

DELIVERABLE IDENTIFICATION

Identification Number	MLAP-364-D-WP3.1.2.2
Type	Report
Title	The Future of SAM Tools
Status	Final
Deliverable	3.1.2.2
Workpackage	3
Task	3.1
Period Covered	1/95 - 10/95
Date	20-10-95
Version	1.0
Number of Pages	x + 49
Author	D. N. L. Howell of Coleman & Howell for: Speech Research Unit, Defence Research Agency, St Andrews Road, Malvern, Worcs. WP14 3PS, UK.
WP/TP Responsible	Speech Research Unit Defence Research Agency, St Andrews Road, Malvern, Worcs. WP14 3PS, UK. Tel (44) 1684 896506 Fax (44) 1684 895103 (44) 1684 894384 Email mga@signal.dra.hmg.gb
Project Contact Point	Dr Harald Höege, Siemens AG
CEC Project Officer	Jose Solér
Status	Unlimited
Actual Distribution	Consortium
Supplementary Notes	DRA reference number: DRA/CIS/374/016/1.0
Keywords	Speechdat speech assessment methodology tools
Abstract	A review of the current state of the SAM multi-lingual speech input/output assessment tools and ways of supporting them in the future. The Speechdat corpora and the SAM tools are fundamental and essential resources for those working to keep European speech and language technology advancing ahead of its competitors. The future of the SAM tools is vital to the successful use of the Speechdat corpora and to the promotion of European standards of assessment.
Status of Abstract	Consortium

**Uncontrolled Copy
Unlimited**

The future of the SAM tools

DRA/CIS/374/016/1.0

Unlimited

Cover + x + 49 pages

D.N.L.HOWELL

October 1995

**Unlimited
Uncontrolled Copy**

**Uncontrolled Copy
Unlimited**

This report has been prepared for MOD and, except as indicated, may be used and circulated under the arrangements of Order

It may not however, be used or copied for any non-Government or commercial purpose without the written agreement of DRA.

Approval for copying should be sought from:

DRA Intellectual Property Department
Tel: +44 (0) 252 39 2616

© **Crown Copyright (1995)
Defence Research Agency
Farnborough, Hampshire, GU14 6TD, UK**

**Unlimited
Uncontrolled Copy**

Authorization

**Prepared
by:** D. N. L. Howell &
A. P. Buckland

Signature:

**PQR
Approval:** M. J. Tomlinson

Signature:

**Authorized
by:** Prof. R. K. Moore

Title: Head SRU

Signature:

Issued by Prof. R. K. Moore

Building EX Room 1

DRA Malvern

Telephone: +44 (0) 1684 894074

THIS PAGE LEFT INTENTIONALLY BLANK

Record of Changes

This is a controlled document.

Additional copies should be obtained through the issuing authority.

In the extreme event of copying locally, each document shall be marked 'Uncontrolled Copy'.

Amendment shall be by whole document replacement.

Proposals for change should be forwarded to the issuing authority.

Issue	Date	Details of changes
0.a	22/06/95	Initial draft
0.b	26/06/95	Comments from initial review incorporated
0.c	13/07/95	Comments from review incorporated
0.d	17/07/95	Formatted
0.e	05/08/95	Appendix 2 received from CSELT and included
0.f	19/10/95	Reformat after R&A
1.0	19/10/95	Issued

THIS PAGE LEFT INTENTIONALLY BLANK

Abstract

A review of the current state of the SAM multi-lingual speech input/output assessment tools and ways of supporting them in the future. The Speechdat corpora and the SAM tools are fundamental and essential resources for those working to keep European speech and language technology advancing ahead of its competitors. The future of the SAM tools is vital to the successful use of the Speechdat corpora and to the promotion of European standards of assessment.

THIS PAGE LEFT INTENTIONALLY BLANK

Executive Summary

This report reviews the current state of the SAM multi-lingual speech input/output assessment tools and ways of supporting them in the future. The Speechdat corpora and the SAM tools are fundamental and essential resources for those working to keep European speech and language technology advancing ahead of its competitors. The future of the SAM tools is vital to the successful use of the Speechdat corpora and to the promotion of European standards of assessment.

The SAM tools were developed under the ESPRIT project 2589 (multi-lingual speech input/output assessment, methodology and standardisation) known as SAM and ESPRIT project 6819 (speech technology assessment in multi-lingual applications) known as SAM-A. The multi-lingual speech corpora gathered under the Speechdat project will use the SAM standards developed under SAM and SAM-A. In this report the SAM Input Assessment Tools (IAT) have been investigated in depth and models for distribution and support mechanisms have been proposed. These models for distribution and support cover the range of proposed communities of users. The logic of the arguments applied to the input assessment tools can equally well be applied similarly to the output assessment, data gathering and validation tools developed under SAM, SAM-A and Speechdat.

The European Language Resources Association (ELRA) is seen as a key player in the future of the tools, acting to impartially set, monitor and promote standards and to act as a focus for development, support and promotion. Customers for speech input technology want and need to make informed buying decisions. If the European speech technology community is to satisfy this need then it must support standards and tools set up to achieve the goal of standardised assessment.

