TECHNICAL MEMORANDUM
WSRL-TM-3/90

RAPID PROTOTYPING FOR COMBAT SYSTEMS - A RESEARCH TASK

A.P. Gabb

SUMMARY (U)

This paper describes the requirements and activities for a research task relating to the development of facilities for the rapid prototyping of operator interfaces in combat systems. Emphasis is placed on requirements analysis, simplicity of modification, and portability of the prototype.

Author's address:
Combat Systems Division
Weapons Systems Research Laboratory
PO Box 1700, Salisbury
South Australia

Requests to: Chief, Combat Systems Division

TABLE OF CONTENTS

1. INTRODUCTION

The need for prototyping in any system involving complex operator interfaces is now generally accepted, and is commonplace in the development of custom commercial software packages. Its main advantage is that the user and developer can reach a mutual understanding of the requirements for the product, and at a far higher level of detail than is possible by other means (ref.1).

There are benefits for both parties. The user is given an early example of what he is buying, and can then see deficiencies in his specified requirements or in the proposed implementation. The developer on the other hand can recognise where his understanding of the requirement is incorrect or incomplete, and take early corrective action, avoiding the development of an inferior product.

The reasons why rapid prototyping is not used in the development of some systems which would seem to benefit from it are complex, and include the possibility that the developer may see no direct benefits to himself. If the developer has a requirements specification for a product and a contract to produce it, he can meet the letter of the requirement without using prototyping. The system will probably be usable, but may require significant modification before it is optimal. (In this case the delivered system *is* the prototype.) The fact that the total effort, cost and time required could be much larger if prototyping were not employed may be inconsequential to the developer, or may even be seen as a benefit in terms of company profits.

It is evident, therefore, that the user has the most to gain from prototyping, and must take the lead in requiring it in his projects.

1.1 Aim

This paper describes the requirements and activities for a research task relating to the development of facilities for the rapid prototyping of operator interfaces in combat systems. Unlike some forms of prototyping, the intention is not necessarily to evolve the prototype into the final system, but to use it to define or refine the requirements for the system.

In this paper the set of facilities is referred to as the Rapid Prototyping Environment (RPE).

In Phase 1 of the task, preliminary development and investigations were carried out to reduce risks and form a basis for the major elements of the research task - Phases 2 and 3. The Phase 1 findings are briefly described and the activities for Phases 2 and 3 are defined.

1.2 Rapid prototyping in combat systems

The need for prototyping in the development of combat systems can be seen from even a superficial review of existing systems. In most cases the operator interfaces are far from optimal and the effects of this are quite serious:

- The systems are used inefficiently, resulting in a combination of excessive operator workload and an impaired response in critical situations.

- Many functions for which the system was designed are not used, either because of the complexity of the interface to use those functions, or because the functions, although implemented as specified, do not meet the true requirements of the user.

- More training is required than should be necessary, to cater for features which may not work correctly or which are complicated and difficult to use.

Most of these problems can be avoided by prototyping. The fact that they have not been rectified many years after the first system has been developed is a consequence of the cost of modifying a completed system, coupled with a reluctance on the user's part to make major changes which will result in impacts to established maintenance and training programmes. In addition, the user is obliged to make the most of the system (he *has* to use it) and will often become blind to deficiencies which were obvious on first sight. Often only critical problems are corrected.

There are other less obvious advantages to prototyping:

- The development of requirements is usually undertaken by potential users with the assistance of system consultants skilled in requirements analysis and complex systems design. The users are experts in the operational requirements and the consultants use this knowledge to develop a requirement which is suitable for specification and implementation. Usually the exchange of information is not complete and there are deficiencies both in the consultants' understanding of the requirement and the users' model of the proposed system. A prototype completes the exchange with the users seeing what is proposed and the consultants seeing how it will be used.

- Most users who are contributing to the definition of requirements for a new system are not familiar with current operator interface techniques and this restricts their ability to contribute adequately to the definition. Typically, such users are experts on the use and technology of the system being replaced, which may have been designed twenty years before. A prototype allows them to obtain experience with the new technology and see the advantages which it can provide.

