the associated libraries, but not how to implement or test a system. (Knight & Leveson 2006)

Computer Science graduates can secure their qualifications, yet many cannot construct a simple program (Jenkins 2001).

3.4.2 Motivation

"A student will not learn unless they are motivated. It must be a teacher's main task, therefore, to ensure that all their students are properly motivated" (Jenkins 2001).

Despite the work of Biggs, Ramsden and others, a recent survey by Development Dimensions International shows that motivation is often overlooked (King 2006).

Much of the current literature sees lack of motivation as the primary failure (Biggs 2003) (Ramsden 2003). The advocated solution is to implement a student-centric approach. The assertion is that there are two approaches to IT training. In the first, you start by showing all the features of the new system and then eventually demonstrate how they can do the specific things they'll need to get real work done. The second way is to start by demonstrating how they can do their work with the new system and only later point out the rest of the features that are available somewhere in the software. (Hayes 2005) Or to put it another way: You can focus on your work, or you can focus on their work. The rhetorical question then asked is: *"Which approach will work better?"* (Oliver 2004).

We need to motivate. The question is how to motivate.

3.4.2.1 Motivators

Fear can be a motivator. Fear is culturally accepted as a motivator. Marketers use it to sell products, and politicians use it to get elected. But using fear as a motivator is wrong. It increases error. It destroys creativity (McManus, 2006)

Financial rewards can be a motivator. Money is effective is in lighting a motivational fire -- even in employees who claim money doesn't matter to them (Welch and Welch, 2006).

Personal affinity, where the students take ownership, achieves a high level of participation. (Way, 2006).

Others are recognition and celebration. An older list is: (LeDuc, 1980). This list does not recognise the issue of 'answering challenges'.

- Full appreciation of work done
- Feeling of being in on things
- Sympathetic help on personnel problems
- Job security
- Good wages
- Interesting work
- Promotional growth in the organization
- Personal loyalty to employees
- Good working conditions
- Tactful disciplining

We will now further explore this issue, seeking to maximise personal affinity. The recent literature on teaching programming has focused on motivation, and on-line courseware. This paper proposes another such initiative.

3.4.3 Structure of the Observed Learning Outcome

John Biggs argues against the assessment system, which simply rewards memorising information and repeating it on cue (Biggs 1982). This, he describes as “declarative knowledge”, rather than “*performative understanding where students use what they know to solve problems that reflect the real world.*” (Later, in this document, the InnerWorkings challenge will be described as “moving ‘declarative knowledge’ to ‘procedural knowledge’.”)

Bloom’s taxonomy has been used to analyse and understand computer science courses. (Lister 2005) However many merge Bloom’s six classifications (Bloom 1956) into three.

Biggs proposed a five-level model of teaching, which he dubbed 'SOLO' or Structure of the Observed Learning Outcome. The levels being: 'Pre-structural', 'uni-structural', 'multi-structural', 'relational', and 'Extended abstract level'. (Lister, 2006) These five levels will be referred to again, later in this document, when the COBOL Quiz questions are discussed, on page 44.

When we consider these levels, the objective is to move understanding to the higher level. Programmers, as in any other discipline, will have different levels of understanding. In predictable situations, programmers with lower-level knowledge can adequately function. Higher level knowledge is required when dealing with a novel or unforeseen situation. Richard Gabriel spoke of (civil) engineering going so far, but to create you need an architect (Gabriel, 1996).

In recognition of the artistic components in software creation the University of Illinois now awards Master of Fine Arts in Software, it has been followed by a Bachelor of Software Development at New Mexico Highlands University. These programs follow the philosophy of Richard Gabriel (West and Rostal, 2005).

3.5 Reflection

3.5.1 Relevance

Although the syntax of programming can be learnt, there is a difficulty in putting it into practice. This is illustrated by the observation that although computer science qualifications continue to be awarded in ever increasing numbers, the catalogue of IT failures is undiminished. Reflection addresses the internal construction of knowledge which results from actual experience of the action. InnerWorkings claim that their offering is unique in offering practice based learning for IT. The literature on reflective learning discusses its application in the fields of teaching, health and business (Boud, 1985) (Pearson 1985) (Candy 1985). We are discussing the relevance of reflection to learning how to program, in the context of practice based learning.

3.5.2 Definition of reflection

There are various definitions of reflection. Some regard it as just thinking about a learning experience in order to understand it. (Boud 1985) (Reed and Koliba 1995) (Jacovi 2004). This would not distinguish between simple recall of memory, what some programmers might call “documentation” and a mental activity which would construct actual knowledge. Kemmis goes somewhat further saying that “*reflection is a political act which either hastens or defers the realization of a more rational, just and fulfilling society*” (Kemmis, 1985).

This issue is, at least partially addressed, by stating that there are three types of, or stages in, reflection;

Hatton describes these as: firstly: personal judgements; secondly: conversations with oneself; and thirdly: critical reflections. Only critical reflection influences behaviour. Hatton (1995)

Schon encourages us to 'think about our thinking'. His three levels are: firstly: spontaneous or thoughtless, secondly: repetitive, there has been unconscious learning and thirdly: we are aware of the understanding. (Schon 1984)

Our specific interest in reflective learning is in how it might apply to writing program code. This literature does say that reflection is necessary for learning to be internalised, although it does not speak of the necessity to actually do something, in our instance to write code; what some call performance learning. Learning Theory, heretofore has neglected adult learning. Ideas on reflection and transformation discuss adult learning.

These definitions seem to have lost an earlier insight. Back in 1929, Dewey was emphatic that reflection was not "*thinking cooped up in the mind*" (Dewey 1929). Mezirow returns to Dewey's perspective (Mezirow 1991)

Jack Mezirow's ideas, while still under the general description of reflection are described as "Transformational Learning". Mezirow enhanced Dewey's ideas on reflective learning with input from psychoanalytic theory (Boyd and Myers 1988) (cited by Imel 1998) and from critical social theory (Scott 1997).

3.5.3 Mental Blocks

Edsger Dijkstra spoke of how programmers can be limited by their knowledge of the syntax of a programming language. He reported: *“I have run a little programming experiment with really experienced volunteers, but something quite unintended and quite unexpected turned up. None of my volunteers found the obvious and most elegant solution. Upon closer analysis this turned out to have a common source: their notion of repetition was so tightly connected to the idea of an associated controlled variable to be stepped up, that they were mentally blocked from seeing the obvious. Their solutions were less efficient, needlessly hard to understand, and it took them a very long time to find them. It was a revealing, but also shocking experience for me.”* (Dijkstra 1972)

3.5.4 Motivators that work

3.5.4.1 Programming contests

Programming contests have always been attractive. Winning is, no doubt, is attractive (Gomes, 2006); yet, given the number of entrants that cannot be the sole motivation. Perhaps it is because young computer programmers like to battle for fame, money, and they love algorithms (Arefin, 2005). Whatever the motivation, it exists. InnerWorkings exploit that desire to compete.

There are many programming contests. Two, frequently mentioned are: the IOI (International Olympiad in Informatics) and the ACM-ICPC (Association for Computing Machinery) - (International Collegiate Programming Contest)

When the ACM (Association for Computing Machinery) first organised its ICPC (International Collegiate Programming Contest) in 1999, over 2,400 teams entered (Manne, 2000) In the autumn of 2005, more than 5,600 teams representing 1,733 universities from 84 countries participated in regional contests. The top 83 teams will compete at the 2006 ACM-ICPC World Finals championship on April 9-13, 2006 (Wessner 2006). Contest participation has increased seven-fold since 1997.

The United Nations Educational, Scientific and Cultural Organization (UNESCO) proposed the IOI (International Olympiad in Informatics). The first was held in 1989, see: <http://www.ioinformatics.org/>

Some others are:

- The 'USA Computing Olympiad' is at: <http://ace.delos.com/ioigate>
- The 'Internet Problem Solving Contest' <http://ipsc.ksp.sk/>, who have different rules, will be held on Friday, May the 19th, 2006.
- For the last three years, Google has sponsored a 'Code Jam' in which computer geeks from around the world compete to solve thorny programming problems for a \$10,000 grand prize. This year there were more than 14,500 contestants from 32 countries. (American Enterprise 2006)
- "TopCoder" <http://www.topcoder.com/>, which has weekly on-line contests with the winners completing, annually, live in Los Vegas. However, this organisation is different. It really is contract programming. Prize money is actually a wage or a contract payment. One competitor 'won' \$75,000 (Hammonds 2004). However he is only one among 38,000 competitors.

While these contests are interesting in their own right, do they actually evaluate programming, let alone programming skills? It has been pointed (Shilova and Shilov 2005) they only evaluate:

- Art of problem formal modelling,
- Ability to remember a 'cook book' of algorithms
- Rapid typing skills

However, they do demonstrate response to a challenge.

While the 'art of modelling' is an important research skill, it is not a technical skill. While memory and keyboard speed are useful, they are not vital components of 'software quality'.

These competitions do have different code judging engines. However they judge the code from the outside. Other than a scan for unapproved function calls, the 'quality' of the source is not evaluated. (This may not apply to 'Top Coder'). They compile and execute the code. They have pre-prepared test data, for which predicted results are expected. They are then, evaluated on the memory required, execution time and sometimes the time taken to write the code.

There is a book and many advice pages on how to compete in these contests. This advice can be at odds with good programming practice. Some make this point (Calder 2005).

One has to have sympathy for the disgruntled entrant who complained that his entry failed because he included too many comments! Since these contests punish rather than reward good programming practices, they could well be counter-productive.

This author was gratified that Eberhard Sturm of the University of Münster, Germany, writing of a contest *"I reviewed the postings of 1998 in the newsgroup "comp.lang.pl1" and here is my favourite solution posted ..."* *"I declared it "elegant" (Sturm 2000).*

No code judging engine even claims to identify 'elegance'.

3.5.4.2 Robot Wars

Robot Wars is yet another phenomenon illustrating how competition can encourage learning. Competition among humans has accelerated robot evolution (Branwyn 2003).

There was an event here in NCI. These 'wars' are evidence of response to a challenge.

3.5.4.3 Quizzes

Quizzes can be used to assess and reinforce learning. On-line testing better motivates students to do their own work, and allows the raising of standards in the courses. (Woit and Mason 200)

4 Method

The research question is whether 'Answering Challenges Enhances Learning', in the context of software development. This question will focus on the deployment of the Inner Workings practice-based learning system.

4.1 Discussion on Methods

There are difficulties in organising any experiment in software quality (Harrison 2005). To remove random effects, an experiment requires numbers. But very few experiments can afford that number of programmers. Some experiments depend on the deployment of students. However industry is understandably sceptical of results based on the performance of college students.

Harrison points out that although few experiments are rarely correctly configured, that we could validly study an individual subject. He quotes Skinner who said: *"Instead of studying a thousand rats for one hour each or a hundred rats for ten hours each, the investigator is likely to study one rat for a thousand hours"* (Skinner 1966). Harrison argues that single-subject experiments are valid, however they are almost non-existent.

4.2 Field Observation

The first exercise, the 'field observation' was to examine the role of answering challenges to reinforce knowledge of COBOL. The course is described in Field Observation on page 41. It took place in the context of a complete course. A review of the course formulated the hypothesis, that 'answering challenges enhances learning'. For the purposes of this dissertation, the final ten questions, where writing code was required, are of interest. It is intended that actually writing code will move knowledge from 'knowing' to 'doing'.

4.3 Field Study

The field study, as will be related later, followed the field observation. Using InnerWorkings Developer™, the answering of challenges was the focus of attention. InnerWorkings Developer™ is described on page 51. After three months, the participants were asked to rank the improvement, if any, in their learning. From their replies, we have a ‘wave pattern’ of their opinions. They were then asked how various factors could have contributed to their learning; factors such as ‘answering challenges’ or ‘reading the manual’.

Factors with a wave pattern which closely matched the pattern that learning had improved were deemed to be associated with that improvement. Those with patterns which did not match were deemed to be less associated with the learning which took place.

5 Field Observation

5.1 Relevance of the field observation

The objective of all educational courses is to continually improve. After each course is completed, it is reviewed to consider what went well, and what could be improved. This field observation considers a programming course, in industry. This observation will note positive aspects of the course, such as the quizzes. It will also note shortcomings, such as the use of a plain notepad-style editor.

The intention of the field observation was to consider an existing class in industry. It was during this observation that the hypothesis 'answering challenges enhances learning' was formed. This hypothesis was further tested by the later 'field study'.