THIS PAGE LEFT INTENTIONALLY BLANK

List of contents

AUTHORIZATION	i
RECORD OF CHANGES.....	iii
ABSTRACT	v
EXECUTIVE SUMMARY	vii
LIST OF CONTENTS	ix
1. OVERVIEW OF THE SAM SPEECH INPUT ASSESSMENT SYSTEM.....	1
2. THE STATE OF THE SPEECH INPUT ASSESSMENT SYSTEM AS REPORTED BY SAM-A.....	3
3. THE PRESENT STATE OF THE INPUT ASSESSMENT TOOLS.....	7
4. NEW TECHNICAL DEVELOPMENTS	9
5. MANAGEMENT	11
6. A NEW HARDWARE SPECIFICATION	13
7. SURVEY OF PROSPECTIVE USERS OF THE SAM INPUT ASSESSMENT TOOLS	15
8. THE MARKET FOR SPEECH INPUT ASSESSMENT TOOLS.....	17
9. MODELS FOR THE DISTRIBUTION AND REGULATION	19
10. THE RELATIONSHIP WITH OTHER STANDARDS	23
11. CONCLUSIONS	25
12. REFERENCES.....	27
APPENDIX 1- MODIFICATIONS BY DRA FARNBOROUGH	29
APPENDIX 2- MODIFICATIONS BY CSELT TURIN.....	35
APPENDIX 3- MODIFICATIONS BY ALCATEL - MILAN	45
INITIAL DISTRIBUTION	47
REPORT DOCUMENTATION PAGE.....	49

THIS PAGE LEFT INTENTIONALLY BLANK

1. Overview of the SAM speech input assessment system

The speech technology community in the European Union has recognised the need for widely available tools for standard assessment. Those involved in the SAM projects have developed the SAM tools for this purpose. However, the demand for assessment tools stretches further than just those involved in SAM. Whilst some suppliers currently use the SAM tools for assessment of their in-house products, prospective users are currently left with no reliable way of informing their buying decisions. They are distrustful of suppliers claims and have no way to make an informed choice as to which product is appropriate to their needs.

The speech input assessment system consists of the SAMPAC and SAM_SCOR speech Input Assessment Tools (IAT) programs running on a personal computer workstation (SESAM) equipped with a CD-ROM drive and signal I/O card. Software is written in C and is currently compiled using the Microsoft C compiler for MS-DOS. The system performs three main functions:

1. Management of speech signal, annotation, configuration and results data.
2. Control of the automatic speech recognition (ASR) system whilst being assessed.
3. Performance scoring of the ASR system under evaluation.

SAMPAC performs the management and control functions, whilst SAM_SCOR performs the scoring functions. Results may be exported to external scoring routines such as the National Institute for Standards in Technology (NIST) software. Annotated test and training data must be supplied where appropriate and ASR system responses are then collected and scored. An ASR system specific driver must be supplied to translate the system responses to the SAMPAC protocol and vice versa.

THIS PAGE LEFT INTENTIONALLY BLANK

2. The State of the speech input assessment system as reported by SAM-A

2.1. Software - SAMPAC and SAM_SCOR

A survey of users and a review the of SAM tools was conducted by the last SAM project (SAM-A). The results of this survey were presented in report I2 [1]. There was also a review of the technical status of the tools. This was presented in report I6 [3].

In particular, report I6 [3] and appendix B of report I3 [2] give a comprehensive list of known bugs and problems as found by users, these include bugs and problems with SAMPAC and SAM_SCOR. Report I2 [1] collects information on the state of all tools via questionnaire based survey of then current users. Report I6 [3] reviews the state of all tools. Report I8 [5] deals with future architecture in broad terms - only speech output assessment software is covered in detail. SAMPAC is not covered. The idea of a 'test designer' was presented as a module in the top level architecture but is not described in detail.

2.2. Reported deficiencies in SAMPAC

1. Does not deal with DC-offset in speech files.
2. Directory path in control file requires a backslash as last character.
3. Continuous ASR systems that have a long response time may send their last response after SAMPAC has finished collecting them. SAMPAC does not wait at end of a section to accept these slow continuous ASR system responses.
4. Speech signals played to the ASR system by SAMPAC often contain an audible click near the beginning. This tends to cause spurious insertion errors.
5. When playing a signal file, the SAMPAC PAUSE instruction is ineffective except if it is executed at the beginning of the file.
6. Needs a method of interrupting a test of a continuous ASR system.
7. The display incorrectly shows phrases and incorrectly manages long strings of characters.
8. Incorrectly aligns responses with the speech signal.
9. Incorrectly transfers the test label string to the ASR system driver.
10. Does not have a method of switching the active syntax node in an ASR system under test. (This is also known as context switching.)
11. Has no agreed mechanism for testing adaptive ASR systems.

12. Does not copy the contents of the 'errors window' to a log file.
13. Cannot be run in a batch file more than once.
14. The heavy use of DOS conventional memory space to run SAMPAC, SAM_SCOR and the ASR system drivers means that little or no memory is left. This leads to system crashes and difficulties when writing ASR system drivers to fit in the available memory.

2.3. Solutions to deficiencies in SAMPAC

Some possible solutions to the identified deficiencies in SAMPAC are documented in appendix A of report I7 [4]. Comprehensive solutions will rely on the harmonisation of the current versions.

2.4. Reported Deficiencies in SAM_SCOR

1. Does not score word and sentence performance.
2. Does not measure false positive frequencies. This is particularly important when evaluating word spotting systems.
3. Does not score response time.
4. Does not output variation of scores with time.
5. Incorrectly parses the comment rows in the scoring control files.
6. Does not accept control file with extension different to .SCL.
7. Only handles up to about 350 tokens.
8. McNemar test causes maths error.

2.5. Solutions to deficiencies in SAM_SCOR

Some possible solutions to the identified deficiencies in SAM_SCOR are documented in annexIV of report I2 [1]. Comprehensive solutions will rely on the harmonisation of the current versions.

2.6. General comments

- ASR system drivers require substantial software effort to produce. Much better documentation and guidance is required.
- The development of control files to configure the system to produce a particular test is time consuming and requires the editing of many files.

- Software should make more use of extended/expanded memory.

2.7. Hardware - the SESAM workstation

The following deficiencies have been identified:

1. Incompatibility between the OROS AU21 driver (iau21drv) and OROS AU22 signal acquisition board.
2. The specified Phillips CD-ROM is now seen as obsolete but more modern drives produce intermittent problems when used with the SAMPAC software.