- Even using modern design and specification techniques it is extremely difficult to specify a complex system, particularly with regard to correctness, consistency and completeness (ref.2). A prototype provides a visibility which identifies many specification faults, and highlights areas which are often poorly specified, such as error handling, default system states and the interaction between different modes of operation.

- A prototype helps to stabilise the requirement at an early stage, and hence reduce the significant costs involved when requirements change during the project (ref.3).

2. BACKGROUND

Combat Systems Division is responsible for maintaining a technology base relevant to the design, development and analysis of combat systems. Relevant activities and research tasks include the following:

- Requirements analysis for proposed combat systems
- Evaluation of contending systems
- Monitoring of development, testing and trials
- Modification and enhancement of existing systems
- Software development tools for real time processing
- Rapid prototyping for combat systems
- Distributed processing and integration techniques
- Display technologies for combat systems
- Human factors research.

The Rapid Prototyping Environment will provide a basis for enhancing many of these activities, as well as a set of tools to assist in the furthering of specific research areas.

3. OBJECTIVES

3.1 Prime objective

The prime objective of this task is to be able to develop a prototype for a single combat system workstation in one to three months. The prototype will include representative displays and controls for the station itself, and sufficient simulation of other functions of the system to provide the normal modes of operation for the station.

Specific attention is to be paid to system performance, particularly with regard to the response times for display changes as the result of operator actions.

Displays and controls are to be as representative as possible, depending on the aims of the prototyping application.

Where possible, the definition of the displays and controls is to be in a form suitable for inclusion in a requirements specification. It is not acceptable that the prototype itself is the definition of requirements. It should also be possible for the users to understand and modify these definitions with minimal training.

The prototype should be portable, preferably using equipment which is likely to be available at the user's and developer's locations.

3.2 Assumptions

It is assumed that the workstation will include one or more of each of the following (possibly in combination):

- A labelled plan display (LPD) of the area of operations.
- An alphanumeric display (Tote).
- A position entry device (PED) such as a mouse, rollball or forcestick.
- A keyboard.

- Function keys - these include dedicated function keys and programmable control devices such as keys with programmable legends, multifunction keysets and touch screen devices.

No assumptions are to be made with regard to the specific control methodology (such as the layout and activation of menus), data presentation or LPD symbology.

3.3 Secondary objective

The secondary objective of this task is to undertake research into software engineering techniques suitable for the design and development of combat systems, their prototypes and prototyping tools.

4. DISCUSSION OF OBJECTIVES

4.1 Restricted versus unrestricted prototyping

Many existing prototyping facilities are relatively restrictive with regard to display and control methodologies (ref.4). They may offer only one form of menu control, for example, and provide a standard LPD layout. Limitations of this form are an advantage in terms of the development time for the facility and the time to generate a prototype. Such an approach is appropriate where the developer of the prototype is also the developer of the system and he is committed to a standard man-machine interface. In this case, the prototyping tools match the developer's system design tools, and the approach is ideal for evolutionary system development, where the prototype is evolved into the final system.

This approach, however, lacks flexibility. Any suggestions for changes which transcend the limited format provided will usually be resisted by the developer. This problem arises because, although the prototype is designed for rapid changes, the prototyping environment and tools are not. The result of this limitation is that the final system design requirements may be restricted by limitations in the prototyping facility.

It is also difficult, or impossible, to use such a facility to prototype systems using a different interface methodology, as will be necessary if the user requires compatibility with an existing system, or when prototyping enhancements to an existing system.

It is sometimes argued in defence of restricted prototyping environments that the operator interface is prototyped *conceptually* and the eventual interface will be quite different. While a conceptual interface can be very useful for the system designers, it can also be confusing for users, who typically do not have the experience to mentally extrapolate a conceptual design into the final implementation. Their role in the prototyping activity, and hence the value of their participation, can therefore be severely reduced if the operator interfaces are not genuinely representative.

