PREPARING FOR THE DIGITAL BATTLEFIELD MODELLING TRAINING FOR C^4I SYSTEMS

Major Chris Lewis-Cooper AGC(ETS)
Officer Commanding Training Systems Group
Directorate of Individual Training Policy (Army)
Upavon, Wiltshire, United Kingdom

ABSTRACT

Computers are improving in power, speed and affordability by an order of magnitude every five years. Thanks partly to parallel improvements in miniaturisation and ruggedization, the use of this speed and power in C^4I systems on the battlefield seems set to increase dramatically in the next few years. In spite of received wisdom about children's familiarity with computers, there is no evidence of any equivalent improvement in the ability of recruits to operate these systems. If this lack of ability is not to become a limiting factor on the "Digitized Battlefield", an affordable, dependable and practical training programme for C^4I systems is urgently needed.

Training for C^4I systems inevitably involves extensive use of computers as training devices. The widespread use of Computer-Based Training (CBT) and Distributed Training (DT), possibly embedded in operational C^4I systems, will be essential in future to combat the twin scourges of skill-fade and rapid version upgrades for large, highly distributed user populations. Synthetic Environments (SE), of varying degrees of abstraction, will need to be incorporated within most, if not all, stages of such training. But C^4I systems, unlike weapon systems and vehicles, tend to be developed using Rapid Applications Development (RAD) techniques. The use of RAD means that "design freeze" may occur after roll-out or may actually never occur at all. The long lead-times usually associated with CBT, DT and SE design and production are inconsistent with such rapidly changing requirements. At the same time, the costs and risks associated with the development of CBT, DT and SE make some form of rapid yet rigorous justification process highly desirable.

Thanks largely to the emerging standardization of computer user interfaces, it is proposed that a generic model of C^4I systems training is now feasible. By adopting a scaleable default training solution at the outset of any C^4I project, a strategy of modifying such a model as the main project develops is likely to be more responsive than the current strategy of starting from "scratch". It should also provide a reasonable initial cost estimate for training, a feature missing from most current C^4I system requirements. Such a model has the added advantage that best practice could be incorporated incrementally, refining it over time. In this way, much of the analysis and design process could be re-used, thereby becoming both faster and more efficient.

ABOUT THE AUTHOR

Major Chris Lewis-Cooper has served in the Royal Army Educational Corps and its successor, the Adjutant General's Corps (Education and Training Services) since 1977. He achieved his first Masters degree in Educational Technology at Bath University, England, in 1986. Since then he has served as a training development specialist with the Royal Engineers, a regional education staff officer, the senior recruit selection officer for the Army and the desk officer in charge of Army computer-based training policy. During recent a two year exchange posting to the Canadian Forces, he was responsible for the development of training for interactive courseware design teams. He has experience of a wide range of major computer projects, both as a project manager and as a policy maker. He has presented papers on computer-based training and simulation at NATO conferences, IMTA, ITEC and several other international forums. As a soldier, Chris conducted pre-Sandhurst training for officers, saw active duty in Northern Ireland and has served in places as far apart as Canada, Belize and Switzerland.

Chris is currently serving as the Officer Commanding the Training Systems Group at Upavon, Wiltshire, England; the Headquarters of British Army Training. His Group is responsible for advising the Army on Best Practice for all aspects of technology-based training analysis, design and delivery. He is also studying for his second Masters Degree in Modelling and Simulation at Cranfield University.

PREPARING FOR THE DIGITAL BATTLEFIELD MODELLING TRAINING FOR C^4I SYSTEMS

Major Chris Lewis-Cooper AGC(ETS)
Officer Commanding Training Systems Group
Directorate of Individual Training Policy (Army)
Upavon, Wiltshire, United Kingdom

INTRODUCTION

Training Support Branch, part of the Adjutant General's Personnel and Training Command, is responsible for the quality assurance format and procedures associated with Training Needs Analysis (TNA) and subsequent training systems design and implementation for the British Army. Recent developments in "smart procurement"[1] prompted a study into the ways in which TNA interacts with mainstream procurement methods, in particular with Integrated Logistics Support.