2.7.1. GENERAL COMMENTS

Report I6 [3] reviews hardware issues. Some thought that disk caching would solve problems with the CD-ROM. The desire to make SAMPAC independent of any particular signal acquisition board was expressed and a minimum specification was given. Annex II of Report I6 [3] discusses the use of telephone interface hardware both for testing ASR systems over the telephone network.

THIS PAGE LEFT INTENTIONALLY BLANK

3. The present state of the input assessment tools

SAM-A (ended 1993). However, due to the premature cut in funding, much of the work is incomplete. Before and since the end of SAM-A, users have modified the code in an uncontrolled way, often loosing backwards compatibility. This has produced the current situation where there are at least as many versions of the speech input assessment software as there are users. How compatible these versions are is currently unknown.

3.1. Survey of users of the SAM input assessment tools

The author has conducted a survey to find current users of the SAM-IAT. The results of the survey are summarised in Table 1. Current users are largely involved with assessing the performance of a particular ASR system but also evaluate or will soon be evaluating other ASR systems. Users were concerned by a number of bugs in the software, lack of memory for new ASR drivers and compatibility problems with CD-ROM drives other than the ageing drive in the original specification. In general, bugs and deficiencies in the code have been fixed locally, producing divergent versions. Problems that have been encountered and their solution are documented in appendices 1 to 3.

The LOADRISE and RISE, designed to manage speech data utilising Oracle database software are not being used as users find them unnecessary.

User	SAMPAC version	SAM_SCOR version	Usage
DRA - Farnborough -UK	3.1+unofficial modifications	3.1 + dpscore	version comparison ¹ -McNemar test, confusion matrices
DRA - Malvern - UK	3.2	dpscore	version comparison - accuracy, quality assurance ² .
Alcatel - Milan - Italy + more .. awaiting more info	3.2+unofficial modifications	4.0	version comparison and evaluation of competing products
CSELT - Turin - Italy	3.2+unofficial modifications	?	version comparison and evaluation

Table 1 current SAMPAC users

¹ version comparison refers to the assessment of successive versions of a particular (usually in-house) device.

² examples of a device are tested to ensure that they perform to a minimum standard.

3.2. Summary of active users

Alan South-DRA Farnborough has an extensively modified version of version 3.1 of SAMPAC - and uses accuracy score, McNemar test and confusion matrix to evaluate changes to Marconi ASR1000 system parameters - doesn't compare recognisers. May use to assess different recognisers in near future.

Brian Mellor-DRA Malvern has version 3.2 of SAMPAC and has replaced SAM_SCOR with the DRA 'dpscore' routine. The original version of SAM_SCOR used was 3.1. The SAM-IAT is used to compare versions of the DRA Aurix ASR system.

The use of v3.1 instead of v3.2 illustrates the problems that some users have had due to the uncontrolled modification of the code. v3.2 is not backwardly compatible with v3.1 and removed a function essential to the DRAs work. It was therefore necessary to stick with v3.1 even though v3.2 may have had other desirable features and bug fixes.

DRA Malvern has also produced some documentation covering use of the SAM-IAT.

Francesco Senia - CSELT- Turin. Uses an extensively modified version of SAMPAC based on version 3.2. to compare versions of the CSELT telephone based ASR system and to evaluate ASR systems for third parties. The version of SAM_SCOR used is unknown (awaiting info).

Antonello Riccio - Alcatel - Milan. Uses an unknown version of SAMPAC (awaiting info) and version 4.0 of SAM_SCOR to assess performance of in-house ASR system. Also evaluates competing products. Antonello Riccio has distributed the SAM-IAT to

- Alcatel SEL in Stuttgart
- Alcatel Mobile Phones in Colombes
- Alcatel Business System in Illkirsch

ASR Applications that SAM has been used with are: Telephone Answering Recording Machines with speech recognition capabilities and Voice Dialler for in-car telephones.

3.3. Modifications to SAMPAC by users since end of SAM-A

SAMPAC has been modified by the DRA, by CSELT and by Alcatel. The work done by the DRA has been documented and is presented in Appendix 1. The work done at CSELT is presented in Appendix 2 and the work done by Alcatel is presented in Appendix 3.

3.4. Modification to SAM_SCOR by users since SAM-A

Although no modifications have been made to SAM_SCOR since SAM-A version, DRA Malvern has replaced SAM_SCOR with a connected word recognition scoring routine using dynamic programming and an accuracy measure that includes insertion and deletion errors and response time scoring.

4. New technical developments

SAM-A Report I8 [5] describes a possible technical future for the SAM tools based on new software for Microsoft Windows written in Microsoft C/C++ using Microsoft foundation C++ classes to provide user interface and data streaming. This approach is described in the context of a unified architecture for the SAM-tools.

However, current users are unanimous in expressing the desire to keep the SAM tools DOS based. Although MS-Windows provides a well supported, easy to use (though not always Common User Access (CUA) compliant) desktop metaphor for users, it is based on an event driven architecture that fails to provide a way of managing serial processing at the level of the operating system. MS-DOS, for all its failings has an elegant and flexible command line interface and batch command facility, ideal for running a test as a batch of commands in an auditable and easily reproducible manner.

The current source code is reasonably portable. However, portability may be compromised if the code is based on Microsoft Windows. Many of the desirable features of using Microsoft Windows rely on the use of well supported foundation classes or Software Developers Kit (SDK). This may render the source code very Windows specific and non-portable unless carefully designed.

For these reasons, moving to Windows presents problems.