The objective then is to attempt to prototype as faithfully as possible the "look and feel" of the target system, with the ability to change interface methodologies as required. It is accepted, however, that no matter how sophisticated the tools available, it will always be possible to invent a methodology that will be impossible to prototype without a major software development.

A brief review of existing combat systems, however, would show that most systems are remarkably similar in concept and use a relatively limited number of interface methodologies. For example, consider the difference between a touch screen menu system and a pull-down, mouse activated menu system. While the methods of display and activation may differ considerably, the menu structure, contents and actions could be defined using a common definition format, suitable for other menu systems as well.

The deliberate lack of restriction in user interfacing methodologies is therefore an encouragement to identify prototyping techniques which take advantage of the conceptual similarities of existing and proposed methodologies, and which can be adapted to new methodologies.

4.2 Representative displays, controls and performance

As discussed above, the value of the user's participation in the prototype's iteration will depend to some extent on how faithfully the target system's displays, controls and performance are represented. An unrepresentative prototype may be counterproductive with changes being directly influenced by limitations in the prototyping environment. Alternatively, deficiencies in the system design may be obscured in an inadequate prototype.

The performance of the prototype, particularly with regard to the response time for a display change as a result of an operator's action, will often influence the way a system is used. In a recent project to improve the response time for certain functions in an existing system, it was noticed that operators quickly changed their normal procedures when using the new system, and used the improved functions much more often (ref.5).

The need for a representative interface will change according to the application. The use of a mouse (in the prototype) instead of a rollball (in the target system) may be quite acceptable when the device is used for simple pointing operations requiring small cursor movements. Where the device is used for pull-down menus, however, a mouse is likely to provide better performance than a rollball and hence obscure potential deficiencies.

In meeting the objective of a representative operator interface, therefore, it will be necessary to consider the provision of flexibility in the development of the prototype including display size, colour and resolution, entry devices, and workstation layout.

4.3 The aim of prototyping - a requirements specification

The main aim of prototyping in this context is to provide a clear specification of requirements which is easily understood by the user and towards which he has had a knowledgeable input.

The prototype itself could be regarded as a specification (ref.6), but not only is it unrepresentative and incomplete in some aspects, but it is not defined on paper and hence is difficult to include in a contractual agreement.

A computer program is also inadequate as a specification. It may define performance precisely to a systems analyst, but is generally incomprehensible to users. The necessity to provide an analyst/programmer to translate a user's inputs to the prototype (as recommended in ref.7) will dilute the effectiveness of the user and his capability to iterate the prototype when isolated from the prototype builder.

Most system formal design and analysis representations are equally unsatisfactory in this regard - they require too much learning and experience to be assimilated by the average user, although they are used in several prototyping environments (ref.2,8,9). For example, the author's experience with Data Flow Diagrams for the development of operational and functional requirements is that the typical military user finds them confusing and hence cannot contribute adequately to the definition process.

It is important that the requirements for the operator interface of a system are described in a form which operators can understand (and which is also unambiguous to the developers). Because the definition of the requirements is incorporated in the prototype, it should be possible to generate the requirements automatically from the prototype. This is not to say that the objective of the Rapid Prototyping Environment (RPE) is to produce executable specifications (ref.8,10), but that the prototype can produce clear requirements which contribute to the requirements specification.

It is therefore an objective of the prototyping environment that operator interface definitions are represented in plain text and in a form suitable for assimilation by users and incorporation in a specification. This approach will allow the users to assist in the generation and iteration of the prototype, as well as reducing the effort for post-generation of acceptable specifications.

4.4 Portability

When a prototype is developed, it is important that it is accessible by those who need to use it. If the combat system developers, prototype developers, system consultants and users are geographically distant from each other, which is often the case, locating the prototype at a single site is likely to restrict its potential advantages. Similarly, there are advantages in being able to demonstrate problems or solutions at remote sites.

It is desirable, in fact, that the prototype may be demonstrated on computing equipment already available at remote sites, or on equipment that may be readily hired. In this case, only the software for the prototype would need to be transferred.