5.2 Audience

The class represented a cross-section of new entrants to commercial programming. The course was delivered to ten new programmers, over a three month period. Three of the students had little or no previous computing experience. Three had BSc degrees in Computer Science. Two had MSc degrees in computing disciplines. Two were existing staff members. One previously worked in quality assurance, testing computer systems prior to production. This candidate would therefore have had considerable exposure to computing, but not to programming. The final candidate was not strictly part of the class. She works half-days availing of 'family-friendly' hours. This candidate progressed at a slower pace than the rest of the class, and has still to complete the course, as this paper is being written. The class, therefore, represented a cross-section of new entrants to commercial programming

5.3 Description of the Course

This course has been delivered over many years. Its format has been refined and improved. The principal textbook was 'Stern & Stern'. This particular text, now in its eleventh edition, is well regarded in COBOL circles (Stern & Stern 2006). The course material was hosted on an LCMS (Learning Content Management System). The bulk of the course covered the COBOL programming language. This portion of the course is discussed in this paper. Other topics, which are not discussed in this paper, but were covered on the course, were:

- VA, Visual Age™, the IDE 'integrated development environment' used to develop programs on the PC for delivery on the mainframe.
- LPEX, is a language sensitive editor with some diagnostics and a context-sensitive help; a constituent part of the IDE.
- JSP, 'Jackson Structured Programming', a programming design methodology.
- JCL, 'job control language' required to execute programs on the mainframe.
- TSO/ISPF, time sharing option, a mainframe development environment.
- LE, 'language environment', services provided by the mainframe operating system to application programs.
- Utility programs, such as SORT, File Aid™ and Endeavor™
- Testing techniques
- Local rules, regulations and standards

However the prime interest of this paper was that part of the COBOL course where the students wrote program code in answer to quiz questions.

The class was split for the final three weeks. Those who would work for the bank studied the HOGAN™ banking system, which those who were selected for Life Assurance studied the CLOAS™ system. Both systems are written in COBOL.

For the purposes of this paper, we will consider the, eight weeks, COBOL portion of the course. It was divided into fourteen parts, roughly corresponding to chapters in the Stern & Stern text. There were deviations from the text. Stern & Stern use the Micro Focus compiler, whereas the course used IBM compilers, so features specific to Micro Focus were excluded, such as the 'screen section'. Chapters 14, 15 and 17 were omitted and they are not compatible with our environment. Chapters 1 to 13 and chapter 16 were studied.

5.4 How typical was the course?

The course might be typical of others offered in industry. The PowerPoint slides were originally based on slides provided by Stern & Stern. The quiz questions were based on questions provided by Stern & Stern. These were extensively amended for the environment these programmers would eventually work in. However the structure remained the same. As noted earlier, in 'Education as a solution' on page 27, it is unusual to see this level of investment in education. Instructor-led training continues to be the norm.

The course might not be typical of courses offered in academia. Even though applications managing 85% of the world's business data are written in COBOL (Stern & Stern quoting Gartner) there are few academic courses in COBOL. In academia a student is responsible for their own success or failure. In industry, a failed student can be a negative reflection on the lecturer. In academia, students pay a fee. In industry they are paid a salary. However their future salary and placement is at the lecturer's recommendation. The lecturer, in turn, is judged on how the pupils eventually perform 'on the job'.

5.5 Structure of the Course

The COBOL portion of this course, which is the subject of this field observation, was divided into fourteen portions. Each portion had classroom sessions, power-point slides, sample computer programs to be written and a quiz.

5.6 Quiz

5.6.1 Structure of the quiz

To reinforce learning, students were asked to complete 14 sets of questions. Each set had 70 questions. Most of the questions were taken directly from Stern & Stern. See discussion, earlier on Bigg's SOLO levels in 'Structure of the Observed Learning Outcome' on page 31.

- There were 20 true/false questions that only required ability to repeat lecture information. Each merited 5 points; 'Pre-structural'.
- There were 20 'fill in the blanks' questions, which sought to endure that information was in context. Each merited 10 points; 'Uni-structural'.
- There were 20 multi-choice questions. Several aspects need to be understood, however they are still treated separately. Each merited 20 points; 'Multi-structural'.
- Finally there were the 10 questions where writing program code was required. Each merited 30 points. It is intended that actually writing code will move knowledge from 'knowing' to 'doing'; from 'declarative knowledge' to 'procedural knowledge'.

5.6.2 Purpose of the Quiz

The purpose of the quizzes was to 'challenge' the students. The students were motivated by these challenges. The quizzes reinforced learning. There was, some, feedback automatically generated as answers were entered, either complimenting and supplying additional information or correcting and encouraging. The issue of feedback is further discussed below.

Obviously the quiz can also be used for assessing the students. However that was a secondary consideration to enhancing their motivation.

5.7 Motivation

The students were motivated by the quiz. Answering 14 sets of 70 questions is not a trivial exercise. They all completed all of the questions. Some completed some sets of questions twice and even thrice when they were dissatisfied with their score.

As this was merely a field observation, rather than a study or experiment, it is speculative to attribute reasons for the motivation. However the eagerness of the students to complete the quizzes, and to complete them to a satisfactory level, is informal evidence that there was motivation; that answering challenges enhanced motivation.

Some of the laggards completed the quizzes in the evening, others early in the morning and some from their home PCs. A feature of an LCMS is that these times are logged.

Finally the course was finished, the class had gone out for a celebratory meal, and there was prize giving and a reception with the rest of the programmers. But two still had some questions unanswered. There could be no penalty for not completing them, and no material reward for completing them. However they did complete them. For whatever reason, they were motivated to complete these challenges.

This was, in part, a reason to embark on the field study.

5.8 Feedback

In conversation with the students, they expressed satisfaction with the immediate feedback they received from some of the questions. True-false, multi-choice and sometimes fill-in-the-blanks type questions can be set up to give immediate feedback. This feedback can encourage; it can motivate. However other questions cannot be configured to give immediate feedback. In particular, the final group of questions, where a segment of code is required to be written cannot be immediately assessed.

This is regrettable, for it is in writing code that procedural knowledge is invoked.

Another shortcoming, expressed by the students, of the LCMS is that code is entered via a notepad-style text editor. They would have preferred to have used a text-editor appropriate for COBOL, such as LPEX from IBM. Nevertheless there was nothing to prevent them from writing their code in LPEX, which would have performed some syntax checking, and then 'cut-and-paste' their code into the quiz answer panel.

The shortcomings they identified, in the writing code segments, were:

- There was an inevitable delay in receiving feedback
- Code was entered via a notepad style editor rather than an IDE.

5.8.1 Correcting Code

If, as this paper postulates, actually writing code is essential; and if, as the students maintained, prompt feedback (or corrections) are demanded, then there is an issue and a solution must be sought. (This paper will suggest that InnerWorkings Developer™ is such a solution).

Correcting program code is tedious. It takes time. Without fully compiling and testing, accuracy is lost. Teachers just do not have the time to do this properly. The net effect, frequently, is that such corrections are neither timely nor accurate.

The very area which is most deserving of our attention, if we wish to transform learning from 'knowing' to 'doing', is the area least supported.

5.9 Conclusions from the field observation

This programming course has been run over many years. It has been refined over those years. The success criteria for this, and similar courses in industry, are not the students' marks in an examination, but their manager's perception of their performance, in terms of productivity and quality of the program code they write. This has led to increasing emphasises on actual writing of code during the course. Previous incarnations of this course were of six months duration. Three months of education followed by a three month project. The project was usually a rewrite of an older system, the original often written in Assembly language. However, all the old Assembly language systems have been retired, replaced or rewritten. The adoption of complete systems, that is HOGAN for banking applications and CLOAS for life assurance, in which programs are, in fact, sub-programs of the overall system, means that it is not easy to identify three month projects which can be completed by novice programmers. So they are now released to their teams after three months education.

From experience of these classes, over the years, it is concluded that:

- Actual practice in writing code is necessary to transform their learning.

The evidence from this field observation is that:

- Students are motivated by challenge, which is answering the quiz.
- Students prefer to enter their code in an IDE or its language sensitive editor.

As InnerWorkings Developer™, possibly uniquely, meets these requirements, it was worthy of further attention, and the field study, discussed next

5.10 Other Observations from the Field Observation

Attending the course were two students (MSC1, MSC2) with MSc qualifications, three with a BSc (BSC1, BSC2, BSC3), two with internal computing experience (INT1, INT2) and three without specific academic qualification or prior experience (NON1, NON2, NON3).

(These grades differ from those declared at the prize giving ceremony, in that these penalise incorrect answers)

Figure 1 - Field Observation - Student Grades

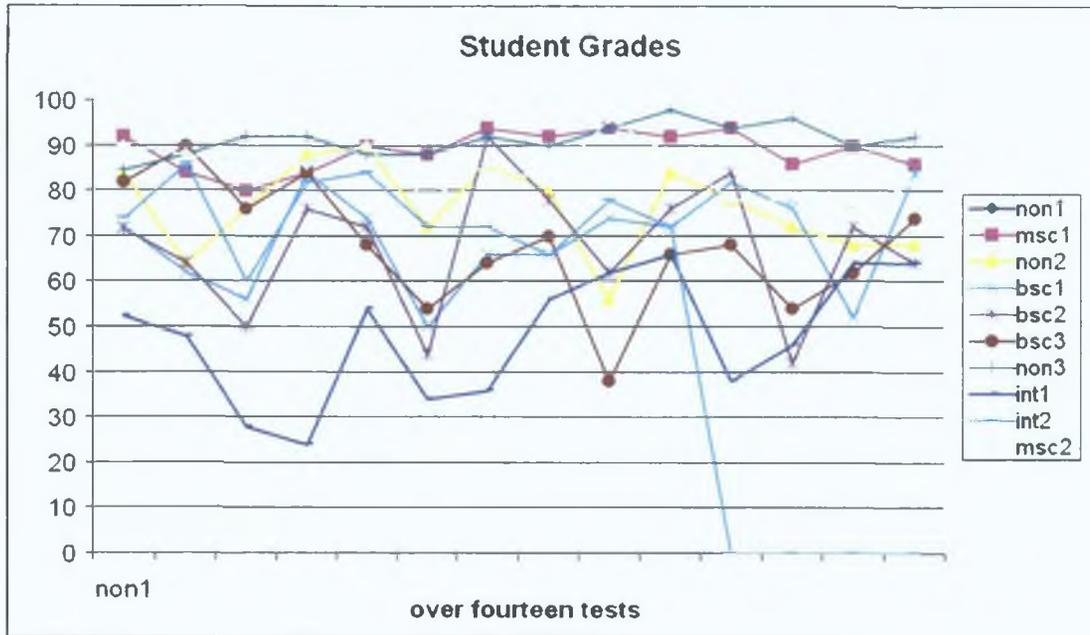


Table 1 - Field Observation - Student Grades over fourteen tests

90	68	78	80	72	86	76	72	86	82	92	88	86	84	non1
92	84	80	84	90	88	94	92	94	92	94	86	90	86	msc1
84	64	76	88	90	72	86	80	56	84	78	72	68	68	non2
72	62	56	84	74	50	66	66	74	72	82	76	52	84	bsc1
72	64	50	76	72	44	92	78	62	76	84	42	72	64	bsc2
82	90	76	84	68	54	64	70	38	66	68	54	62	74	bsc3
85	88	92	92	88	88	92	90	94	98	94	96	90	92	non3
53	48	28	24	54	34	36	56	62	66	38	46	64	64	int1
74	86	60	82	84	72	72	66	78	72					int2
90	88	86	86	90	78	86	82	96	88	78	78	76	82	msc2

Consider their average marks:

In spite of the range of prior academic achievement, there is little to distinguish the students. The best performing student had neither experience nor qualification; however the lowest performer had no qualifications either. The two MSc students did perform well. In simply looking at the grades, one cannot immediately distinguish BSc students from those without qualifications.

Table 2 - Field Observation - Average Grades

Average Marks	
non1	81
msc1	89
non2	76
bsc1	69
bsc2	68
bsc3	68
non3	91
int1	48
int2	75
msc2	85

Consider the standard deviation:

Here we see that the lowest deviation, that is those with the most consistent performances are the two MSc students and the one student without qualifications who was the best performer.

This highest deviation, those with the least consistency, is those with BSc qualifications and the student who was the lowest performer. Such a lack of consistency suggests that if these students are capable of improvement.

Table 3 - Field Observation - Standard Deviation of Grades

Standard Deviation	
non1	7
msc1	4
non2	10
bsc1	11
bsc2	15
bsc3	14
non3	4
int1	14
int2	8
msc2	6

It is outside the scope of this paper to consider why computer science graduates lack consistency. It is merely an observation.

However it is within the scope of this paper to further explore how answering challenges motivates; hence the field study

6 InnerWorkings Developer™

6.1 Field Observation

From the field observation the hypothesis was formed that

- 'Answering challenges enhances learning'

The subsidiary hypotheses formed were:

- 'Prompt feedback motivates'
- 'An IDE is helpful to learning'

It would, therefore, be instructive to identify a way in which these could be further studied to see if the hypothesis is confirmed in another environment. Hence the field study.

By serendipity or happy coincidence, InnerWorkings have developed a learning tool, called *InnerWorkings Developer™*, which does provide challenges, prompt feed-back and uses an IDE.

The field study was undertaken to further develop the hypothesis that 'answering challenges enhances learning'.

6.2 What is InnerWorkings Developer™?

InnerWorkings Developer™ is an interesting learning tool. The current implementation is focused on learning VB .NET and C# .NET. It does not seek to teach these computer languages, indeed one has to have a working knowledge of the language and of the Microsoft .NET IDE (Integrated Development Environment) to use it.

Rather, it seeks to 'make real', 'reinforce' or 'cement-in' knowledge. InnerWorkings term this as 'structured practice'. This paper considers it to be 'transformational learning'.

The usual approach to teaching programming is to 'learn by instruction'. Students are taught how to analyse problems, the syntax of a computer language and they are then expected to construct solutions. The emphasis is on syntax. They are told how to test and how to debug.