It may be possible to address the points raised by those preferring Windows and those preferring DOS by referring to the needs of the wider group of prospective and current SAM tools users. SAMPAC users have repeatedly commented that the production and debugging of control files is the most time consuming and difficult part of running a test and they would like an easier method. However, the DOS batch file system provides an essential function. What is demanded is the well supported, easy to use desktop and software integration of Windows together with the easily audited reproducibility of the DOS batch file system.

These seemingly contradictory preferences might be easily resolved by creating a Windows utility that will help to design tests in a way that is more intuitive to the less expert user. This “test designer” would be independent of the test management software but would store the “test” to be run as a set of SAM control files and/or commented batch files that could be used to run a test. Documentation would be aimed at the non-expert user. This type of tool can easily be written using the C/C++ language and standard Windows components, would be Windows specific and non-portable. Tests could be run either by spawning a DOS process from the Windows environment or by running a DOS program directly.

The way forward may therefore be to combine DOS and Windows programs and to use each where appropriate. The proposed “test designer” would insulate the non-programmer, speech technology buyer and less-expert user from the rigours of direct control file editing and would provide the benefit of *ease-of-use* asked for by the potential users of the SAM-IAT. This approach would not compromise audit and the reproducibility that is so absolutely essential for transparent and fair comparisons to be made in the scientific arena

and in the marketplace. An added benefit of retaining the DOS text file approach would be that complete and standard test configurations could be easily distributed to users as a set of DOS text files. This would form an easy and transparent regulation mechanism. The adjustments required by those using the SAM tools for output assessment could be accommodated using the test designer.

Any new technical developments should be managed as part of the version control process initiated by the system of management that is decided upon.

5. Management

The EU funding for SAM-A was cut prematurely, leaving the tools in a less than perfect state. One of the main reasons given for the premature cutting of EU funding was the lack of effective software management. The poor control may have been due to a number of factors that have made management extremely difficult. These factors include;

- the large number of laboratories involved
- the turnover of partners
- the number of languages involved
- the rapidly evolving theoretical basis for the tools
- the rapidly evolving nature of the technology to be assessed
- the technical enthusiasm of researchers for local fixes

Since SAM-A, individual research laboratories have not had the incentive or funding to support the impartial management of the software tools for use by third parties. The real co-operation that has been achieved has been remarkable considering the difficulty of the task. A simple example of the difficulties involved is illustrated by one of the repeated comments by users that other users have (out of necessity) modified software unilaterally to meet their own needs without the consent of the other members. This practice has produced a number of differing versions and has resulted in confusion as to which version is current and who is supporting them.

Whatever the model that is chosen for the distribution and regulation of the SAM-IAT, if an agreed software management system is not introduced then the management of the tools will remain problematic.

5.1. Proposed software management scheme

There are international standards that cover software management systems applicable to software engineering projects of the size of the SAM-IAT. An appropriate scheme that embodies many of these standards is the set of software engineering standards (PSS-05-0) adopted by the European Space Agency (ESA). These have been modified to remove references to the ESA to make a set of procedures and guidelines applicable to a wide range of software engineering projects. These standards encapsulate the experience of many hundreds of software engineers around Europe and are therefore likely to find favour amongst the many European parties involved in the SAM-IAT.

The ESA standards are published as a book - 'Software Engineering Standards' [7] and can be directly applied to the management of the SAM-IAT. For example, procedural standards covering project management, software configuration, verification and validation and quality assurance are all clearly laid out as well as procedures for capturing user requirements and managing changes.

THIS PAGE LEFT INTENTIONALLY BLANK

6. A new hardware specification

The hardware platform used to run speech technology assessment is common to all the SAM tools. These include those for speech output assessment, data collection and possibly validation of the Speechdat corpora. Any change of hardware will require agreement on a hardware specification across user groups for the input, output and possible validation functions. This change should be conducted in the way specified in the appropriate distribution and regulation model set out in '**Models for the Distribution and Regulation**'. It has been suggested that workstations might be supplied complete via a central source that would be responsible for their calibration. This would ensure that hardware and software is compatible and performs to a known minimum standard.

THIS PAGE LEFT INTENTIONALLY BLANK

7. Survey of prospective users of the SAM input assessment tools

The author has made an informal survey of a number of prospective users who are not currently using the SAM tools. A market pull-through for assessment tools was found from those needing to make informed buying decisions about ASR technology and suppliers of ASR technology.

Sixteen expressions of interest were elicited from a range of companies and universities ranging from very large computer and telecomm suppliers and consultancies to research groups with an interest in the man-machine interface.

Those buying speech technology felt that even simple comparisons of ASR technology would be helpful. They were very aware of the limitations of tests but felt that something was better than nothing. However, most felt that only 'real' in-situ tests of the technology would convince their end-user customers. When asked, they said that they would find speech input assessment tools useful only if they were controlled by a recognised and impartial body. The promotion of the SAM standards was seen as very important to the future take up of input assessment tools.

The following factors were mentioned by ASR buyers as being desirable or essential:

- essential:
 - Standard tools that will perform widely recognised tests on all competing suppliers technology using standard data and data supplied by the customer.
 - Impartial distribution, control and certification was perceived as essential by speech technology buyers, due to the distrust of information supplied by organisations possibly competing for the users business.
- desirable:
 - Easy to install and use.
 - Good documentation for software and testing procedures aimed at non speech technologists.

The suppliers of ASR technology that were surveyed expressed a desire for speech input assessment tools. Although some had heard of the SAM tools they were generally unaware of their capabilities and of how to obtain them. When asked, these prospective users expressed the wish that the tools were impartially distributed, controlled and certified.

7.1. Summary

Information from prospective users indicates that they would find SAM-IAT very useful in informing buying decisions and/or assessing in-house technology. However, prospective buyers of ASR systems unanimously agreed that to be credible, SAM-IAT must be controlled and successfully managed by an impartial body and would not find it useful if it were not.