This study[2] recognised that, while the products of TNA[3] are likely to be similar for all projects, the procedures for obtaining these products might be very different for certain classes of project. One such class includes projects where the main user-training requirement is directed at the use of computer software. Such projects are referred to as Software Intensive Projects (SIP). The Director of Individual Training Policy (Army) subsequently commissioned a follow-on study of TNA in SIP in February 1999. This paper covers the preliminary findings of that study with particular reference to training for battlefield, Command, Control, Communication, Computer and Information (C^4I) Systems.

TRAINING THE "DIGITAL WARRIOR"

The British Army's fifteen-year initiative for Digitization of the Battlefield (Land) (DBL) will cost at least £10Bn. Stage One of this programme is in the implementation phase. Stage Two includes the future communications system BOWMAN, a series of infrastructure projects such as the Formation Battlefield Management System, Battlegroup Management System, geographic and other database applications and a raft of 17 Battlefield Information System Applications (BISAs). Experience from Stage One[4] indicates that training for these systems will present the biggest challenge faced by the Army training organisation since the Second World War.

Training for any computer application inevitably involves the use of computer hardware and software, either simulating the real application or, where appropriate, using the real system. Most training on battlefield C^4I systems also requires some degree of simulation of team members, opposing and friendly forces, weapon effects, terrain, weather etc. Increasingly, training computers are also being required to simulate some of the instructional roles such as task setting, training performance measurement and training information management. Building a training system for C^4I is, by consequence, a complex software project in its own right.

It has been noted elsewhere[5] that conventional TNA lifecycle management may have lessons to learn from software engineering lifecycle models. Because of the software-intensive nature of C^4I training, it was considered that other analysis techniques from software engineering might be applicable to the business of TNA. If this turned out to be the case, the TNA process itself would benefit and the similarity of methods and terminology would also improve communications, and hence integration, between training and the main C4I project staff.

MODELLING AND SYSTEM DYNAMICS

Training for operational systems is rarely a one-off event. Examination of training for existing C4I systems revealed that various forms of training are delivered on as few as three and as many as nine occasions during preparation for operations. This examination also revealed poor consistency in the terms used to describe these stages of training and little consideration of the cumulative interaction between them. This confusion is largely due to the complex organisational issues surrounding responsibilities for, and funding of, the various stages.

Systems Dynamics provides a set of useful tools for modelling complex systems involving human policies and activities. In particular, these tools are suitable for illustrating the management of continuous flows such as are found in training "pipelines"[6]. Applying these techniques to existing C4I training systems resulted in a generic C4I training pipeline (Figure 1).