The InnerWorkings Learning Methodology addresses the neglected area of 'practice'. Although conventional programming education will expect students to write sample programs, there are practical limitations. This is discussed further, later, in a field observation.

The consequences of deficiencies in our programming skills were discussed earlier. Could it be that we have neglected a lesion from earlier ages in how skills are learnt? Could it be that this neglect has led to the current failure rate in software projects?

Many trades and professions have, at their core, systems of 'master and apprentice'. This has stood them well, sometimes for centuries. Employing one-to-one tuition in learning programming would be prohibitively expensive. Apart from which the idea would run counter to our ideas of being 'modern' and technologically innovative.

6.3 InnerWorkings Developer™ in Operation

It is a set of 'challenges', or 'drills', or problems, along with an environment (or desktop) in which to solve them. The actual writing of code is done in the usual Microsoft .NET IDE. This is the IDE which is used for the development of production (or real) programs. Each challenge is designed to take eight hours to complete

Help or 'learning support' is provided. Access is provided to the usual Microsoft help web-site 'Patterns and Practices'. Access is provided to Safari Books, but this assumes that such access has been purchased. Finally email support is provided by InnerWorkings. There is further comment on this, below, in the field study.

Once a challenge is completed, it is submitted to the code-judging engine, known as Inferent™. It identifies any errors and advises the student.

There are some other components which were not available for the field study, an administration platform and a reporting component. They were announced in June 2006. However these are more concerned with assessment rather than the learning process, which is of interest to this paper

6.4 InnerWorkings Developer Learning Issues

The principal differences between traditional learning activity and the learning activity which takes place using InnerWorkings Developer lie in the area of 'practice'. This is described as 'practice-based learning'. It moves learning from 'knowing' to 'doing'; from the theoretical to the practical.

The term 'drills' is sometimes used; this should not be confused with rote-learning. It is not an issue of memorising a repeated activity; rather it is the application of existing knowledge to answer a challenge or to solve a problem.

This method takes advantage of the students desire to answer challenges. The hypothesis of this paper is that answering challenges enhances learning. Students are motivated by answering challenges. This motivation is discussed in the second field report.

Students are motivated by immediate feedback. Traditional teaching relied on teacher or an instructor to review an answer. Just because a program compiles clean, and processes some test data, does not mean that it is correct. While feedback might be given, there is an inevitable delay. Inferent™ was found to give prompt and accurate feedback,

6.5 Claims of InnerWorkings Developer™

InnerWorkings Developer™ claims to:

- Teach
- Improve Quality
- Validate Skill

The field study did confirm its ability to improve knowledge.

6.5.1 Validating skills

InnerWorkings Developer™ has ability to 'judge code' with the Inferet™ engine. This can be used to assess programmers. As the reporting ability of InnerWorkings Developer™ was not available when the field study was contemplated, it was not tested. However the ability of Inferet™ to give immediate feedback was important.

6.5.2 How InnerWorkings Developer™ functions

The claims for this toolset were studied in the Field Study; the measurement methods are discussed, below.

There is a gap between 'knowing' and 'doing'. Traditionally many skills are acquired using the 'master and apprentice' system.

Inner Workings describe their toolset as unique.

The Inner Workings toolset is focused exclusively on Microsoft technologies.

This particular approach claims to be unique, if so we would not expect to find literature, which addresses this precise issue. However there is a wealth of literature describing other approaches to addressing the three primary objectives.

- Teaching software creation
- Improving software quality
- Assessing software skills

7 Field Study

7.1 Purpose of the field study

The purpose of the field study was to confirm the main hypothesis that “answering challenges enhances learning”. Subsidiary hypotheses would also be tested.

The principal hypothesis is:

- “Answering challenges enhances learning”

The subsidiary hypotheses are:

- “Prompt feedback motivates”
- “An IDE is helpful to learning”

These three hypotheses confirm the Field Observations.

How help features are used will be explored. Perhaps a future study will explore the use of context-sensitive help systems.

There will be a summary of how InnerWorkings Developer™ matches these hypotheses as well as a summary of the product from those testing it.

7.2 Participants in the field study

The volunteers participating in the field study were four groups of four. Most of the participants were employees of Irish Life and Permanent plc. IL&P is a financial services company formed from a series of recent mergers. The participants in this study were all on a voluntary basis. The approval of their managers had been obtained. However, participation was not a requirement of their job.

They all had prior experience of Visual Basic, but less exposure to .NET. Their participation in the study was facilitated by; and had the approval of their local management. The two participating groups were separated by geography and came from different parent companies. There was no contact between the groups. This adds to the confidence of the findings of the field study. Two groups, in different cities, came to similar conclusions.

All of participants are commercial programmers. Their programming experience ranged up to fifteen years, nine years being the average. They are, therefore, well positioned to express a judgement on any new learning software. In general, they endorsed the product and its 'practice based learning' methodology.

7.3 Organisation of the Field Study participants

Initially, there were sixteen volunteers, in four groups of four.

- Four were from IPSI, a subsidiary of IL&P, based in Dublin, which provides support for foreign, mainly Italian and German, Life Assurors. All four participated.
- Four were from the TSB, Trustee Savings Bank, which has recently merged with IL&P. All four participated.
- Of the group of four nominated by NCI, only one participated in the study.
- Of the final group of four volunteers, three were from other parts of IL&P; only one participated. (Actually two participated, but only one reported)

Of the two groups of four who knew each other, all completed the field study. Their participation in the study had been agreed by their managers. Of the other eight, who had weaker associations with one another, only two completed the field study. The implication is that mutual support, and management approval, ensured the success of the former, while the latter group suffered from isolation. For the purposes of this paper, this is merely an observation, deserving of further study.

The results are therefore grouped by the four from TSB in Cork, the four from IPSI in Dublin, as well as the two individuals, one from Irish Life Assurance and the final participant from NCI.

There was little or no monitoring of the participants during the study, so as not to contaminate their findings. While the IPSI group had a positive attitude towards InnerWorkings Developer™ from the outset, the TSB group were initially negative. Their attitude changed during the study. On enquiry, this was the result of one of their group spending time, at home, experimenting with the product, and then influencing their colleagues. There was no external influence on the participants. A negative conclusion would have been as acceptable as a positive conclusion. A difference between the IPSI and TSB groups was in their use of the InnerWorkings Developer™ 'Developer Support' email help facility, this is discussed later.

7.3.1 Field Study Questionnaire

The actual questionnaire is in the appendix. It was, mainly, a set of questions with a five point Likert scale. Its purpose was to establish:

How qualified were the participants to express judgement on this method (answering challenges) and on InnerWorkings Developer™. In short, how valid is this field study.

The important, central, question was whether, or not, learning was enhanced. This was explored. Was it reinforcement of existing learning or the acquisition of new knowledge? We conclude that it was both. Although both were positive, there is a minor divergence between the IPSI and TSB groups, which is explored. Another anomaly was that the participant with most programming experience, fifteen years was neutral on all questions in this section.

InnerWorkings Developer™ transforms learning by issuing challenges. So the role of these challenges is discussed. If these challenges were critical to the learning process, we would expect to see a level of correlation between the (total) actual learning reported and the learning which resulted from answering challenges.

7.4 How valid is this field study?

Or: How qualified are the participants?

How qualified are the participants to express an opinion on the hypothesis that answering challenges enhances learning and on InnerWorkings Developer™? To answer this question we will establish:

- Are they comfortable with computer assisted learning?
- Have they high expectations from the study?
- Are they representative of the target audience?
- Have they the expertise?

Are they comfortable with computer assisted learning?

Ideally they should be; if not, extra care would be required in interpreting their experiences. Would we be assessing their interaction with the computer or their interaction with InnerWorkings Developer™? **All were comfortable with computer assisted learning.**

Have they high expectations from the study? If the participants had low expectations, then they might be easily satisfied. A positive result would not predict a similar finding from a more demanding audience. **They had high expectations.**

Are they representative of the target audience? This study benefited from the selection of the participants. All, but one, of the participants are professional programmers. **They are the target audience for this learning method.**

Have they the expertise? This study benefited from the practical programming experience of the participants, who have a combined programming experience of well over seventy years.

This study was further enhanced by the prior specific expertise of the participants in the areas being addressed in the 'challenges'

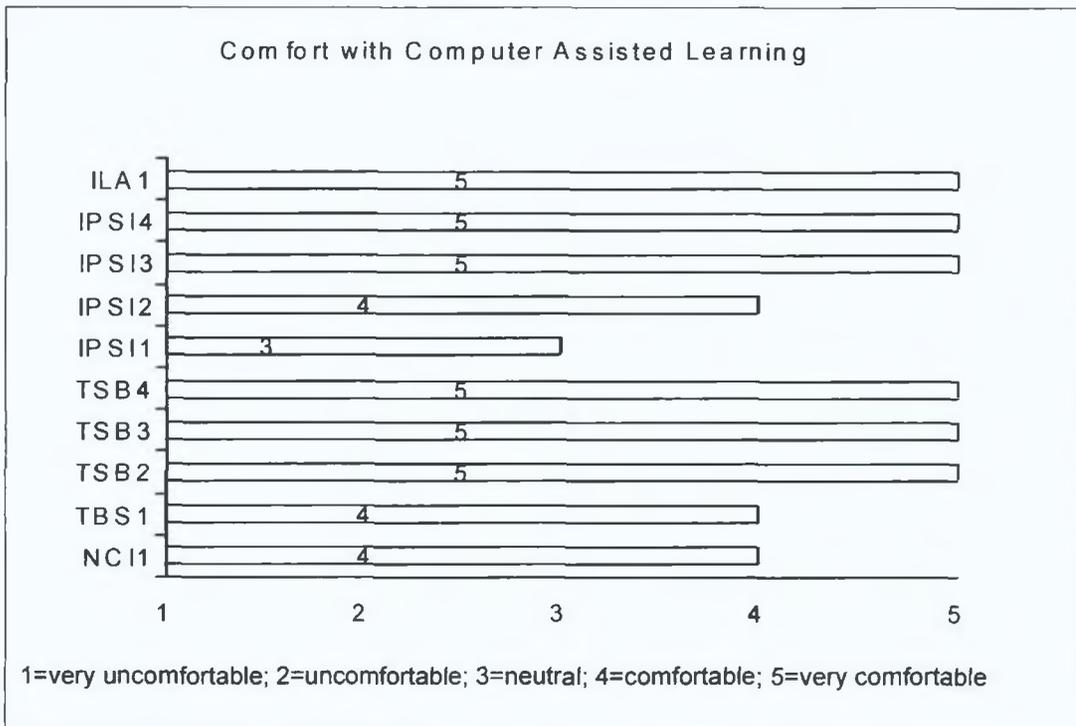
7.4.1 How Comfortable were they with Computer Assisted Learning

This might have posed some difficulty, if they were uncomfortable with Computer Assisted Learning. In the event, none were negative. The majority were “very comfortable” with Computer Assisted Learning

Table 4 - Field Study - How Comfortable are the Participants with Computer Assisted Learning?

NCI1	TBS1	TSB2	TSB3	TSB4	IPS11	IPS12	IPS13	IPS14	ILA1
4	4	5	5	5	3	4	5	5	5

Figure 2 - Field Study - How Comfortable are the Participants with Computer Assisted Learning?



7.4.2 Did the participants have a high or low expectation?

The prior expectations of the participants are relevant to the field study. If the participants had low expectations, then they might be easily satisfied. A positive result would not predict a similar finding from a more demanding audience.

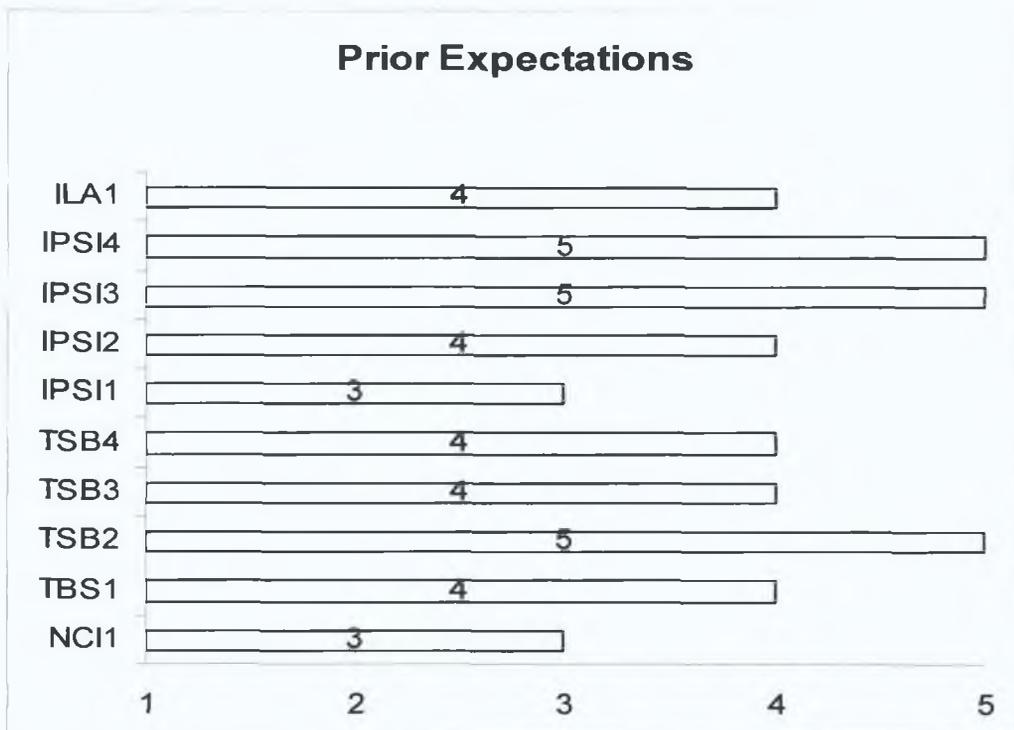
The participants did expect that their participation in the study would improve their knowledge

Table 5 - Field Study - Expectations of Participants

NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
3	4	5	4	4	3	4	5	5	4

In answer to "I expected InnerWorkings to benefit my knowledge"
 1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree.