THIS PAGE LEFT INTENTIONALLY BLANK

8. The Market for speech input assessment tools

The SAM Input Assessment Toolset (IAT) and the assessment standards themselves only make sense if they are widely used. The use of a commercial channel to promote and distribute SAM-IAT could provide an excellent mechanism to rapidly achieve widespread use.

8.1. Who needs SAM-IAT and what would they use it for?

SAM-IAT can be used in two roles:

1. For characterising a particular ASR system by measuring its performance under standard test conditions.
2. For comparing the performance of ASR systems under conditions relevant to a particular application.

The first role is relevant to those working on improving ASR performance and those maintaining the quality of ASR systems supplied to customers. Identifiable market segments that use IAT in this role are research groups and those responsible for quality control in organisations producing ASR systems.

The second role is required by those comparing different ASR systems under conditions relevant to a particular application of interest. Those wishing to perform this task form two market segments - purchasers of ASR systems and those providing support for these buying decisions. The cost of performing meaningful assessments will preclude all but those focused on providing specialised information and those having to make large and/or important buying decisions. Identifiable market segments are large consultancies, government agencies and system integrators in the telecom, healthcare and military markets.

Input assessment tools already in place in target organisations will represent a major hurdle for SAM-IAT. To overcome this hurdle, customers must feel that SAM-IAT represents a strong standard backed up by an impartial and informed organisation such as ELRA. SAM-IAT will need to be easily obtainable, well controlled, stable, well supported and documented with a future that is perceived to be secure.

The first and second roles differ in their requirements. The first requires that standard test conditions must be made available to all those doing assessment in a controlled way so that standards can be maintained. The second requires the use of test conditions that adequately reflect the operating environment. In both cases test conditions must be reproducible. Without reproducible conditions, no scientific results can be provided.

8.1.1. SWOT ANALYSIS FOR THE SPEECH INPUT ASSESSMENT TOOLS

Strengths - Technically advanced, strong theoretical basis, well known in the European speech community, high user satisfaction, possible ELRA backing.

Weaknesses - Multiple versions, difficult to configure, poor documentation of current versions, poor hardware specification, software effort required to implement driver for new ASR systems, not currently productised or regulated.

Opportunities - To dominate a small but accessible marketplace.

Threats - Major potential customers will develop or continue to use their own assessment tools. This threat will be greatly increased if the SAM standards are not successfully promoted by the EU. Ownership is not established by ELRA, uncertainty and confusion arises as to the future of the tools and their supply.

9. Models for the Distribution and Regulation

9.1. Discussion

The control and distribution of the SAM-tools will depend on the intentions of the European Union with regard to the promotion of the speech technology communities tools and standards.

If the EU decides that it does not want to promote speech assessment standards then it is adequate for the SAM-IAT to be loosely controlled by the participants in the SAM projects with free access to modify and extend the tools software in a self-regulatory framework. This approach will require little input from bodies such as ELRA. It will also allow those currently modifying code to continue developing their tools as new speech technology and testing procedures are devised with no reference to outside agencies. The number of available versions being used will tend to increase and the ability to easily cross reference between laboratories will tend to decrease over time. The non-standard tools that we have at present are attractive to their users in that they can be modified locally to fix any bugs or problems and can be used without reference to other users. In this task they appear to be very successful with users expressing a high degree of satisfaction.

In practice, the SAM standards are dependant on the SAM tools because the application of the SAM standards can only be achieved using the SAM tools designed for the purpose. The SAM tools however, can implement whatever assessment procedure is required by the user. If the SAM assessment standards are to be promoted as widely as possible then the SAM tools must be similarly promoted but in a standard form that ensures reproducibility and consistency. The implication of this for the production of new versions of the software is that it must be strictly regulated by the body responsible for upholding the assessment standards. Moreover the currently fragmented SAM tools must be brought together into a definitive version.

Prospective users also require the tools to be standard as the survey described in the section ‘ Survey of prospective users of the SAM input assessment’ suggests. Potential users were only interested in using SAM-IAT if it supported widely recognised and impartially administered assessment standards. This would suggest that the successful distribution of SAM-IAT will depend on having a more rigorous software management system in place to ensure that only authorised versions are released. Any confusion as to which version embodies the latest standard would damage the prospect of its widespread use. Successful distribution will also depend on the vigorous promotion of assessment standards by a body such as ELRA.

Described here are three models for the distribution and regulation of the SAM-IAT. These models could also be applied to the other SAM-tools.

9.2. Model 1-self regulation

The self-regulation model is appropriate where adherence to a single standard is less important than technical flexibility and where the number of anticipated users remains low.

Those who have the SAM-IAT would continue to modify and distribute software without regulation. One suggested method of distribution would be via ftp servers. Users would continue to modify code and post their modifications to the ftp sites for other users to copy and modify further. Issues of ownership and support on an ad hoc basis would have to be agreed between users, but inevitably this may lead to the tools finding their way into the public domain. Self regulation would also lead to the increasing divergence of versions, with increasing confusion as to which embodied the most appropriate assessment scheme.

9.3. Model 2-self regulation + management by the European Language Resources Association (ELRA)

This model would be suitable for a medium sized community of users where a single standard is important and consensus can be reached as to which is the definitive version for distribution.

In this model the ELRA management committee would be responsible for authorising versions of SAM-IAT as embodying the assessment standards. Modified versions would have to be resubmitted to ELRA for testing. Modifications would be regulated by a revision board appointed by the ELRA management committee. Distribution could be achieved by establishing a single ELRA ftp site where authorised versions would be posted by ELRA. The freedom of access to the ELRA ftp site could be controlled by password if this was considered necessary. This solution would allow for multiple versions of the SAM-IAT with only their minimum function being regulated.