4.5 The prototyping environment

To meet the prime objective it will be necessary to develop a Rapid Prototyping Environment (RPE). It is assumed that the RPE will consist of the following:

- A variety of equipment, including computers, displays and control devices suitable for interconnection to form the hardware of the prototype.

- Software tools used for generating and running the prototype. Some of these will be executable programs, used for generating data for the prototype, such as a menu generator. Others will be program components, suitable for incorporation in the prototype itself, such as display drivers.

- Staff experienced in generating and iterating the prototype.

Generation of the prototype will be based on tentative (possibly broadly defined) requirements, and consist of the interconnection of appropriate hardware, and the selection and integration of relevant software into a suite of software for the prototype. It is accepted that this stage will require the adaptation of existing software and the development of new

software (particularly for new applications and interface styles), but where possible this will be achieved through software generation tools such as the Database Generator.

Iteration of the prototype will mainly consist of changes to the menu control structures and data display formats, and will not normally require recompilation of the prototype software. Configuration data of this type will be managed through definition files (or "scripts"), which will be read directly by the program during prototype execution.

The objective of using of definition files where possible, rather than recompilation, contrasts with other rapid prototyping research (ref.9,11), and is influenced by the following considerations:

- Translation, recompilation and linking of the prototype is likely to be time consuming and will reduce the responsiveness of the prototype to change.

- If definition files are translated directly by the prototype software, the necessity for compilation facilities at remote sites will be significantly reduced, enhancing the portability of the prototype.

5. THE RAPID PROTOTYPING ENVIRONMENT (RPE)

5.1 Processor

The selection of the main processor type is influenced mainly by the need for portability. Four processor families were considered: VAX, Macintosh, Sun and IBM compatible personal computers (PCs). The selection of PCs as the processors for the RPE is based on the following:

- General availability throughout the world.

- Low cost (and low hiring cost if necessary at remote sites).

- Open architecture allowing simple interfacing.

The VAX processor was initially selected as the main processor in Phase 1 of this task. Although VAX processors are installed in most of the sites considered, availability for prototyping is often limited and there is no guarantee that the response times required can be met when operating in a multi-user environment. Most sites do not support colour displays and there are likely to be problems associated with the installation of special purpose interfaces in these processors.

5.2 Software

5.2.1 Language

The Turbo Pascal language (version 5.5 or later) will be the main programming language used in the RPE. The initial intention was to develop software in Ada, and in Phase 1 software was developed in VAX Ada, running on a VAX 8200. Although this development was successful, the change to Turbo Pascal was proposed as a result of the change of the main processor type to PCs, and supported on the following grounds:

- Although Ada compilers are available for PCs, they are slow to compile, very large, and in some cases require special boards to be installed in the PC. This would make software changes at remote sites difficult and complex, and reduce portability.

- Turbo Pascal is now a proven language system which is designed for high productivity. Its unit structure (similar to Ada packages) is ideal for modular and incremental development, and it is supported by high quality development tools.

- Because PCs are now firmly established as the most common general computers (over 20 million systems in the world) it is highly likely that they will be supported for the foreseeable future. This reduces the importance of portability of software between processor types, which is a strength of Ada.

- Turbo Pascal provides many features which are tuned to the PC environment, allowing high performance for display control and handling of interrupts.

5.2.2 Existing software tools

Although prototyping tools of varying levels of sophistication are available for PCs (ref.4), few of these are suitable for installation in a complex integrated environment and source code is rarely provided to ensure adaptability to varying applications.

5.2.3 Software concepts

This section describes some of the software engineering concepts that will be used in the task.

(a) Recoding and reusable software

While the RPE software will be designed to cater for most existing and proposed combat system user interfaces, some recoding will almost always be necessary to meet differing interface methodologies and to implement application specific functions. Software therefore must be designed to be reusable: it must be highly modular, table driven where possible, and modules must be designed for generic rather than specific use.