Figure 3 - Field Study - Prior Expectations of the Participants



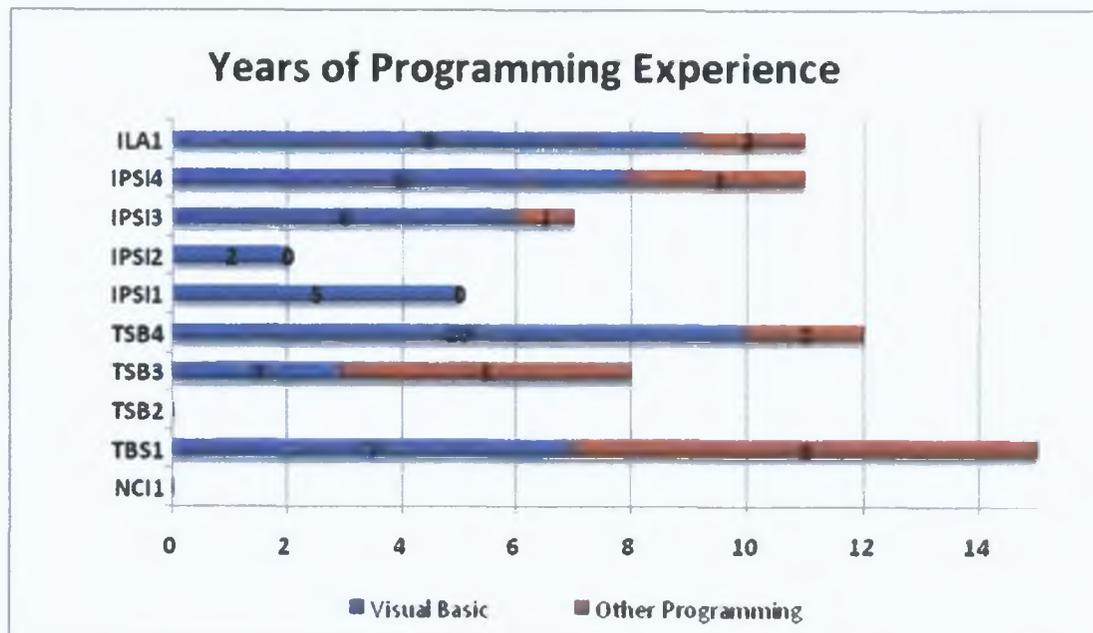
7.4.3 How qualified were the participants?

This set of question was to ascertain their prior knowledge and experience. This was to gauge their qualification to express opinions on the hypothesis and on InnerWorkings Developer™.

Table 6 - years of programming experience

Years of	NCI	TSB			IPSI				ILA	
All programming experience	?	15	?	8	12	5	2	7	11	11
Visual Basic Programming	?	7	?	3	10	5	2	6	8	9

Table 7 - Years of Programming Experience



The participants had considerable experience of programming, in particular programming in Visual Basic. They had, on average nine years programming experience. (Two participants choose not to answer this question). The TSB group were the most experienced, with an average twelve years programming experience. As noted earlier, this group were initially rather negative about the merits of this learning method and of on InnerWorkings Developer™, but later revised their opinion. It has been suggested that more experienced programmers are slower to adopt change and need more evidence before changing. However, for the purposes of this paper, this is just an observation, not a finding.

7.4.4 How knowledgeable were the participants?

While years experience of programming is an indication of a general ability to express a considered opinion, specific expertise in the technology being learnt would further enhance credibility.

Four questions were asked to ascertain specific expertise in:

- Object Oriented concepts
- Microsoft .NET
- web technologies
- XML

Again they display a high level of expertise, particularly the IPSI and TSB participants.

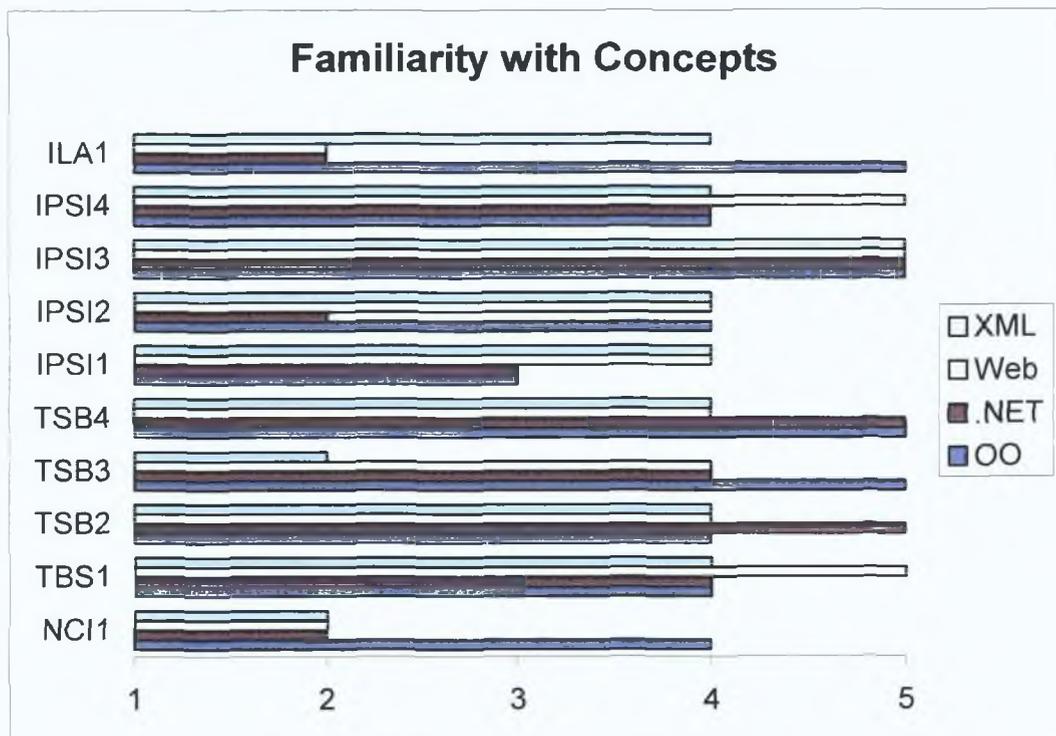
Table 8 - Participant Expertise

	NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
OO	4	4	4	5	5	3	4	5	4	5
.NET	2	4	5	4	5	3	2	5	4	2
Web	2	5	4	4	4	4	4	5	5	2
XML	2	4	4	2	4	4	4	5	4	4

In answer to the question "I was familiar with":

1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree.

Table 9 - Participant Expertise



7.4.5 How qualified are the participants - conclusion

We can conclude that based on the above criteria, the participants were qualified to express judgement on this method (answering challenges) and on InnerWorkings Developer™.

7.5 Was learning enhanced?

For the purposes of investigating InnerWorkings Developer™, this was the central question. For the purposes of the hypothesis, that answering challenges enhances learning, some refinement is required. However InnerWorkings Developer™ issues challenges. That is how it supports learning. In a very real sense, if InnerWorkings Developer™ enhances learning, then the hypothesis is established.

7.5.1 Was learning enhanced?

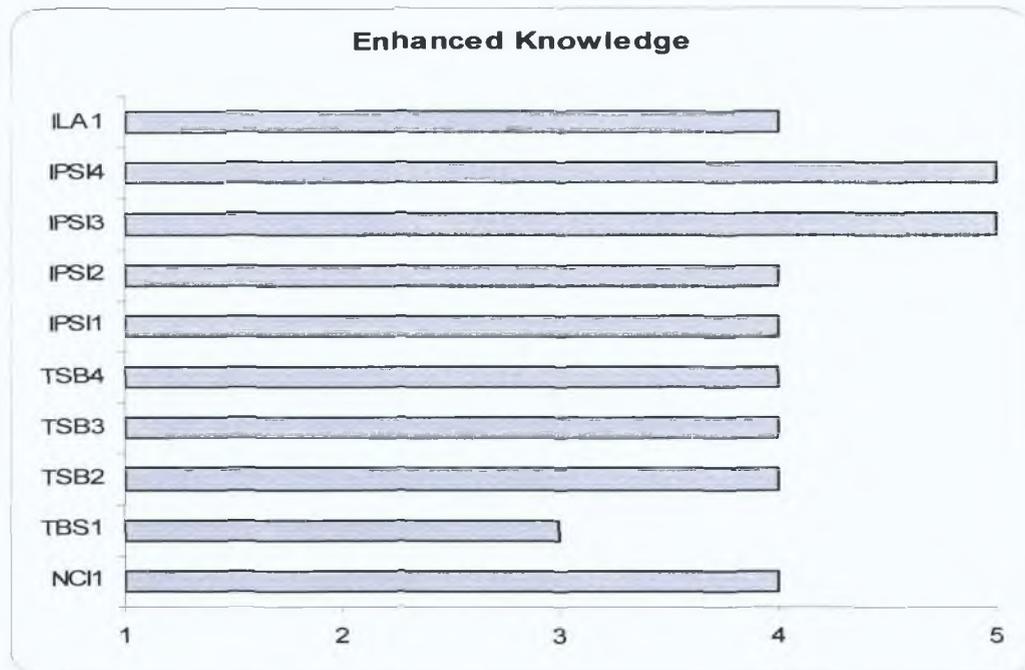
The central question on the questionnaire was: “InnerWorkings did enhance my knowledge”

Table 10 - Was Learning Enhanced?

NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
4	3	4	4	4	4	4	5	5	4

1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree.

Table 11 - Was Learning Enhanced?



All agreed that their knowledge had been enhanced by their participation in the field study. The lead TSB programmer, with fifteen years experience, was neutral on this, and all other, questions in this section. The simplistic explanation is that more sophisticated learning software would be required to impact on one already so skilled. He was critical of the programmer support (help feature), which would have influenced his views. This is discussed later. The field study lasted for three months. After the first month, the TSB group was rather negative in their conclusions. Without any outside intervention, they re-evaluated their opinion, prompted by one of their own number who continued to use InnerWorkings Developer™ from his home PC. Although both the TSB group and the IPSI group were, in overall terms, positive towards InnerWorkings Developer™, there were interesting, albeit minor, differences between them. These are discussed later.

7.5.2 Knowledge Transformation

Or: 'Reinforce Existing Knowledge or Acquire New Knowledge'

'Knowledge Transformation' or 'practice based learning', appears initially to be simply just 'drills' or 'learning by rote'. It is that, but knowledge transformation is much more, it is true learning. InnerWorkings Developer™ does not, initially, have any learning content. It is not a primer on the VB .NET computer language.

If all that was happening with practice based learning was that programming exercises were being carried out, we would expect to see that existing knowledge was being reinforced. We would not expect to see any acquisition of new knowledge.

If, on the other hand, we were discussing traditional CBT Computer Based Training, or indeed many classroom 'talk and chalk' learning, we would expect there to be acquisition of new knowledge with little reinforcement of existing knowledge. There would only be reinforcement of existing knowledge during revision sessions.

Yet, we find, that although there is not any explicit delivery of new knowledge, the participants all report the expected reinforcement of existing knowledge, as well as acquisition of new knowledge, in almost equal amounts. Another factor must be operating for this effect. That factor, this paper suggests, is transformational learning. Learning is not truly acquired until it is practiced. If this is true, then it would be natural for the participants to report that new knowledge was acquired.

It is interesting that the participants reports on the reinforcement of existing knowledge and the acquisition of new knowledge are nigh identical.

I reinforced existing learning from InnerWorkings
 I learnt something new from InnerWorkings

Table 12 - New and Existing Knowledge

	NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
reinforced existing learning	4	3	4	4	4	4	4	5	5	5
I learnt something new	4	3	5	4	4	4	4	5	5	4

1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree.

Table 13 - New and Existing Knowledge



7.5.3 How was learning enhanced?

7.5.3.1 Microsoft help feature

It could be argued that since InnerWorkings Developer™ provides a portal through to the Microsoft .NET help facility, that this could have been providing the new learning. The question “*I found the Microsoft help feature helpful*” was asked. There was no correlation. This was not the source of the new learning.