The adoption of this model would not in itself lead to the rigorous promotion of SAM-IAT. The possibility of multiple versions may even discourage use outside the community of current users. However, if free access to the ELRA ftp site was allowed then this might facilitate wider use.

9.4. Model 3-control and distribution by European Language Resources Association (ELRA)

The models so far presented need little or no input from ELRA. Model 3 will require a more active role for ELRA. This model is appropriate where the intention is that SAM-IAT is promoted and distributed to as wide a user-base as possible. Successful promotion will require good, impartial management of the SAM-IAT to ensure that the product is of merchantable quality and adheres strictly to the assessment standards. It will also require the appointment of a distributor and software house to maintain and update the product. The inclusion of a distributor driven by a profit motive will help to ensure rapid and wide spread take up.

The ELRA management committee will procure the SAM-IAT source code and documentation by issuing a request to all those that have modified SAM-IAT since SAM-A. ELRA must also establish rights to the original SAM-IAT code. A third party will then be commissioned to harmonise the current software versions into a single definitive revision incorporating all the features added since SAM-A. The harmonisation process should be

conducted under the direction of a revision board convened by the ELRA management committee and the distributor appointed by the chief executive. ELRA should initiate an appropriate software management system to ensure quality is maintained. The initial 'productisation' and launch will require initial funding.

It would make sense for the distributor to be the same distributor responsible for the distribution of the ELRA corpora. Although this would not be essential, because the SAM-IAT and the ELRA corpora are closely associated, customers for the IAT are also likely to be customers for corpora, sales effort would be focused and costs would be minimised. The distributor would represent the end-users on the revision board, providing valuable information on changing customer needs.

The cost of maintaining the product to the standards required by customers may be high. Ultimately, maintenance and modification would be authorised by the revision board and costs born by ELRA. The degree to which ELRA's costs were recouped from the sale of licences to the distributor would depend on the success of the product in the marketplace.

If the promotion of the SAM-IAT and assessment standards is successful then it would become in the interests of suppliers to produce and maintain drivers for their particular input device and they would be motivated to comply by pressure from the marketplace.

THIS PAGE LEFT INTENTIONALLY BLANK

10. The relationship with other standards

10.1. National Institute for Standards in Technology

The National Institute for Standards in Technology (NIST) in the USA has for some years published software that performs certain standard evaluation tests for ASR. The NIST software performs standard tests on results presented in a standard format. It incorporates no ASR evaluation management system. A previous version of SAMPAC incorporated a version of the NIST tests in its scoring module. The current NIST software has not been incorporated.

It is of benefit to maintain compatibility with the NIST software. Comments have been made by some SAM users that the SAM_SCOR scoring routines are in some aspects superior to the NIST tests. To maintain compatibility it is sufficient to provide the facility to export test results in a format readable by the NIST software. The NIST software is designed to be run under UNIX and most users interested in running NIST software will have access to UNIX machines. It is therefore not necessary for SAMPAC to include the NIST routines. This will reduce support costs and ease maintenance.

10.2. Bellcore Voice Input Technologies Technical Analysis Specification

Bellcore has published a report [6] specifying detailed procedures for the assessment of suppliers' Voice Input Technologies. In the report, the responsibilities of the 'co-ordinator' (usually Bellcore) to conduct the testing procedures, results data, to provide the speech input assessment software and a database of speech data is clearly set out. The responsibilities of the supplier to provide the device to be tested and software to communicate with the assessment software using a documented protocol are also described in detail.

It is obvious from the Bellcore document that by having firm control on the assessment procedures and software development, Bellcore has been successful in obtaining consistency and standardisation. However, the documentation of scoring metrics appeared to be quite trivial and lacked the theoretical background of the SAM-IAT.

THIS PAGE LEFT INTENTIONALLY BLANK

11. Conclusions

The SAM-IAT is a unique assessment system and is being used to successfully perform technically advanced performance assessments on a wide range of speech input devices. Users find it very useful and have a high degree of satisfaction with its performance.

In the future the tools will need:

- a good system of software management and quality control for example the European Space Agency system.
- a clear idea of who the customers for the SAM-IAT are and what they will use the SAM are to be used for - quality control, evaluation, comparison etc.
- an appropriate mechanism for distribution and support - ftp site, commercially motivated distributor etc.

The successful management of the tools is a key factor for their success in the future. The European Language Resources Association (ELRA) is an obvious candidate for performing this role. Although, however well the tools are managed and however good their performance, ultimately the overriding issue effecting the tools is - how vigourously and how widely will the speech assessment technology standards be promoted? The potential users outside the small group currently using the SAM-IAT only see the need for them if they are backed up by strong, well supported standards. Here again the role of ELRA is crucial.

THIS PAGE LEFT INTENTIONALLY BLANK

12. References

1. Report I2 “Review of SAM tools and software development strategy” SAM-A Periodic Progress Report Year 1 October 92 - April 93.
2. Report I3 “Initial Specification of Recognisers, Assessment, Methodology and Database requirements”, SAM-A Periodic Progress Report Year 1 October 92 - April 93.
3. Report I6 “Hardware and software requirements specifications” SAM-A Periodic Progress Report Year 1 April 93-Sept 93
4. Report I7 “Initial Specification to update SAMPAC”, SAM-A Periodic Progress Report Year 1 October 92 - April 93.
5. Report I8 “Description of unified architecture, data structures and user front-end concept” SAM-A Periodic Progress Report Year 1 April 93-Sept 93
6. Bellcore “Voice Input Technologies Technical Analysis Specification” Special Report SR-TSV-002242. issue 1, March 1992.
7. ‘Software Engineering Standards’, C.Mazza, J.Fairclough, B.Melton, D.De Pablo, A. Scheffer & R.Stevens, Prentice Hall Int. (UK) Ltd. ISBN 0-13-106568-8.