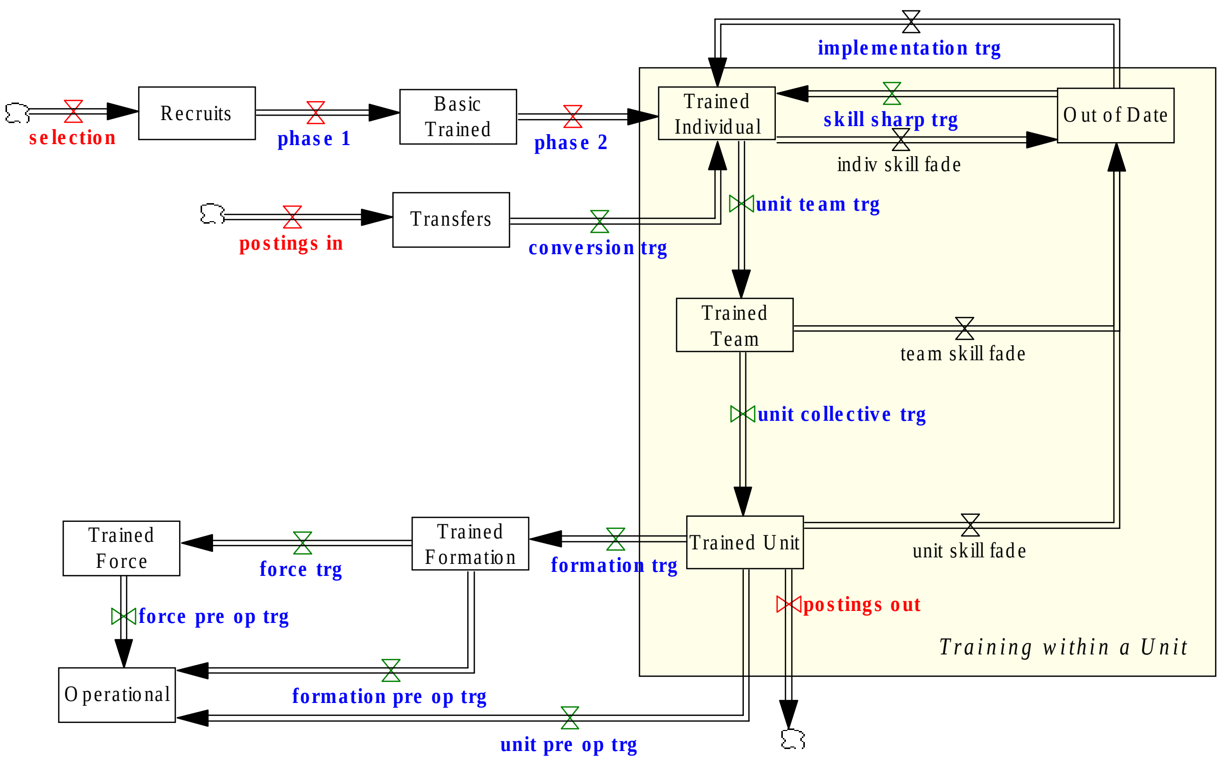


Figure 1 – Generic Training Pipeline for C4I Systems

Using this model, it was possible to explain several key points of similarity between training systems for C^4I projects. One such point is the tendency to focus a project's training analysis resources on providing implementation training. The model shows that, while this is the "crocodile nearest the boat" for C^4I project managers, it is by no means the only concern. Implementation training designed as part of the main project is unlikely to be suitable for normal, steady-state training in Phase 2 because it has a different functional purpose. For the same reason, it is almost certainly unsuitable for the in-unit "skill-sharp" training necessary for countering skill-fade and addressing version upgrade training. In most cases, implementation training leaves the questions of Phase 3 Crew and Collective training unanswered.

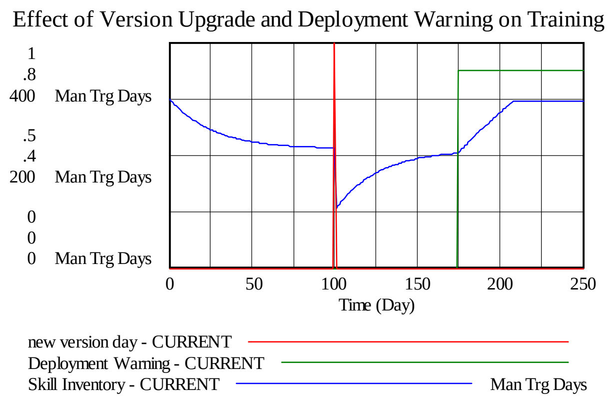
Another insight gained from Figure 1 was that the existing training delivery methods used in each stage of training have more in common with other C^4I projects at the same stage than with the same project at other stages. This raises the interesting possibility of establishing a common training solution at each stage across all C^4I systems, solving at a stroke the endemic problems of stove-piping and lack of training co-ordination found in existing training provision. Inevitably, there are substantial organisational and financial barriers to such a proposal, not the least of which is that project-sponsored TNA studies are classically directed at training for equipment, rather than role or capability.

A third insight is the importance of the so-called "skill-sharp" training after a trainee arrives at a Field Army unit. As we will show in the next section, without this corrective feedback loop, the combination of skill fade and new software versions ensures a very rapid degradation of unit-level operational capability. Another valuable use of this feedback loop could be to collect empirical evidence of both the nature and the extent of skill-fade, through the use of comprehensive pre and post-testing. Pre-testing would also enable a tailored, just-in-time training response, resulting in major time and cost efficiencies and improved overall training system performance. Currently, training management structures do not support individually tailored training responses or detailed information feedback for existing skill-sharp training.

SIMULATION AND ESTIMATION

Managing an Army-wide, role-based, training pipeline based on Figure 1 and catering for all C^4I training would involve substantial investment. The expectation of high levels of technology-based and distributed training would require not just expensive hardware delivery platforms but also centralised simulation and computer-based training design assets, closely linked to appropriate standards agencies. The prospect of annual software upgrades[7] would demand the close integration, even collocation, of these agencies and the main C^4I configuration management teams. Such a large re-organisation requires detailed justification and risk assessment. At present, there is little empirical evidence to support this strategy.