7.5.3.2 Other sources of learning

Further questions were asked to ascertain whether other possible sources of learning could be responsible for this new learning:

- I found the Microsoft help feature helpful
- I helped myself, learning through trial & error
- I helped myself, by reading the manual
- I helped myself, by reviewing other course notes
- I got help by asking a friend
- I got help via the InnerWorkings email facility

These did not correlate to the learning experience, whereas the challenges did:

- I got satisfaction from responding to challenges
- I learnt from the challenges

7.5.4 Graph of Results

When the answers to the question, the replies can be plotted:

7.5.4.1 Knowledge

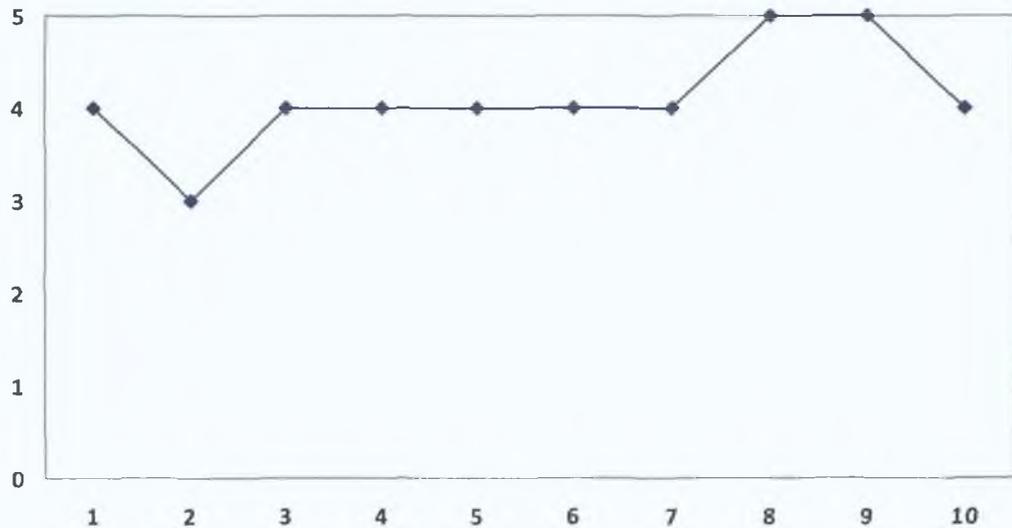
“InnerWorkings did enhance my knowledge”

1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree.

Table 14 - Knowledge was enhanced

NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
4	3	4	4	4	4	4	5	5	4

Figure 4 - Knowledge was Enhanced



This is the base line. Other graph lines can be compared with this line. If there is a close match, then the probability is that there is a relationship between the two sets of data, as with cause and effect. If there is no match then there is unlikely to be a relationship.

7.5.5 Expectation

To illustrate, compare the answers to 'I expected InnerWorkings to benefit my knowledge' to the base line 'InnerWorkings did enhance my knowledge'

When we compare that set of answers to some other questions, such as: I expected InnerWorkings to benefit my knowledge, and plot both together; there is a very similar pattern:

InnerWorkings did enhance my knowledge

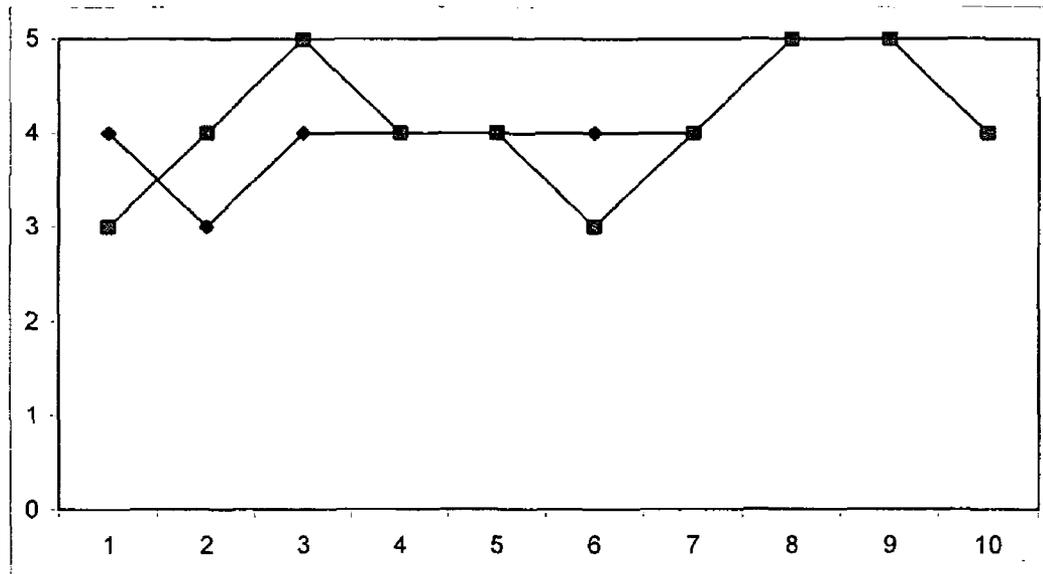
I expected InnerWorkings to benefit my knowledge

1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree.

Table 15 - Expectation and Actual

NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
4	3	4	4	4	4	4	5	5	4
3	4	5	4	4	3	4	5	5	4

Figure 5 - Compare Knowledge and Expectation



7.5.6 Challenge

Similarly, and importantly, for this study the graph of knowledge and challenges are very close. This confirms that there is a close relationship, during this field study, between answering challenges and the learning acquired.

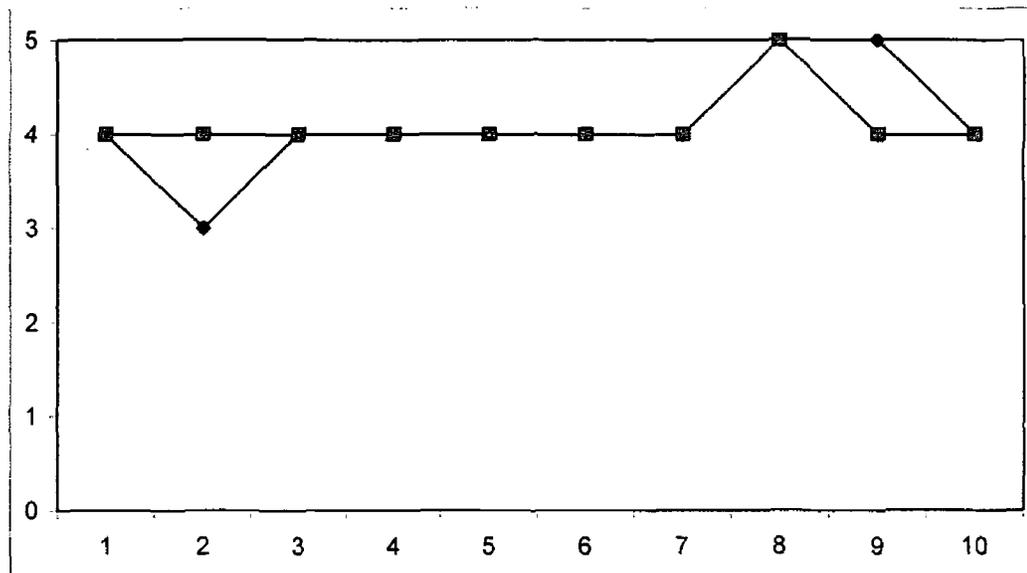
InnerWorkings did enhance my knowledge
I learnt from the challenges

1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree

Table 16 - Compare Challenge and Knowledge

NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
4	3	4	4	4	4	4	5	5	4
4	4	4	4	4	3	4	5	4	4

Figure 6 - Compare Knowledge and Challenge



7.5.7 Read the Manual

It might have been the case that significant learning was acquired in other ways, so a series of questions were asked:

- I found the Microsoft help feature helpful
- I helped myself, learning through trial & error
- I helped myself, by reading the manual
- I helped myself, by reviewing other course notes
- I got help by asking a friend
- I got help via the InnerWorkings email facility

Their graphs were not similar to the 'knowledge' baseline. Consider 'I helped myself, by reading the manual'

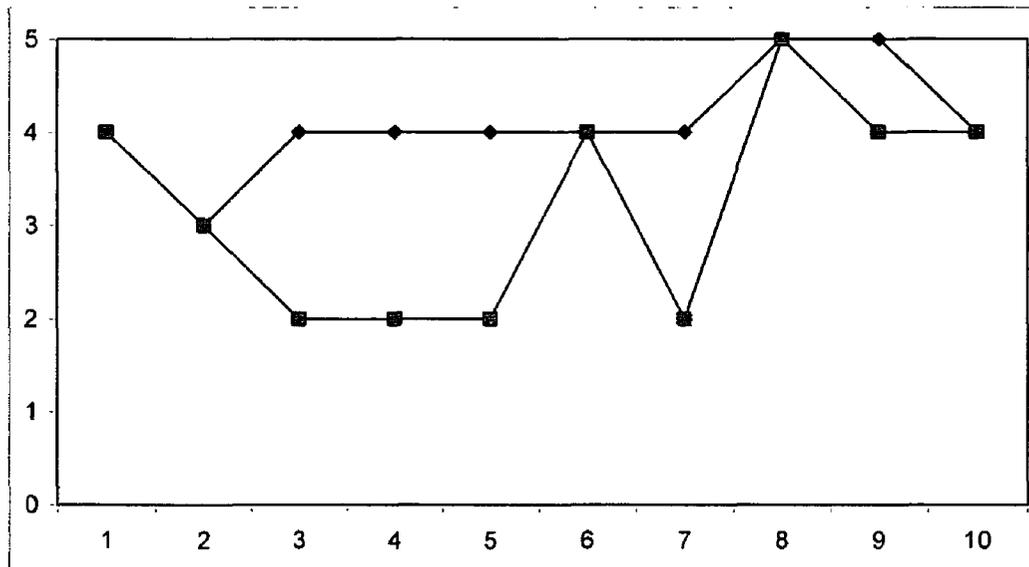
I helped myself, by reading the manual'

1=strongly disagree; 2=disagree; 3=neutral; 4=agree; 5=strongly agree

Table 17 Reading the Manual to Acquire Knowledge

NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
4	3	4	4	4	4	4	5	5	4
4	3	2	2	2	4	2	5	4	4

Figure 7 - Reading the Manual & Acquiring Knowledge



7.5.8 Summary

Using this mechanism, there appears to be a high correlation for some questions, and weak or low correlation for others.

There was a high correlation for:

- I got satisfaction from responding to challenges
- I learnt from the challenges
- I helped myself, learning through trial & error

There was a weak correlation for:

- I helped myself, by reading the manual
- I helped myself, by reviewing other course notes

There was a low correlation for:

- I found the Microsoft help feature helpful
- I got help by asking a friend
- I got help via the InnerWorkings email facility

7.5.9 Spearman's rho

Spearman's rho test can be used to verify correlations. Computing the non-parametric correlations between 'knowledge' and 'challenge' gives a result of .574. This is a high correlation. However repeating this test, comparing 'knowledge' and 'reading the manual' gave a similarly high result of .499.

This test confirms a close relationship between answering challenges and acquiring knowledge. That this test appears to contradict the earlier graph on the issue of learning through reading the manual does not negate the primary hypothesis, that 'answering challenges enhances learning'. Indeed, it would be of positive interest if the exercise of using InnerWorkings Developer™ resulted in learning through reading the manual.

7.6 Field Study Conclusion

These statistics, therefore support the hypothesis that, in this case, learning was enhanced by answering challenges

8 Future Perspectives

This exercise illustrates the promise of this new method of improving programming skills, and hopefully future software. The study was limited in its scope and extent. However, as earlier mentioned, there were positive attributes to the study.

The following list might be useful for future studies and experiments:

The study was not a true experiment as there was no 'control group'. The original intention was to trial the product with a larger group of programmers. Unfortunately their availability was postponed.

The extent of improvement was not measured:

Teaching / Learning – was it better? Could it be achieved with less effort?

Quality – is there an improvement in quality? Is there a reduction in the number of defects?