THIS PAGE LEFT INTENTIONALLY BLANK

Appendix 1- Modifications by DRA Farnborough³

Background

SAMPAC V3.10 will run tests of the ASR1000 with lists of isolated digits. These may be run in ISOLATED or CONTINUOUS modes, with the recogniser PTR set ON permanently during the test. Information other than the word label and score is lost. The ideal test for the ASR1000 would allow the use of the PTR as operated by the subject during the recording, so that the recogniser could update its gain and noise mask settings continually, as it would in normal operation. For scoring, a DP algorithm should be run within each phrase, and the recogniser's syntax should be set to a pre-determined node before each phrase is spoken.

Outline of changes to SAMPAC

- Define a new operating mode, PTR and add a parameter START_NODE to the configuration file.
- Alter the ASR1000 driver to copy all output to the "label" string without processing it.
- Write a new function to write the contents of the MemoryFile into a new output file, .RES, when in PTR mode.
- Modify proc_test_continuous() to call the new write function instead of WriteIRFFile() and WriteScoreFile() when in PTR mode.

Add a new mnemonic to the CTL file to run ASRTOSAM after each list is tested.

These changes assume that the SAMPAC/ASR1000 interface will reset the syntax after each phrase is output, by detecting the GAIN and MASK line. The initialize_recogniser() function of the ASR1000 driver will send the required syntax START_NODE to the interface.

³ based on 'Definition of modifications required to SAMPAC V3.10 in order to enable tests of ASR1000 in continuous mode with PTR operating' by A J South MMIA.

Details of changes

Source File	Function	Changes
var.h		Add ASR to command mnemonics. Define PTR as 'P'.
cmd.h		Add ASR to command mnemonics.
manager.c	proc_ctl_file()	Add case ASR to the switch parsing the CTL file.
c_test.c	proc_test_continuous()	Add pointer to RES file. Add code to generate RES filename, open the file and write header, etc. Add code to run writeRESFile() instead of writeIRFFile and writeScoringFile when in PTR mode.
recog.c	Recog()	Modify the last two lines so that the label can be extracted from ASR1000 output lines when in PTR mode.
check.c	checkMode()	Modify to allow PTR mode.
	checkCtlFile()	Modify to allow PTR mode and ASR mnemonic, the latter only in PTR mode.
wrscore.h		Add function prototype for writeRESFile.
wrscore.c	writeRESFile()	Write new function.
asr.c	set_par()	Add code to interpret PTR mode and START_NODE parameter.
	initialize_recognizer()	Add code to send START_NODE parameter to interface.
	get_recognizer_response()	Modify to copy whole line to "label" without interpreting it. Change L_LINE to 80.
asrtosam.c		Remove output of word number during operation.

Files required for testing the software

ASR_P.CFG

TEST.CTL

Suitable training files,

Two TST files from Tornado trials with corresponding signal files having 8 kHz PTR tone added.

20 July 1994

Extra changes as a result of de-bugging

Source File	Function	Changes
var.h		<ul style="list-style-type: none">Define P_T_R as 'P' because PTR is already used in the vv lib.
	c_test.cproc_test_continuous()	<ul style="list-style-type: none">Over-write InitFile and EndFile with 0 and length of file when in PTR mode, so that whole file is replayed. scan_file() function is still called (first) so that TST file pointer is correctly positioned. I decided to leave in the code to write the IRF.
recog.c	Recog()	<ul style="list-style-type: none">The positions in the string "buf" of the first character and the start of label needed to be 1 and 32 respectively, instead of 0 and 31 as expected. I don't know why.
manager.c	proc_ctl_file()	<ul style="list-style-type: none">Alter "case MOD:" to include PTR mode. Declaration of "log_f" changed; cannot assign to an array in declaration inside a function definition in C5.1.

25 July 1994

More changes to incorporate ADDTONE into MANAGER

Source File	Function	Changes
manager.c	proc_ctl_file()	Add "case ADT:" to specify path and filename if required and whether to keep or delete the TST and SIG files created. Call adtone() and then change tst_f to new name. Add "case ASR:" to specify log file name and run ASRTOSAM as a spawned process.
check.c	checkCtlFile()	Test that ADT follows TSF and comes before RTF. Check that ASR follows RTF. Check that both only occur in PTR mode.
adtone.c	adtone()	Produce new version to work as a function in "manager". Will be given output file name rather than construct it, and will output progress messages to monitor window. Return statements to have suitable values. Separate definitions into a header file.
manager	make file	Include adtone compilation and linking with other modules of manager.

1 August 1994

Extra changes introduced during implementation

Source File	Function	Changes
adtone.c	adtone()	Modified to start the output file 1.25 seconds before the first PTR. The TST file UTS start and finish times are modified accordingly. This reduces the file size slightly and allows part of a file to be used if required.
asrtosam.c	main()	Modified to write time and date of run into log file.

9 August 1994

Bug fixes

Problem with mis-alignment in continuous mode was caused by SAMPAC taking <CR> as end of line whereas ASR1000 outputs <CR><LF> at end of each line. This meant that the <LF> was taken as the first character of the next line, and hence start time was the same as the end time of previous line. Fixed by changing the #define END_OF_MESSAGE to '\n' in RDKERN.C and re-compiling ASRDRV.

There is another "#define END_MESSAGE 0x0D" in RS232.C! I left this one as it was. Why are these definitions not in the header file RS232.H? One definition could then be common to both modules.

The sscanf format statement at the bottom of Recog() (which extracts the label for display on the Monitor window during continuous or isolated mode) used a list of characters [A-z0-9 etc] which C5.1 did not interpret properly. I changed this to a simple %s format.