Fortunately, Systems Dynamics can contribute to the justification of such a re-organisation of Army training management. Based on the model in Figure 1, it is possible to conduct quantitative "what if" experiments on parts of the model, for example examining optimum unit training policies for coping with new versions of BISAs and warnings



Explanatory Notes:

1. Notional Unit establishment of 400.

2. Continuous flow training (Unit-Based Trainer).

3. Skill Fade factor of 1% per month of skill inventory

4. New User Interface on day 100 (version severity 0.5) (recovery circa 50 days).

5. Return from Operations on Day 1, Deployment Warning on Day 175.

Figure 2 – Example Quantitative Skill Inventory Model of a Notional Field Army Unit

for operations(Figure 2). Such detailed models can be used to evaluate various policies and other influences on trainee flow for existing training systems, calculate re-scaling factors for new training delivery and estimate costs. Of equal importance, simulation of the model can be used to estimate such factors as the training readiness dates throughout the system necessary to support a given operational readiness date.

The graph in Figure 2 was produced using a System Dynamics model representing a notional Field Army Unit, of strength 400, on returning from operations. The graph represents the effects, over 250 days, of a steady skill-fade rate, introduction of a new User Interface on Day 100 and a New Deployment Warning on Day 175. Other influences used in the model include the capacity of a proposed Unit-Based Trainer and the training policy of the Commanding Officer.

MODULAR DESIGN AND RE-USE

Arguably the most important advance in software engineering to date has been the concept of re-use. The creation of general-purpose modular designs, with highly specified functions and interfaces with other modules, releases software designers from the tyranny of the blank sheet of paper and increased productivity by several orders of magnitude.

By contrast, and in spite of attempts by the academic community to achieve this for training systems[8], there is no widely recognised repository of such modules for general training. Still less is there a systematic approach to verifying and validating the design of such modules against real training outcomes. Some proprietary templates for multimedia training do exist[9], but most are based on content or organisation-specific designs, rather than training function. Interestingly, a large proportion of these templates is directed at training for software applications. Unfortunately, they are not normally in the public domain and are rarely documented in a modular fashion suitable for re-use. Trainers traditionally seem to prefer custom-built solutions to “off-the peg” training.

The “object-orientated” design method used by software engineers calls for layers of design at increasing levels of detail. The aim is to identify functional modules that are highly complex and cohesive internally, but interact (or couple) very simply with other modules at the same level. This allows such modules to be assembled quickly and simply into a system.

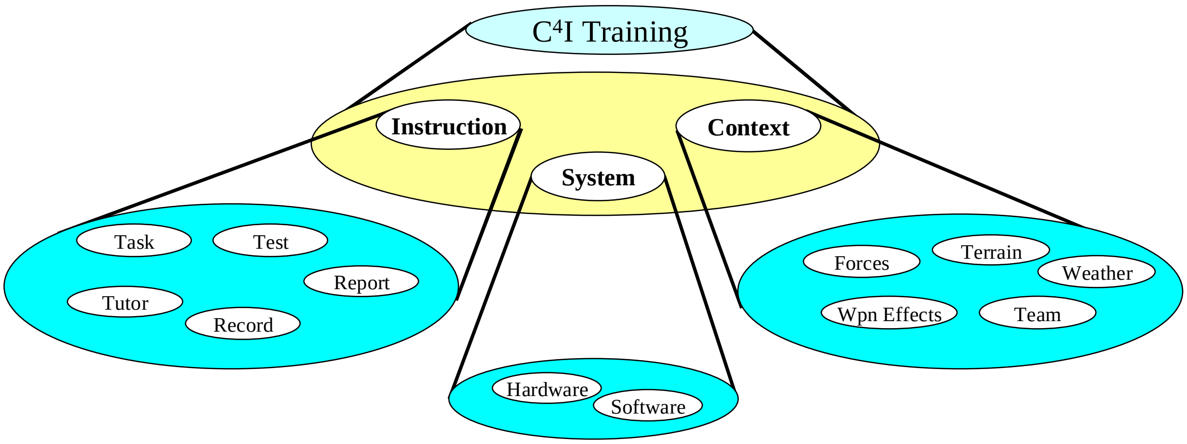


Figure 3 – High Level Design for a C^4I Stage Training Module

Applying these principles to the design of a general-purpose C^4I training module produced the high-level design given in Figure 3. While it is recognised that this is just one of many possible designs, this C^4I design leads to considerable simplification of the training options analysis process, a core function of TNA.