Assessment – how useful is the toolset in assessing students, (the reporting facility was not available)

9 References

- ACM (2005), Press Release,
http://campus.acm.org/public/pressroom/press_releases/3_2005/newell_award_3_15_2005.cfm Retrieved December 12 2005
- American Enterprise (2006), (anonymous) Nerd Inequality, *American Enterprise* March 2006, Vol. 17 Issue 2, p8,
- Arefin, Ahmed Shamsul (2005) Art of Programming Contest, *ACM Solver Publications*, Dhaka, Bangladesh December 2005
- Averbuch, Sheila M., (2004) *Irish League of Credit Unions looks to develop new IT platform* Electric News March 12, 2004
<http://www.electricnews.net/news.html?code=9403671> retrieved 7 August 2006
- Bernecky, Robert (2004) *Rewriting History Computing Canada*, March 12, 2004
- Biggs, John B. & Collis, K. F. (1982) *Evaluating the quality of learning: The SOLO taxonomy*. New York, Academic Press.
- Biggs, John B., (2003) *Teaching for Quality Learning at University: What the student does* (2nd edition). *Open University Press*, Maidenhead, Buckinghamshire. ISBN: 0-334-21168-2
- Bloom, B. S. (Ed.). (1956): *Taxonomy of Educational Objectives Handbook 1: Cognitive Domain*. New York: Longman, Green & Co.
- Boud, D., (1985). Keogh, R., & Walker, D. (Eds), *Reflection: Turning Experience into Learning*. New York, Nichols
- Boyd, Robert D., and Myers, J. Gordon. (1988) Transformative Education, *International Journal of Lifelong Education*. Vol 7, no. 4 (October-December 1988): pp 261-284.
- Branwyn, Gareth (2003) *Absolute Beginner's Guide to Building Robots* *Que* September 19, 2003; ISBN: 0-7897-2971-7
- Brooks, Frederick P. Jr., (1987), No Silver Bullet: Essence and Accidents of Software Engineering, *Computer*, Vol. 20, No. 4. April 1987 pp. 10-19.
- Bush, George, W., (2005), *Whitehouse Press Briefing*
<http://www.whitehouse.gov/news/releases/2005/03/20050314.html> Retrieved December 12, 2005
- Calder, Brad (2005) *General Tips*
<http://www.cse.ucsd.edu/users/calder/UCSDProgramContest/tips.html> Retrieved November 11, 2005

- Campbell-Kelly, Martin (1998) Programming the EDSAC: Early Programming Activity at the University of Cambridge *IEEE Annals of the History of Computing* October-December 1998 Vol. 20, No. 4, pp 46-67
- Commission on Electronic Voting (2006)
http://www.cev.ie/htm/report/download_second.htm Retrieved 7 August 2006. Page 106 and 189
- CompTIA (Computing Technology Industry Association) (2006), Tech Workers Get Little Career Guidance or Support, *Certification Magazine*; March 2006, Vol. 8 Issue 3, p8
- Candy, P., Harri-Augstein, S., & Thomas, L. (1985). *Reflection and the Self-Organized Learner: a model of Learning Conversations. Reflection: Turning experience into learning* D. Boud, Keogh, R., & Walker, D. (Eds), London, Kogan
- Cox, Brad, (1992), What if there is a Silver Bullet and the competition gets it first?, *Journal of Object-oriented Programming*, June 1992. Reprinted in Dr. Dobb's Journal October 1992
- Dewey, J. (1929). *The Quest for Certainty*, Southern Illinois University Press, USA.
- Dijkstra, Edsger W., (1972) The Humble Programmer *Communications of the ACM* Vol 15 Issue 10 (October 1972) pp 859-866
 ISSN:0001-0782
- Dijkstra, Edsger W., (April 18, 1975) Craftsman or Scientist? *Luncheon Speech at "ACM Pacific 75" at San Francisco*
- Donald Schon (1983), *The Reflective Practitioner: How Professionals Think in Action* p 54
- Dunsmore, Alistair and Roper, Marc, (2000), A Comparative Evaluation of Program Comprehension Measures *The Journal of Systems and Software* vol. 52, no. 3, 2000, pp. 121-129
- Eagles (Evaluation of Natural Language Processing Systems) (October 1996) *Final Report* <http://www.issco.unige.ch/projects/ewq96/> retrieved December 2005.
- Fagan, M. E. (1976) Design and Code Inspections to Reduce Errors in Program Development, *IBM Systems Journal*, No. 3, pp. 184-211
- Fisher Lawrence, (1994), Pentium flaw creates confusion for PC buyers, *New York Times* December 14 1994, pp. D1, D18.

- Gabriel, Richard P., (1990). Lisp: Good News, Bad News, How to Win Big, Keynote address, known as "Worse is Better," presented at the *European Conference on the Practical Applications of Lisp*, Cambridge University, March. 1990, Reprinted in *AI Expert*, June 1991, pp. 31–39.
Published in *Patterns of Software*, 1996, Oxford University Press; ISBN: 0195121236, see also:
<http://dreamsongs.com/NewFiles/PatternsOfSoftware.pdf>
retrieved 5 August 2006
- Gabriel, Richard P., (1996) *Patterns of Software*, Oxford University Press; ISBN: 0195121236 (see link above)
- Gabriel, Richard P., (2000). "Mob Software: The Erotic Life of Code". *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, October 19, 2000, in Minneapolis, Minnesota, USA. See also:
<http://dreamsongs.com/NewFiles/MobSoftware.pdf> retrieved 5 August 2006
- Gartenberg, Michael, (2005), The High Cost of Not Training, *Computer World*, July 11, 2005 p 21
- Garvey, Martin J., (2005), Logic Library application helps developers improve software quality, reduce quality-assurance times, and ensure secure software distribution, *Information Weekly* 11 April 2005
- German, Daniel M., (2002), Debugging,
http://turingmachine.org/courses/2002/senq265F02/lectures/12_debug.pdf retrieved on 12 March 2006.
- Gibson, Cyrus F. and Nolan, Richard, L (1974) Managing the four stages of EDP growth *Harvard Business Review* January-February 1974 Vol 52 No. 1, pp 76-88
- Gomes, Lee (2006) Programming Contest Pits World's Top Geeks In Battles Over Coding *The Wall Street Journal* February 8 2006
- Halpin Mark (1992) *Memoirs (Part 2) Annals of the History of Computing*, Vol 14, No. 1, pp 61-69,
- Hammonds, Keith H. (2004) There are good people everywhere *Fast Company*, July 2004 Issue 84, p40,
- Harrison, Warren (2005) Skinner Wasn't a Software Engineer *IEEE Software* May / June 2005 pp 5-7
- Hatton, N. S., D. (1995). "Reflecting in teacher education: towards definition and implementation" *Teacher and Teaching Education* // pp. 33 – 49

- Hayes, Frank, (2005), Messy Training, *Computer World*, August 15, 2005, p 50
- Hopper, Grace Murray, (1981), The First Bug, *Annals of the History of Computing*, vol. 3, no. 3, pp. 285-286, 1981.
- Hopper, Grace Murray, (1945), <http://ei.cs.vt.edu/~history/Bug.GIF> retrieved January 6 2006, also <http://www.waterholes.com/~dennette/1996/hopper/bug.htm> retrieved March 18, 2006
- Humphrey, Watts S, (1994) *A Discipline for Software Engineering*, Addison-Wesley Professional, Reading, MA., ISBN: 0201546108
- Humphrey, Watts S., (1997) *Introduction to the Personal Software Process* Addison-Wesley Professional, Reading, MA., ISBN: 0201548097
- IEEE Computer Society (1993) *IEEE Standard 1061-1998, "IEEE Standard for a Software Quality Metrics Methodology"*, March 12 1993
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=6079&arnumber=237006&count=1
- Imel, Susan (1998) *Transformative Learning in Adulthood*, *ERIC Digest* no. 200. <http://www.calpro-online.org/eric/docs/dig200.pdf> retrieved 9 August 2006
- Information Week (2005) Do your IT division's priorities include improving software quality? *Information Week* July 4, 2005. <http://www.informationweek.com/story/showArticle.jhtml?articleID=165600249> retrieved Dec 4, 2005
- INO (Irish Nurses Organisation) *Press Release* 4 October 2005
<http://www.ino.ie/DesktopModules/Articles/ArticlesView.aspx?TabID=6129&ItemID=5243&mid=8026> retrieved 20 March 2006
- Irish Health (2005), *Irish Health*, 12 November 2005
<http://www.irishhealth.com/index.html?level=4&id=8321>
Retrieved March 4, 2006
- ISO 9126
ISO/IEC 9126-1:2001 Software engineering -- Product quality -- Part 1: Quality model
<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=22749&ICS1=35&ICS2=80&ICS3=> Retrieved 6 August 2006
ISO/IEC TR 9126-2:2003 Software engineering -- Product quality -- Part 2: External metrics
<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=22750&ICS1=35&ICS2=80&ICS3=> Retrieved 6 August 2006

- ISO/IEC TR 9126-3:2003 Software engineering -- Product quality -- Part 3: Internal metrics
<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=22891&ICS1=35&ICS2=80&ICS3=> Retrieved 6 August 2006
- ISO/IEC TR 9126-4:2004 Software engineering -- Product quality -- Part 4: Quality in use metrics
<http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=39752&ICS1=35&ICS2=80&ICS3=> Retrieved 6 August 2006
- Jacovi et al., M., Shahar, E., and Soroka, V. (2004). "Blogs for Corporate Learning" *IBM Research Report - Computer Science H-0231*, November 9, 2004
- Jedras, Jeff, (2004), Quality is job one in software engineering, *Computing Canada*, November 11, 2004, Vol. 30 Issue 17, p 28
- Jenkins, Tony (2001) Teaching Programming – A Journey From Teacher To Motivator, *First LTSN-ICS one day conference on the Teaching of Programming*. April 2, 2001 at University of Leeds
- Kelly, Joanne, (2005) Taoiseach's IT PPARs system comments incorrect" *Irish Developers Network News*, October 5, 2005
- Kemmis, S. (1985). *Action Research and the Politics of Reflection; Reflection: Turning experience into learning*. D. Boud, Keogh, R., & Walker, D. (Eds),. London, Kogan p140
- Kennedy, John (2005) Plan or be damned, warns expert on PPARS *Silicon Republic* October 5, 2005
- Kenny, Enda, (2005), St James's CEO PPARS Letter Blows Taoiseach's Defence of System Out of the Water, *Finn Gael Press Briefing*. October 5, 2005
- King, Bill (2006) Motivation is the Key to Successful Employee *Expansion Management*, January / February 2006, Vol. 21 Issue 1, p 2
- Knight, John C. & Leveson, Nancy G, (2006), Software and Higher Education. *Communications of the ACM*, January 2006, Vol. 49 Issue 1, p 160
- Knight, John C. & Leveson, Nancy G, (2002), Should Software Engineers be Licensed? *Communications of the ACM*, November 2002, Vol. 45 Issue 11, pp 87-90.
- Kontzer, Tony (2005) IT Execs to Vendors: Your Software Stinks *Information Week* April 28, 2005
<http://www.informationweek.com/story/showArticle.ihtml?articleID=161601417> retrieved 4 December 2005

- LeDuc Jr., A.L. (1980), Motivation of Programmers. *ACM SIGMIS Database*, Volume 11, Issue 4 (Summer 1980) pp 4-12, ISSN:0095-0033
- Lettice, John (2004) Failed Windows XP Upgrade Downs 60,000 UK Gov't PCs *eWeek*, November 27, 2004
<http://www.eweek.com/article2/0,1759,1732672.00.asp> retrieved October 5, 2005
- Lister, Raymond (2005). Methods for Evaluating the Appropriateness and Effectiveness of Summative Assessment via Multiple-choice Examinations for Technology-Focused Disciplines, *Making a Difference: 2005 Evaluations and Assessment Conference*. 30th. November – 1st. December, Sydney
- Lister Raymond (2006), Not Seeing the Forest for the Trees: Novice Programmers and the SOLO Taxonomy, *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education* Bologna, Italy pp118-122 ISSN: 0097-8418
- Lubar, David (1995). "It's not a bug it's a feature!" *Addison-Wesley, Reading, Mass.*, ISBN: 0201483041
- Magnusson, Peter S. (2006) Biting Bugs Back - System Level Simulation Speeds Software Debugging, *Embedded Technology Journal* March 14, 2006
- Manne, Fredrik (2000) *Competing in Computing* SIGCSE Bulletin 2000
- McCall, J., Richards, P. & Walters, G. (1977), Factors in software quality, *Technical Report: RADC TR-77-369, Vols. 1-3, Rome Air Development Centre, United States Air Force, Hanscom AFB, MA.*
- McManus, Kevin (2006). Are you Afraid? *Industrial Engineer*, April 2006, Vol. 38 Issue 4, p18
- Mezirow, J. (1991). *Transformative Dimensions of Adult Learning*, San Francisco, USA, Jossey Bass.
- Morgan, Jeanette Nasem, (2005), Why the Software Industry Needs a Good Ghost buster. *Communications of the ACM*, August 2005, Vol. 48 Issue 8, p129-133
- Newman, John Henry, (1845), *Development of Christian Doctrine*.
- Oliver, D., Dobele, T., Greber, M. and Roberts, T. (2004), This Course Has A Bloom Rating Of 3.9. *In Proceedings of the Sixth Australasian Computing Education Conference (ACE2004)*, Dunedin, New Zealand. CRPIT, vol. 30. Lister, R. and Young, A. L., Eds., ACS. 227-231