It must also follow the "shell" concept where appropriate, providing clearly separated layers where there are likely to be major changes (such as in the use of different display devices).

(b) Translators

Several translators will need to be generated for the task. For major changes in the interface methodologies or display and control devices, code generators will be required (vertical translation - ref.10). For changes in display formats and control structures, the translation will normally occur during initialisation of the main program, and consist mainly of syntax checking, conversion to a table structure and cross-referencing to improve response times (lateral translation). Initial investigations have shown that this translation is likely to be reasonably fast for a

medium sized application (of the order of one minute), but pre-translation will be considered if necessary.

5.3 Displays

In Phase 1 the main display processor was a Tektronix 4128 system which was found to be inadequate in flexibility and general performance. This will be replaced by a PC based display system including a Matrox PG-1281 graphics controller. This controller consists of a single card installed in a PC and requires a high resolution display.

Because of the requirement to use existing PC equipment where possible, the system will also be designed to use standard PC display systems where possible including VGA, EGA and monochrome displays. For some applications this may not be possible or will result in reduced performance in terms of display appearance, resolution and response times.

5.4 Controls

5.4.1 Position entry devices (PEDs)

The types of PEDs envisaged are rollballs, mice, forcesticks and joysticks. These are widely available for PCs, usually using a mouse or joystick interface.

5.4.2 Function keys, switches and other control devices - PIPS

In Phase 1 a PC based touch screen system was used to simulate dedicated function keys. Because of the wide variety of control devices used and proposed for combat systems, a general purpose control device incorporating a flat panel display and touch screen is now being developed for the RPE.

The Programmable Interface Panel System (PIPS) will enable the simulation of a variety of control devices including touch screen interfaces, keyboards, knob selectors, slider controls, thumbwheels and multifunction keysets (which have fixed keys and programmable legends). The PIPS will also provide for auxiliary display devices such as alphanumeric displays (Totes), meters and digital readouts (DROs).

6. ACTIVITIES

6.1 Outline

The activities are divided into three phases.

Phase 1 is the preliminary phase which is now complete. In this phase, a small combat system for a minehunter was developed. Although designed and developed using traditional methods, it incorporated some prototyping techniques, particularly with regard to the menu structure. The aim of "hard coding" this example was to familiarise staff with the environment (particularly Ada) and the problems of system development. It was also intended to provide a stable baseline from which further developments could advance. Other activities in Phase 1 included the development of a package to facilitate the use of geographical coordinates, the development of specifications for the Programmable Interface Panels and the refinement of requirements for the following phases.

In Phase 2 the identification, design and development of prototyping tools is the main activity. These will include the following:

- LPD interface standard
- Database generation and control
- LPD generation and control
- Menu generation and control
- Navigation generation and control
- Simulation control.

Phase 3 is the integration phase where the other tools are combined and coordinated by the Reactor, the executive of the prototype. In addition, the existing tools will be refined including the development of interactive tools for prototype generation and iteration.

6.2 Detailed activities

6.2.1 Package GEO

Most combat systems use geographical coordinates (latitude and longitude) to refer to the positions of objects and geographical features. Even when the main working coordinates are cartesian (often called "grid" coordinates, relative to a predefined reference point), geographical coordinates must be used for some calculations and for correlation and communication with the outside world.

Package GEO (ref.12) provides conversion between geographical and cartesian coordinates, and other functions using geographical coordinates. Three different levels of accuracy (and hence speed of computation) are defined for different applications. Commonly used functions such as the distance of a point from a line, and the bearing and range of one point from another are included.

GEO is implemented both in Ada and Turbo Pascal.

6.2.2 Programmable Interface Panel System (PIPS)

The purpose of developing the PIPS is to provide simulation of operator entry and control display devices including the following:

- Dedicated function keys
- Programmable legend function keys
- Multifunction keysets (incorporating both fixed keys and an alphanumeric display)
- Individual readout displays (such as LEDs)
- Status lamps
- Touch panel devices.