Each of the sub-modules in Figure 3 may be decomposed into further layers of detail. For example, the System Module software may comprise a set of simple screen captures of the real application, a custom-built emulation of the application, or possibly the real application itself. Likewise, the hardware may be operational, based on a vehicle or weapon system, or cheaper and more convenient desk-top training hardware. As was noted above, existing decisions for operational C^4I systems start to look remarkably similar for each stage of the training pipeline, particularly regarding hardware options.

The Context module comprises a set of synthetic environment databases, each providing a specific simulation component, for example terrain, opposing forces or weapon effects simulation. If the real system is being used for the System Module, it may need to contain a security module to ensure that simulation data cannot be mistaken for real data. If an emulation of the real system is being used, the Context Module may become quite trivial. Even for a simple screen capture, however, the synthetic data shown should be consistent with the training task and carefully considered as a discrete training design task. At later stages of training, generating context becomes increasingly complex and expensive. Fortunately, a high degree of contextual commonality across different projects at the same stage of training should allow substantial common usage of context modules.

The Instructional Module may be relatively simple, particularly if a human instructor is used, since most decision-making can be made in real time. Common tasking, guidance and information management procedures for human instructors are well developed in a training school environment. However, battlefield skills are rarely in constant use and opportunities for frequent practice are highly desirable. The rapidly evolving nature of most software makes this need for frequent practice imperative as part of upgrade training. In most cases, this means providing on-demand training in Field Army Units where human instructors are at a premium.

Training for C^4I systems can be expected to include a high degree of distributed training, making the use of automated instructional techniques essential for reasons of standardisation and economy. Developing a range of general-purpose, re-usable instructional strategies, for example based on the work of Merrill[10], could have a dramatic effect on future training development costs. These instructional strategies would interact with the task list to drive one or more training systems and their associated training Management Information Systems.

One interesting effect of adopting the training module design in Figure 3 is a potential modification of the Training Objective (TO) structure. By integrating the TNA output data with the software design, the former becomes virtually self-documenting. The tedious task of updating conventional TOs is thereby avoided. Given this design, a TO could read "Given the <System Module>, and in the context of <Context Module>, carry out the <Instructional Module (tasks and standards)>.

The top-down design approach in Figure 3 is still under development. Much of the discussion concerns the degree to which this design can be generalised at lower levels. The Instructional Module, for instance, reveals functional commonality between instructional roles at relatively low levels. The Context Module may be more project-specific for some stages. The objective of this on-going study is to provide a general model of a training system that can be re-used with minimum modification to suit any specific C^4I training need.

LIFE-CYCLE MANAGEMENT

An area of interest to both project manager and training systems analyst is that of life-cycle process control. For qualitative aspects of training, management is conventionally considered in two phases; training systems development (specifically TNA, design, development and implementation), and through-life maintenance and configuration control using a simple feed-back loop called the Systems Approach to Training (SAT).

TNA studies currently in progress for the British Army use a life-cycle model similar to the "waterfall" model used by software engineers in the 1970's[11]. This involves sequential stages with backward iterations through one or more stages at each step. Commercial firms often use a more incremental approach, with several modular stages being developed slightly out of step to even out resource usage. The Royal Navy has recently suggested the use of the more sophisticated V-model approach to life-cycle management for TNA[5]. So far, it seems that the latest prototyping or hybrid approaches used by modern software engineers have not been applied to "training systems engineering".

Prototyping involves the rapid development of a series of working prototypes. Such life-cycle models are designed to be responsive to the kind of rapidly evolving requirements experienced in C^4I projects. Typically, libraries of re-usable, general-purpose modules are "bolted" together very quickly and at relatively low cost. The aim is to provide basic, functional examples to the customer around which the final requirement can be articulated. In some cases, these working prototypes evolve to become the actual system, although this practice is not generally recommended since the evolutionary development method usually generates design inefficiencies.

Conventional TNA procedures, by contrast, go to considerable lengths to avoid considering training options before a detailed requirements specification is produced. The aim is a highly specific custom-built training response. This is intended to avoid prejudicing the front-end analysis in favour of solutions. In the case of C^4I systems developed using rapid prototyping, a TNA developed like this will inevitably arrive too late. Further, for C^4I systems the training options at each stage appear to be similar, differing only in the systems software and tasking sub-modules. If many components of the training design already exist in generic form, the influence of a detailed front-end analysis is less significant. It seems that there may be considerable scope for the development of a rapid prototyping approach to the design of C^4I training systems.