- Pall Mall Gazette, (March 11, 1889), quoted in the Oxford English Dictionary.
- Paulk, Mark C.; Curtis, Bill; Chrissis, Mary Beth; Weber, Charles V. (February 1993), *Capability Maturity Model for Software, Version 1.1 Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania*
<http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html> retrieved April, 8, 2006
- Paulk, Mark C., (October 16, 1994), *The Evolution of SEI's Capability Maturity Model for Software, Software Engineering Institute, CMM Evolution,*
- Pearson, M., and Smith, D. (1985). *Debriefing in Experience-based Learning. Reflection: Turning experience into learning.* D. Boud, Keogh, R., & Walker, D. (Eds),. London, Kogan
- Perlis, A.J. , (1970) *Software Engineering Techniques*, page 33, J.N.Buxton and B.Randell (editors), NATO Scientific Affairs Division, Brussels 39, Belgium, April 1970.
- PPARS (2005) *PPARS Newsletter* 14 April 2005
http://www.ppars.ie/Public/News/Newsletters/NewsletterV.3.%20Final14_04_05.ppt , Retrieved March 4 2006
- Pratt, Vaughan (July 26, 2005) *Anatomy of the Pentium Bug*
<http://boole.stanford.edu/pub/anapent.pdf> Retrieved March 18 2006
- Putrich, David, (2005), *Employee Development on a Shoestring*", *Computer World*, July 4 2005, p 32
- Ramsden, Paul (2003) *Learning to Teach in Higher Education*, 2nd edition *Routledge Falmer*, London ISBN: 0-415-30345-1
- Reed, J., and Koliba, C.,. (1995). "*Facilitating Reflection: A Manual for Leaders and Educators.*"
http://www.uvm.edu/~dewey/reflection_manual/index.html
 Retrieved January 6, 2006,
- Reid, Liam (2005), *HSE to suspend second computer project*, *Irish Times* 6 October 2005
<http://www.ireland.com/newspaper/front/2005/1006/2636179300HM1COMPUTERS.html> Retrieved 4 March 2006
- Reid, Liam (2006), *Renewed pressure to abandon electronic voting*, *Irish Times* 6 October 2005
<http://www.ireland.com/newspaper/front/2006/0428/1190312100HM1VOTING.html> Retrieved 4 August 2006
- Richards Ian, (2002). *Leading parallel lives: journalism and professional ethics. IPE/AAPAE 2002 Conference*

- Saltzman, Sarah (2005), Application development Giving developers the time to improve software quality at the development stage is more cost-effective than fixing bugs during the testing phase False economies of beat-the-clock coding, *Computer Weekly* July 5, 2005; p 18
- Sarin Cliff, (2005), Don't Let the Software Bugs Bite, *Computer Weekly*, 11/8/2005 pages 34-35
- Schneidewind, Norman F. (1996) Do Standards Improve Quality? *IEEE Software*; January 1996, Vol. 13 Issue 1, p22, 3p
- Scott, Sue M. "The Grieving Soul in the Transformation Process." In *Transformative Learning in Action: Insights from Practice. New Directions for Adult and Continuing Education* no. 74, edited by P. Cranton, pp. 41-50. San Francisco, CA: Jossey-Bass, Summer 1997.
- Shea, Garry, (2006) Better Beta, *Computer World*, January 30 2006
- Shilova, S. O. and Shilov, N. V., (2005) On Mathematical Contents of Computer Science Contests *Proceedings of KAIST International Symposium on Enhancing University Mathematics Teaching* May 12-16, 2005 Daejeon, Korea
- Silicon Republic (2003) *Learning from IT disasters*, 29 May 2003 <http://www.siliconrepublic.com/news/news.nv?storyid=single1387> retrieved 2 April 2006
- Skinner, B.F. (1966) *Operant Behavior: Areas of Research and Application*, Appleton-Century-Crofts,
- Sophocles, (440BC), Ajax
- Sosbe, Tim, (2006) Balancing Act, *Certification Magazine*, Vol 8, February 2, 2006.
- Stackman, H., Eeickson, W. J. and Grant, E. E, (1968) Exploratory Experimental Studies Comparing Online and Offline Programming Performance *Communications of the ACM*, January 1968, pp 3-11
- Stahl, Stephanie (2005) Ethernet Inventor And Software-Quality Visionary Earn Technology Medals, *Information Week* March 14, 2005
- Stern, Nancy B.; Stern, Robert A.; Ley, James P. (2006) *COBOL for the 21st Century* 11th edition, John Wiley & Sons Inc ISBN: 0471722618

- Sturm, Eberhard (2000) Eberhard's PL/I Problem, originally published by IBM in *The PL/I Connection*
<http://www.ibm.com/software/ad/pli/pli1297.htm> (now apparently unavailable) re-published in *The PL/I Newsletter* September 2000
http://www.users.bigpond.com/robin_v/pli-n2.htm (retrieved 2 April 2006)
- Swartz, Nikki, (2005), Employees Not Receiving Critical Training, *Information Management Journal*, March/April 2005, Vol. 39 Issue 2, page 7.
- Turing, Alan Mathison (1949). Checking a large routine. *Paper given by Alan M Turing on June 24, 1949 at the inaugural conference of the EDSAC computer at the Mathematical Laboratory, Cambridge*
<http://www.turingarchive.org/viewer/?id=462&title=01> retrieved April 9, 2006
- Turner, R., (2003), Seven pitfalls to avoid in the hunt for best practices, *IEEE Software* 20, 1, January / February 2003, pages 67–69.
- Uchida, Shinji and Shima Kazuyuki, 2005, An Experiment of Evaluating Software Understand ability, *Journal of Systemics, Cybernetics and Informatics* Volume 2 - Number 6, May 2005.
- Way, Thomas, P., (2006), A Virtual Laboratory Model for Encouraging Undergraduate Research, *Proceedings of the 37th SIGCSE technical symposium on Computer science education* Houston, Texas, USA, ISSN:0097-8418
- Welch, Jack and Welch, Suzy (2006), Keeping Your People Pumped. *Business Week*, March 27 2006 Issue 3977
- West, David and Rostal, Pam (2005). Apprenticeship Agility in Academia, *Conference on Object Oriented Programming Systems Languages and Applications*, San Diego, CA, USA, pp 371 - 374
- Wessner, Laura (2006) College Coders Converge on San Antonio for 30th Annual International "Battle of the Brains" *IBM Press Release* issued Armonk, N.Y. February 6, 2006
- Woit, Denise and Mason, Dave (2000) Enhancing student learning through on-line quizzes *ACM SIGCSE Bulletin, Proceedings of the thirty-first SIGCSE technical symposium on Computer science education* SIGCSE '00, p 367 Volume 32 Issue 1. ISSN:0097-8418
- Zenger, Jack; Folkman, Joe and Sherwin, Bob (2006) Make Learning Stick *Leadership Excellence*; January 2006, Vol. 23 Issue 1, p10-11

10 Appendices

- A. Two answers from the field observation.
- B. Interim report on the field study from the TSB group
- C. The answers from the participants in the field study along with their comments.
- D. A summary of the field study Likert-scale scores, as used in the graphs
- E. ISO 9126 standard on quality

A. Answers to Field Observation

These questions were asked four months after the course completed. They were only asked of the two MSc students

Sent: 18 April 2006 12:19
To: MCGANN, CLEMENT
Subject:

Clement,

Here are the few questions you wanted us to have a look at.

Age: 23

Previous Programming Knowledge: Mostly Java, some C++, assembler, SQL, HTML, php

Previous COBOL knowledge: None

Previous eLearning: None

To what extent did 'answering the WebCT challenges' contribute to your learning? It was good, encouraged re-reading of the book to make sure of answers

To what extent did the WebCT exercises enhance existing learning? The quizzes were a good way to reinforce what we learned through the slides

Was there new learning? If so how much? Yes, all of it was new to me

How would you compare WebCT with conventional education? Being able to do the quizzes in your own time was very beneficial

How much benefit was there in writing COBOL code? A great amount, being able to put into practice what we had seen was a great advantage

Would you attend another course using WebCT? Yes, definitely

Would you recommend the COBOL WebCT course to others? Yes, it was very good.

Thanks,

Sent: 18 April 2006 12:51
To: MCGANN, CLEMENT
Subject: WebCT questionnaire

Hi Clement

They're treating us pretty well so far over here, and the heating is a little better.

Thanks for all your help,

Age: 23

Previous Programming Knowledge: Mostly C++; some Java, Eiffel, assembler, SQL, HTML

Previous COBOL knowledge: None

Previous eLearning: None

Q: To what extent did 'answering the WebCT challenges' contribute to your learning?

They were a good secondary contribution, after writing programs. They highlighted things I was unclear on, reinforced existing knowledge and forced me to learn the theory/terminology to go with the practical knowledge gained by coding.

Q: To what extent did the WebCT exercises enhance existing learning?

They reinforced what I had already learned by going back on it a few days after we had covered it in slides/programs, and acted as a check on what I thought I already knew.

Q: Was there new learning? If so how much?

There was some new learning on the theory side, and in clarifying things, but not a huge amount, as we had already written relevant test programs and moved on.

Q: How would you compare WebCT with conventional education?

It would not be a full replacement but it was a helpful adjunct to conventional education. The quizzes were helpful and the having access to the slides online was also useful (particularly as it meant we didn't have to take notes while going through slides).

Q: How much benefit was there in writing COBOL code?

Questions that required the writing of COBOL code were of equal benefit to the non-code questions, even if the sample programs were necessarily fairly simple. It made sure we knew how to use different verbs and code structures we might have been able to avoid in writing programs, e.g. all the different types of PERFORM, different versions of SUBTRACT with multiple operands, different operand order etc.

Q: Would you attend another course using WebCT?

Yes, it was definitely helpful, though I would be reluctant to do a course which only used WebCT if a full course was available.

Q: Would you recommend the COBOL WebCT course to others?

Yes. Some of the questions were very particular to the book, and there was some repetition of questions within quizzes, but overall it made a good contribution to learning COBOL, by testing and reinforcing existing knowledge. Once I had successfully completed a quiz and checked the answers I got wrong, I was confident about moving on to the next section.

B. Interim report on the field study from the TSB group

Review of InnerWorkings training software

Reviewer Details

Course Layout

There is a problem with the software here. The organisation of the course is very confusing to work with.

You would expect to see an introductory course that would be common to all of the .Net Languages, i.e. an introduction to .Net and OO Principles. From here you would expect to see the course broken down into the various language tracks available within the course. For each language you would also expect to see a menu that would present the course topics from fundamental up to expert level.

The inclusion of special topics such as bridging courses (VB6 to VB.Net) and supplemental courses (SQL a tutorial or Client / Server Systems Architecture – An Introduction) could be of benefit to most students, even as a reminder.

At present the course layout does not appear to be in any particular order and this lead to each member of our group starting work on topics that were not suitable.

Module Layout

The course module layout is missing a very important element. Modules should begin with an optional theoretical section. This would allow the student to become familiar with the concept that the module is related to.

While some principles such as LSP are explained they are not explained in a manner suited to a beginner level student.

Exercise Consistency

The learning experience of the student could be enhanced if the each exercise used a common example application. At present each sub-topic within a module works from a different example application. It would be more efficient to allow a student to become familiar with one application and to build on this with more advanced concepts as they are introduced through each topic.

MSDN References

The use of MSDN references within the training software should be kept to a minimum. While MSDN reference material does provide excellent reference guides this material is only suited to a person who is already proficient with the language and concept being covered. It is certainly not suited to a student of a topic.

Conclusion

This training software does have great potential however it is not suited to a novice developer or a developer migrating to the .Net platform, in its current state.

C. Answers to Field Study Questionnaire

Answer 1 - NCI1

	Years of programming experience					_____
Name:	Years of Visual Basic Programming					_____
		Strong Agree	Agree	Neutral or N/A	Dis- agree	Strong Disagree
I was familiar with Object Oriented concepts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I expected InnerWorkings to benefit my knowledge	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings did enhance my knowledge	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
Entering code using the .NET IDE helped	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I like to receive prompt feedback on my effort	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
The Code Judge did give prompt feedback	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I found the Microsoft help feature helpful	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help via the InnerWorkings email facility	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The InnerWorkings email facility was helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I learnt something new from InnerWorkings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from InnerWorkings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I would use InnerWorkings in the future	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings needs to be improved	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

State how it can be improved

The program allows login to the practice set without internet connection only within the MS visual studio, so to review or read the practice I have to be either on the net or running MS Visual Studio.

The other thing, if I started the program before running the Visual Studio and launched a practice set the program loads MS Visual Studio and tries to logon again, it seems that the program does not recognise that it is already running, and it figures it out by failing to login because it is already logged in.

The other thing is there is no equivalent practice set for both MS Visual Studio 2005 and 2003, so if I have 2005 version of MS IDE I can't do drills which only done in 2003 and not done for 2005.

For beginners there is no enough drills, I needed a lot of work to get to work some of the drills which were for me advance but I did.

Answer 2 - TSB1

Years of programming experience
15

Name:

Years of Visual Basic Programming
7

	Strong Agree	Agree	Neutral or N/A	Dis-agree	Strong Disagree
I was familiar with Object Oriented concepts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I expected InnerWorkings to benefit my knowledge	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings did enhance my knowledge	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
Entering code using the .NET IDE helped	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I like to receive prompt feedback on my effort	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
The Code Judge did give prompt feedback	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I found the Microsoft help feature helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help via the InnerWorkings email facility	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
The InnerWorkings email facility was helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I learnt something new from InnerWorkings	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from InnerWorkings	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I would use InnerWorkings in the future	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings needs to be improved	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

State how it can be improved

Mainly, MSDN should not be used as the help system; it should be replaced by a step by step guide, specifically based around the examples and problems posed by the system.

I didn't use the email help system, but I've never had to use it on any other self paced training systems I've used before. I generally find the turnaround time associated with emailing a tutor stifles the flow of a system and results in you essentially hitting a brick wall until such time as a reply is received.