Each PIP will be a flat panel display device including a touch sensitive surface, and be designed to allow the panels to be mounted adjacent to each other to simulate larger panels. A limited graphics capability is proposed with a pixel resolution of approximately 500 x 500. The panels will be controlled by a PIP controller board in a PC, which will control up to four panels.

As part of this activity a PIPS interface protocol will be developed, providing for the high level control of simulation for devices such as those listed above.

6.2.3 LPD interface standard

An LPD interface standard will be developed to facilitate the use of different types of graphic display equipments in prototypes. The aim of catering for different displays is threefold:

- To take advantage of advances in technology (in a rapidly changing field).
- To allow the use of the RPE at remote sites using less capable standard displays.
- To provide a test bench for complementary research into combat systems display technology.

Unlike existing standards such as GKS and PHIGS, the LPD standard will directly address the features commonly found in tactical displays for combat systems including the following:

- Graphical and legend data
- Graphic objects such as tracks, bearing lines and labelled markers
- Range scale and LPD centre changes
- Hooking of graphic objects
- Position entry devices
- Declutter facilities
- Display update and response priorities.

While in some cases an implementation of the standard may use GKS or PHIGS to provide primitive control, in others it may be more appropriate to implement the standard explicitly for a particular display system, for reasons of efficiency or where GKS is not available.

6.2.4 Generation and control tools

These activities will involve the definition and development of tools for the generation and control of various elements of a prototype.

Database	The generation of database requirements including data types, legal values, alphanumeric and graphic display representation, and the handling of database changes.
LPD	LPD formats and content, and the dispatch of actions generated from the LPD, including inputs from the PED and its associated controls (eg mouse buttons).
Menu	Menus, function keys, keyboards and other control inputs and displays not directly associated with the LPD.

Navigation	The simulation and control of single navigation sensors and integrated navigation systems. Some examples of the sensors to be considered are GPS Navstar, gyro compass, Doppler systems and inertial systems.

6.2.5 Simulation control

This activity involves the definition and development of tools for controlling the simulation for the prototype. Functions requiring control will include:

- Injection of failures (within the prototype).
- Controlling navigation accuracy and parameters.
- Simulating movement of the host platform.
- Generation and control of other platform movement and actions.
- Controlling information from other subsystems, eg radar.
- Controlling data link information from other platforms.

While the interface methodology for the Simulation Controller may be fixed for all prototypes (it will not actually be part of the simulation itself) the design must be sufficiently flexible to provide control for a wide variety of applications.

6.2.6 Reactor generation

The Reactor is proposed as the software coordinating the other elements of the prototype, and it will process actions arising from operator inputs and other events. Essentially it will define the high level functionality of the system being prototyped. During Phase 2, the Reactor will be application specific, and could be regarded as *the* application program using the configured tools described above.

The aim of this activity in Phase 3 is to minimise the necessity for "hard coding" within the Reactor. This will be done by identifying reactions which are common in concept to most systems, and by examining methods of automatic code generation for more specific reactions.

6.2.7 Interactive iteration tools

The objective of this activity is to provide tools for modification of the prototype without seriously interrupting its execution, both increasing the efficiency of the prototype iteration (and generation) process, and enhancing the ability of users in particular to effect changes. Simple examples include the changing of the menu structure and the repositioning of an object's label relative to its symbol on the LPD.

7. CONCLUSIONS

This task will provide a powerful, flexible and portable prototyping environment suitable for the design and analysis of combat system operator interfaces.

Although the task is aimed at the research and development of rapid prototyping environments, much of the work will also be directly relevant to software engineering techniques for the design and development of combat systems themselves. Not only will a significant contribution be made to research in this latter field, but the task is likely to identify several areas for future research.

REFERENCES

1 Alavi, M.
"An Assessment of the Prototyping Approach to Information Systems Development".
Communications of the ACM, Vol. 27, No. 6, June 1984.

2 Gomaa, H. and Scott, B.H.
"Prototyping as a Tool in the Specification of Requirements".
Proceedings, 5th International Conference on Software Engineering, IEEE Computer Society, 1981.