As a general principle, it is proposed that the life-cycle of a TNA study for C^4I systems training should conform, where possible, to whatever management life-cycle is being used by the main project. For safety-critical training systems such as aircrew training, for example, a formally validated V-model may be the only acceptable option. For most C^4I systems, however, adopting the training pipeline system in Figure 1 as a pan-Army system would allow system components based on Figure 3 to be used to develop generic prototype models for training delivery at each stage of the pipeline. This would allow more effort to be allocated to task analysis and Target Audience Description. In this way, a default prototyping or hybrid methodology similar to Rapid Applications Development could be used for TNA process control.

TRAINING SYSTEM MAINTENANCE

The post-roll-out phase of Life-Cycle management, training system maintenance, is conventionally managed by means of testing and external validation, two components of the SAT cycle. These correspond roughly to the validation and verification procedures used by software engineers. Bhoem[12] characterises verification as the process of “building the thing right” and validation as “building the right thing”, both procedures being normally applied before roll-out in software engineering.

The SAT specifies verification (testing) for through-life training systems maintenance and validation (or external validation) procedures for change and configuration control. Testing trainees at the conclusion of a stage of training is normally used, amongst other things, to check that the system is training to specification i.e. trainees can perform the tasks required, to the specified standard and under the specified conditions. This may be interpreted as “stage” system verification.

Since the “acid” test of a training system is on-the-job performance of a trained soldier, validation is only practical some considerable time after roll-out. Validation of the training system is normally carried out at discrete intervals, commonly every other year, by sampling the job performance of qualified trainees. Unfortunately, recommendations are often overtaken by events and the procedure is frequently starved of resources. C^4I systems typically have a rapidly evolving doctrine of use and with software version upgrades expected annually, conventional external validation is hopelessly unresponsive to such a high rate of change. A cynical project manager might add that any management system with a periodicity longer than the average sponsor's tour of duty is unlikely to attract much support or enthusiasm.

The later stages of the training pipeline in Figure 1 have limited mechanisms for identifying specific failures in earlier stages and only primitive systems such as After Action Reviews to identify performance failures in the current phase (a procedure referred to in SAT as "internal validation"). Information feedback to check earlier stages in the training pipeline is also problematic, depending mostly upon anecdotal evidence of major failures. Since most of a soldier's career is spent in these later stages of training, this means that overall training pipeline verification is very crude, by the standards considered acceptable in software engineering. Pre-testing before a stage is rarely applied, except during initial recruit selection, and what results do exist tend to be used for simple screening. Systematic validation of later training stages is frequently reduced to a single post project evaluation exercise.

Both validation and verification of training systems depend upon measurements of human performance. Since C^4I training and operations involve human/computer interactions, there is a real opportunity to automate systematic data collection on training and subsequent job performance in support of both verification and validation. Comprehensive and interlocking pre and post-tests for every stage of training, and the analysis of the data produced by the training standards authority, could be used to provide far more responsive and detailed verification, validation and training system management procedures than those currently available using conventional SAT procedures.

IMPLEMENTATION

Defining TNA in terms of "training systems engineering" calls for an initial phase where existing documents are re-formatted to reflect the new approach. Starting with Stage 2 DBL projects, a single training specification document might be produced for each role connected with DBL. This document provides details of all stages of training, from recruit selection through to pre-operational training, including all C^4I systems associated with that role. The first chapter contains current selection criteria, and references to previous versions, including the detail of a standard Target Audience Description. At each subsequent training stage, the specification could be developed as outlined in Table 1, thereby integrating task and training options analysis.

Subsequent C^4I projects should benefit considerably from this initial phase. All that is required from later projects, by way of initiating TNA, is to identify those roles likely to use the system. By examining the existing Training Specification Document for that role, TNA becomes a matter of specifying the modifications necessary to integrate the new BISA training into the role. The lessons learned from previous training designs and, more importantly, the integration of the various DBL training systems, are immediately obvious to the training analyst. Moreover, changes to the structure and management aspects of the training pipeline can be based on sound principles of feedback and control.