Answer 3 – TSB2

Years of programming experience

2 Industrial; 6 College

Name: Brian O’Sullivan

Years of Visual Basic Programming 3

	Strong Agree	Agree	Neutral or N/A	Dis-agree	Strong Disagree
I was familiar with Object Oriented concepts	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I expected InnerWorkings to benefit my knowledge	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings did enhance my knowledge	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Entering code using the .NET IDE helped	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓
I like to receive prompt feedback on my effort	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge did give prompt feedback	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>
I found the Microsoft help feature helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓
I helped myself, learning through trial & error	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>
I helped myself, by reviewing other course notes	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>
I got help via the InnerWorkings email facility	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	✓
The InnerWorkings email facility was helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	not used	
I learnt something new from InnerWorkings	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from InnerWorkings	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would use InnerWorkings in the future	<input type="checkbox"/>	<input type="checkbox"/>	✓	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings needs to be improved	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	✓	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

State how it can be improved

1. Amend Course Layout
 - a. Use progression ladders to aid students in what path to take through the software
 - b. Clearly mark beginner modules
2. Amend Module Layout
 - a. Include a theory section
3. Exercise Consistency
 - a. For each module use the same application example. For each new topic in the module add to the application
4. MSDN References
 - a. MSDN references are only useful as a refresher for an already experienced individual. They are worthless to a learner
 - b. The MSDN references should be replaced by more friendly examples

Answer 4 – TSB3

Name:	Years of programming experience _____				Strong Disagree
	Years of Visual Basic Programming _____				
	Strong Agree	Agree	Neutral or N/A	Dis-agree	
I was familiar with Object Oriented concepts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I expected InnerWorkings to benefit my knowledge	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings did enhance my knowledge	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Entering code using the .NET IDE helped	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I like to receive prompt feedback on my effort	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge did give prompt feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I found the Microsoft help feature helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help via the InnerWorkings email facility	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
The InnerWorkings email facility was helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NA
I learnt something new from InnerWorkings	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from InnerWorkings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would use InnerWorkings in the future	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings needs to be improved	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

State how it can be improved

1. Less reliance on MSDN which is only a useful resource if you have lots of time to search possible solutions and are an experienced developer used to using such forums. I imagine this tool will be aimed at developers from a beginner's level onward.
2. Clearer progression path for each course i.e. identify a path for the developer so Beginners/Intermediates and Advance programmers have an idea where to begin.
3. E-mail tutoring is a good concept; however it may not be feasible in the real world to wait if you are stuck on an area. Instructor led courses are hard to beat in this sense. Perhaps an always online instructor/monitor (chat room type facility) would help this situation.

Answer 5 – IPS11

Name:	Years of programming experience				12
	Years of Visual Basic Programming				10
	Strong Agree	Agree	Neutral or N/A	Dis-agree	Strong Disagree
I was familiar with Object Oriented concepts	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I expected InnerWorkings to benefit my knowledge	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings did enhance my knowledge	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Entering code using the .NET IDE helped	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I like to receive prompt feedback on my effort	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge did give prompt feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I found the Microsoft help feature helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help via the InnerWorkings email facility	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
The InnerWorkings email facility was helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt something new from InnerWorkings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from InnerWorkings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would use InnerWorkings in the future	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings needs to be improved	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

State how it can be improved

Clearer instructions on topic breakdown in conjunction with difficulty level

Email facility response times would need to be very quick in order for this to work

Answer 6 – IPSI2

Name: Years of programming experience 7
Years of Visual Basic Programming 6

	Strong Agree	Agree	Neutral or N/A	Dis-agree	Strong Disagree
I was familiar with Object Oriented concepts	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I expected InnerWorkings to benefit my knowledge	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings did enhance my knowledge	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
Entering code using the .NET IDE helped	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I like to receive prompt feedback on my effort	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
The Code Judge did give prompt feedback	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I found the Microsoft help feature helpful	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help via the InnerWorkings email facility	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
The InnerWorkings email facility was helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I learnt something new from InnerWorkings	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from InnerWorkings	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I would use InnerWorkings in the future	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings needs to be improved	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

State how it can be improved:

The Inner workings Computer Based training is an excellent package, something I would definitely use again in the future. However from my personal experience of using in the workplace it would be more beneficial to perform the modules on an allocated training day or training lab, possibly as part of an employee induction process.

It's not something you can start / stop and get back to relatively easy as the modules do take up a fair bit of time.

Thanks for the opportunity to evaluate this,

Regards

Answer 7 – IPSI3

	Years of programming experience	_2_			
Name:	Years of Visual Basic Programming	_2_			
	Strong Agree	Agree	Neutral or N/A	Dis-agree	Strong Disagree
I was familiar with Object Oriented concepts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I expected InnerWorkings to benefit my knowledge	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings did enhance my knowledge	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
Entering code using the .NET IDE helped	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I like to receive prompt feedback on my effort	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
The Code Judge did give prompt feedback	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I found the Microsoft help feature helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help via the InnerWorkings email facility	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The InnerWorkings email facility was helpful	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I learnt something new from InnerWorkings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from InnerWorkings	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I would use InnerWorkings in the future	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
InnerWorkings needs to be improved	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
State how it can be improved					

The Judge could give more accurate feedback, pointing you at what has been done badly/incorrectly.

Here's a hastily cobbled together answer, as a general comment I have to say that I found the tool extremely useful but very time consuming, I think the time estimates for each task are too low and as I've been very busy I haven't managed to complete a full module but I would certainly like to continue using it (when I get a chance).

Answer 8 - IPSI3, later

Using the email help facility means waiting forever. I prefer immediate feedback

I use trial & error. I prefer to try 5 different things, rather than waiting

I used it at home. I'm enthusiastic about it. I did this for my own advancement

I missed the theory. It does not include any explanations on theory.

The fourth challenge was way too hard and lacked explanations

Other exercises required less knowledge theory

I was initially lost

I expected an interactive course and then lost track

Start with easy concepts and then more complicated

Every exercise was different - added familiarisation overhead; rather than using the same base and adding to it

I attended a course in DIT Cork - for eCollege - FAS course done by interactive training figure - for studying .NET - like interactive book and audio listen. It had examples and a multiple choice exam -

It doesn't launch the IDE

Overall impression of the tool: very good, very easy to use interface. All relevant information is readily available and navigation thru the app extremely easy and logical. The integration into Visual Studio is useful.

The task oriented learning is a good approach. The best way to learn of course is by doing exercises rather than just reading course material. There was a good range of tasks, some quite challenging for a .NET beginner though.

The availability of relevant reference material and hints is a good feature.

The only negatives I would have -

- Some of the tasks didn't always work when completed, even when following the problem definition exactly.
- The code judging isn't very flexible. If you don't stick to the suggested approach exactly you fail (and variable names must match exactly what's in the problem definition).
- Some of the practice-sets are a bit too time consuming. I would suggest there are too many tasks / modules in some of the practice sets.

Hope this is of some help -

Answer 9 – IPSI4

Years of programming experience **11**

Name:

Years of Visual Basic Programming **8**

	Strong Agree	Agree	Neutral or N/A	Dis-agree	Strong Disagree
I was familiar with Object Oriented concepts	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I expected Innerworkings to benefit my knowledge	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Innerworkings did enhance my knowledge	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
Entering code using the .NET IDE helped	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I like to receive prompt feedback on my effort	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
The Code Judge did give prompt feedback	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I found the Microsoft help feature helpful	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I got help via the Innerworkings email facility	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Innerworkings email facility was helpful	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I learnt something new from Innerworkings	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from Innerworkings	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>					
I would use Innerworkings in the future	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Innerworkings needs to be improved	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

State how it can be improved

This isn't really a criticism but just an observation: For developers new to .NET there is still a lot of learning required to complete some of the tasks. It required a sizable investment of time (worthwhile nonetheless). Some of the individual tasks took hours to complete, meaning modules could take up to 10 – 15 hours to finish (including reading background material as well as doing the tasks).

Answer 10 - ILA1

Thank you for your evaluation of Innerworkings

Please answer the questions below and add your own comments

	Years of programming experience	<u>11</u>				
Name:	Years of Visual Basic Programming	<u>9</u>				
		Strong Agree	Agree	Neutral or N/A	Dis- agree	Strong Disagree
I was familiar with Object Oriented concepts		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was familiar with .NET		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I was familiar with web technologies		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
I was familiar with XML		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was comfortable with Computer Assisted Learning		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I expected Innerworkings to benefit my knowledge		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Innerworkings did enhance my knowledge		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The challenges were well constructed		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got satisfaction from responding to challenges		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I learnt from the challenges		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
Entering code using the .NET IDE helped		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would have preferred a plain text editor		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
I like to receive prompt feedback on my effort		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
The Code Judge did give prompt feedback		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave accurate feedback		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The Code Judge gave useful feedback		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I found the Microsoft help feature helpful		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, learning through trial & error		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reading the manual		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I helped myself, by reviewing other course notes		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help by asking a friend		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I got help via the Innerworkings email facility		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
The Innerworkings email facility was helpful		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I learnt something new from Innerworkings		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I reinforced existing learning from Innerworkings		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It must be used in conjunction with other education		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
I would use Innerworkings in the future		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Innerworkings needs to be improved		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This method has a future		<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<hr/>						
State how it can be improved						

- = ENVIRONMENT: WORK DISK IS NOT THE PLACE, TOO MANY WORK RELATED DISTRACTIONS
- = ~~THE~~ METHOD ONLY SUITS SOME PEOPLE. DIFFERENT NEEDS, PREFERENCES FOR LEARNING + WORKING STYLES.
- = ORGANISATIONS SHOULD ALLOW TRAINING TIME AND IF LARGE HAVE A LOCAL CODE JUDGE.

I found the Microsoft help feature helpful	4	2	2	1
I helped myself, learning through trial & error	4	3	4	5
I helped myself, by reading the manual	4	3	2	2
I helped myself, by reviewing other course notes	2	4	2	5
I got help by asking a friend	2	2	2	2
I got help via the Innerworkings email facility	2	2	1	1
The Innerworkings email facility was helpful	3	3	0	0
I learnt something new from Innerworkings	4	3	5	4
I reinforced existing learning from Innerworkings	4	3	4	4
It must be used in conjunction with other education	4	4	4	5
I would use Innerworkings in the future	4	3	3	3
Innerworkings needs to be improved	3	4	5	5
This method has a future	4	4	3	5

2	4	2	5	4	4
4	4	4	5	4	5
2	4	2	5	4	4
2	2	2	4	4	4
2	3	2	3	2	4
2	4	4	1	3	1
3	4	4	3	3	3

4	4	4	5	5	4
4	4	4	5	5	5
4	4	2	5	3	5

4	4	5	5	5	5
4	4	4	3	3	5
4	4	4	5	5	5

B Field Study Responses

Table 18 - Summary of Field Study Responses

	NCI1	TBS1	TSB2	TSB3	TSB4	IPSI1	IPSI2	IPSI3	IPSI4	ILA1
Years of programming experience	7	15	7	8	12	5	2	7	11	11
Years of Visual Basic Programming	7	7	7	3	10	5	2	6	8	9
I was familiar with Object Oriented concepts	4	4	4	5	5	3	4	5	4	5
I was familiar with .NET	2	4	5	4	5	3	2	5	4	2
I was familiar with web technologies	2	5	4	4	4	4	4	5	5	2
I was familiar with XML	2	4	4	2	4	4	4	5	4	4
I was comfortable with Computer Assisted Learning	4	4	5	5	5	3	4	5	5	5
I expected Innerworkings to benefit my knowledge	3	4	5	4	4	3	4	5	5	4
Innerworkings did enhance my knowledge	4	3	4	4	4	4	4	5	5	4
The challenges were well constructed	2	3	3	3	3	4	4	5	4	4
I got satisfaction from responding to challenges	4	4	3	4	4	3	4	5	4	5
I learnt from the challenges	4	4	4	4	4	4	4	5	4	4
Entering code using the .NET IDE helped	4	5	5	5	4	5	5	5	5	5
I would have preferred a plain text editor	2	1	1	1	2	1	1	1	1	1
I like to receive prompt feedback on my effort	3	4	4	5	5	4	5	5	4	5
The Code Judge did give prompt feedback	4	4	3	4	3	5	4	5	4	3
The Code Judge gave accurate feedback	4	4	3	3	3	3	3	5	3	3
The Code Judge gave useful feedback	3	4	3	2	3	3	2	5	3	3

E The ISO 9126 Standard – Definition of Quality

The ISO 9126 Standard has a definition of quality, entitled 'Software Engineering – Product Quality'. In addition to its 'quality model', ISO 9126 propose three metrics for measuring quality: External, Internal and 'in use' metrics (ISO 9126)

The ISO 9126 definition is:

Functionality -



- Suitability
- Accuracy
- Interoperability
- Compliance
- Security

A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

Reliability



- Maturity
- Recoverability
- Fault Tolerance

A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

Usability

- Learn-ability
- Understand-ability
- Operability

A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

Efficiency

- Time Behaviour
- Resource Behaviour

A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

Maintainability

- Stability
- Analysability
- Changeability
- Testability

A set of attributes that bear on the effort needed to make specified modifications.

Portability

- Install-ability
- Conformance
- Replace-ability
- Adaptability

A set of attributes that bear on the ability of software to be transferred from one environment to another.

- END -