3 Taylor, T. and Standish, T.A.
"Initial Thoughts on Rapid Prototyping Techniques".
ACM SIGSOFT Software Engineering Notes, Vol. 7, No. 5, December 1982.

4 Schwalm, R.C., Thomas, M.E., White, R.J. and Williams, R.D.
"User Interface Prototyping: A Review of PC-Based Tools and their Features".
Proceedings, Human Factors Society, 31st Annual Meeting, October 1987.

5 "An Investigation into the Modification of Program Loading Techniques Used in the AQS-901 Sonics Processor aboard Orion Aircraft".
Gabb, A.P. and Ward, D.J.
WSRL-0561-TR, March 1988.

6 Scharer, L.
"The Prototyping Alternative".
ITT Programming, Vol. 1, No. 1, 1983.

7 Miller, D.P.
"Instant Prototyping Using HyperCard on the Macintosh".
SAND--88-1862C, Sandia National Laboratories, December 1988.

8 Daley, P.C.
"C3I Rapid Prototype Investigation".
RADC-TR-85-216, Martin Marietta Denver Aerospace, January 1986.

9 Raum, H.G.
"Design and Implementation of an Expert User Interface for the Computer Aided Prototyping System".
Naval Postgraduate School, Monterey, California, December 1988.

10 Baxter, R.D.
"A Brief Discussion of Formally Based Approaches to Rapid Prototyping".
BAe-WAA-RP-RES-SWE-440, British Aerospace, Aircraft Group Warton Division, March 1987.

11 Software Engineering Institute, Carnegie Mellon University
"Serpent Overview".
CMU/SEI-89-UG-2, ESD-TR-89-06, April 1989.

12 Carter, C.B.
"A Study of Coordinate System Conversions and Other Algorithms for Use in Combat Systems".
WSRL-TN-2/90, 1990.

DOCUMENT CONTROL DATA SHEET

Security classification of this page : UNCLASSIFIED

1 DOCUMENT NUMBERS

AR Number : AR-005-981

Series Number : WSRL-TM-3/90

Other Numbers :

2 SECURITY CLASSIFICATION

a. Complete Document : Unclassified

b. Title in Isolation : Unclassified

c. Summary in Isolation : Unclassified

3 DOWNGRADING / DELIMITING INSTRUCTIONS

4 TITLE

RAPID PROTOTYPING FOR COMBAT SYSTEMS - A RESEARCH TASK

5 PERSONAL AUTHOR (S)

A.P. Gabb

6 DOCUMENT DATE

July 1990

7

7.1 TOTAL NUMBER OF PAGES 15

7.2 NUMBER OF REFERENCES 12

8

8.1 CORPORATE AUTHOR (S)

Weapons Systems Research Laboratory

8.2 DOCUMENT SERIES and NUMBER
Technical Memorandum 3/90

9 REFERENCE NUMBERS

a. Task : DST 89/218

b. Sponsoring Agency : DSTO

10 COST CODE

11 IMPRINT (Publishing organisation)

Defence Science and Technology Organisation

12 COMPUTER PROGRAM (S) (Title (s) and language (s))

13 RELEASE LIMITATIONS (of the document)

Approved for Public Release

Security classification of this page : UNCLASSIFIED

Security classification of this page : UNCLASSIFIED

14 ANNOUNCEMENT LIMITATIONS (of the information on these pages)

No limitation

15 DESCRIPTORS

a. EJC Thesaurus Terms: Computer interfaces

b. Non - Thesaurus Terms: Prototyping
Rapid prototyping environment

16 COSATI CODES

1205

17 SUMMARY OR ABSTRACT
(if this is security classified, the announcement of this report will be similarly classified)

(U) This paper describes the requirements and activities for a research task relating to the development of facilities for the rapid prototyping of operator interfaces in combat systems. Emphasis is placed on requirements analysis, simplicity of modification, and portability of the prototype.

Security classification of this page : UNCLASSIFIED