Training Stage	*Location*	*Duration (& MTD)*	*Tasks*	***Training System Specification***		
				System	**Context**	**Instruction**
Phase 1	Army Training Regiment	11 weeks (5 days)	Computer Literacy	PC (Net) + OA s'ware	Nil	Instructor + Commercial Packs
Phase 2	Royal School of Signals	24 weeks (16 weeks)	T.O. 1-67	PC (Net) + BISA Emulator	Emulator	Instructor/ CBT
Skill-sharp	Field Unit	Continuous (12 days/yr)	All	PC(Net) + BISA	GP3/QP24 Stimulation	CBT/Field Tutor
Crew	Field Unit	2 weeks (-)	T.O. 68-92	PC(Net) + BISA	Stimulation database	Instructor + CMI
Collective	CATT/HFT	9 Days (-)	T.O. 92-104	Real	Live Sim/ ABACUS	Hi/Locon + CMI
Pre-op	In theatre	4 Days (-)	T.O 104-164	Real	Live Sim/ ABACUS	Hi/Locon + CMI

Table 1 – Outline Training Specification Document – BISA Data Entry Clerk

Availability of training is a key aspect of managing training within the Field Unit. Cost-performance improvements in distributed training techniques for C^4I mean that the costs of a computer-based Field Unit trainer are probably sustainable, once the need is firmly established[13]. What is less certain is that the management structures necessary to maintain such a system are achievable.

Controlling the training system pipeline using training and operational system performance data requires specialists, both for designing the Field Unit training systems and for human performance aspects of the real application design. Testing also carries a significant time penalty for trainees and their Commanding Officers. Trained standards staff would be required to collect and analyse the data to provide timely and comprehensive training management reports. Although such reports would allow significant improvements in the overall management of training, it is not certain that resources will be made available for this enterprise on a wide scale. It is likely that the value of such information will be assessed using a limited trial, once a Unit Based Trainer capability is in place.

CONCLUSION

At no point during this paper is it proposed that the basic principles of TNA, as given in the tri-Service Guide[3], should not apply to C^4I training systems. On the contrary, C^4I is at the heart of the Army's core function, making it imperative that training is both efficient and effective. We are confident that TNA still represents the best guarantee of achieving this.

What is explored here is the potential for techniques commonly in use for software engineering to assist in producing the TNA milestone deliverables in a new, more useful format. In particular, Systems Dynamics modelling has been proposed as a mechanism for standardising and harmonising high-level designs across many C^4I projects. Object-orientated design principles are suggested as a means of encouraging the re-use of training design modules and project life-cycle models are proposed to help manage the process.

So far in the TNA in SIP Study, most of the emphasis has been on modelling C^{4}I training system requirements and assisting design. Little has been said about methods for defining Target Audience Descriptions or producing task analyses. First impressions are that these key TNA deliverables are also very similar across C^{4}I projects and that a high degree of re-use is possible. Consideration will be given to these deliverables in the final report, due in late 1999.

References:

[1] The Strategic Defence Review: Supporting Essay 10 "Procurement and Industry"
[2] D/AG/11000/12/32L0 (DP&RG) Report No 1/98 "Integration of TNA and ILS Procedures" dated 31 Mar 98
[3] "A Tri-Service Guide to Training Needs Analysis for Procurement Projects", TSTSWG Version 1 Nov 98
[4] Admiral Stage 1 DBL TNA Report, Jul 97.
[5] Cook C.B. and Bowden M.N. "The Functional Specification and its Uses in the Evaluation and Acceptance Process", Conference Proceedings, ITEC '99
[6] Clark R. "System Dynamics and Modelling", Institute for Operations Research and the Management Sciences, 1988.
[7] DLD Paper (in draft) "The Frequency of Upgrade of Battlefield Information Systems Applications for Stage 2 Digitization" dated 29 Jan 99.
[8] K.Van Marcke "Learner Adaptivity in Generic Instructional Strategies" Utah State University Summer Institute 1996.
[9] R.H. Blalock "Using Development Shells for Fast and Creative Multimedia Development at American Airlines" Society for Applied Learning Technology Conference Proceedings, Feb 95.
[10] M.D. Merrill "Instructional Design Theory" Educational Technology Publications 1994.
[11] Royce, W "Managing the Development of Large Software Systems" Proc WESTCON Ca, USA,1970
[12] Bhoem "Management of Software Systems", Prentice Hall, 1975
[13] DGD&D "The Army Training Policy for Digitization" 1* Circulation Jun 99.