22 August 1994

Change to ASR1000 Driver

ASRDRV was changed to output a '\n' after the start node sent to the ASR1000/SESAM interface. This was done so that the interpreter could more easily determine when the start node message was complete.

5 December 1994

More Bug-fixes

SAMPLING FREQUENCY

As a result of investigations into testing and calibration of OROS boards, it was found that IAU2DRV loaded the division ratio into the AU21 incorrectly. The set_dac() function of IAU2DRV was therefore modified by making it load (1-ifreq) instead of (-ifreq). Also, a bug in main() caused it to crash if /o1 was specified in the command line. The code that interprets the switch set osamp0 to "0" instead of "1" which caused a divide by zero error. This was also fixed.

As a result of getting the correct sampling frequency, the FRAME_TIME definition in ASRTOSAM could be returned to its expected value of 16.0.

16 February 1995

END_OF_MESSAGE CHARACTER

When ASRTODBF was tried on RES files generated by SAMPTR it did not work. The RES files were found to have two CRs and an LF at the end of each line. This was because the bug-fix in WRSCORE which added a "\n" character (ie CR LF) to the end of line should have added a linefeed only. It was found that an LF character was always interpreted as CR LF, so extra code had to be added to copy the string to a temporary buffer and remove the first CR before printing it to the file with a '\n' on the end. This worked.

22 Feb 1995

Appendix 2- Modifications by CSELT Turin

BACKGROUND

SamPac was officially released as version 3.20 on April 1992. After that date it has been updated but it has not been any more released to third partners due to the premature death of the SAM-A project.

The last CSELT version is the number 4.30! It was derived from the 3.20 after a lot of updates to fix bugs and add new features.

NEW FILE STRUCTURE

A lot of files were in the 3.20 version of SamPac but some modules weren't appropriate as divisions. In the same time some confusion there was between the "manager" files and the recognizer files.

The version 4.30 has redistributed the SamPac files in the directory structure below:

```
C:-----+- SAMPAC +- COMMON
      |
      |
      +- MANAGER +- INCLUDE
      |   |
      |   |
      |   +- LIB
      |
      |
      +- IAU2DRV
      |
      |
      +- TOOLS
      |
      |
      +- recognizer driver #1
      |
      ...
      |
      +- recognizer driver #n
```

fig. 1 - SamPac directory structure

To use such structure users have to had in the INCLUDE and LIB environment variables the C:\SAMPAC\COMMON path to have full visibility of *.h and *.lib files contained in that directory. For example using Microsoft C v 6.00 they have to set the following environment variables.

```
SET INCLUDE = C:\C600\INCLUDE;C:\SAMPAC\COMMON
SET LIB     = C:\C600\LIB;C:\SAMPAC\COMMON
SET PATH   = %PATH%;C:\C600\BIN;C:\C600\BINB;C:\SAMPAC
```

The advantages of such structure will be more clear in the following paragraph, because now we have to different software that are strictly linked each other but really independent.

SAMPAC LIBRARY

The COMMON subdirectory will contain the SAMPAC.LIB library used by all the recognisers and DAC drivers, plus the some include files with common declarations (defines and function prototypes). The files included in COMMON are:

```
+-----+-----
| makefile | Microsoft nmake file to build SAMPAC.LIB
+-----+-----
|         |
| rdkern.h | Recogniser driver kernel: it contains only the function
| dac.h    | driver() used as interface between manager and the
| interrpt.h | recogniser drivers. rdkern.h contains the function
| rddefs.h | prototype and the other three include file the
|         | definitions used to exchange commands and answers
| rdkern.c | between drivers (both REC and DAC drivers) and
|         | manager
|         |
+-----+-----
|         |
| rs232.h  | Functions designed to RS232 connections:
|         | rprintf(), rgets(), clear_rs232(), irq4(), reset_it4(),
| rs232.c  | set_reset_it4(), init_rs232(), times(), rs232_putch(),
|         | rs232_getch(), rs232_kbhit()
|         |
|         | Have to be used only if users need serial connection
|         |
+-----+-----
```

```

|      |
| tsr.h | Functions designed to Terminate and Stay Resident
|      | program:
|      | rscanf(), rfgets(), already_loaded(), prog_size(),
| tsr.c | install(), make_resident()
|      |
+-----+-----+
|      |
| ctr.h | Function designed to write messages on the screen
|      | directly from the resident drivers:
| ctr.c | put_char(), put_str(), scroll(), clear_window(),
|      | init_window()
|      |
+-----+-----+
|      |
| rec.h | Function prototypes to be implemented by recogniser
|      | designers:
|      |
|      | initialize_recognizer(), train_recognizer(),
|      | test_recognizer(), get_recognizer_response(),
|      | end_continuous_test(), set_rec_par()
|      |
|      | This functions are called by the driver() function
|      | included in the module rdkern.c. The driver()
|      | function is called via interrupt software by the
|      | manager program
|      |
+-----+-----+
|      |
| sampac.lib | Library containing all functions described before to be
|      | linked with designers supplied functions.
|      |
+-----+-----+

```

As users can see some functions have been moved from the original file to some other file and some new functions, mainly to debug purpose have been written. The main advantage of this structure is that now we have a library that does not need to be recompiled during recogniser driver developing, but only linked. This allow a more quick developing time.

The new debug functions allow to see the driver during running directly over a manager window, reserved for recogniser messages. The previous release of SamPac allowed to see only the RS232 exchanged strings but now we can display anything we want!

The SAMPAC library lives of own life, so we can use a different version number that the one used for MANAGER, but at the moment we use the same number.

MANAGER

The manager programs have been restructured too. Currently we have the following files

```
+-----+-----+
| makefile | Microsoft nmake file to build MANAGER.EXE
+-----+-----+
|         |
| rdriver.h | Recogniser driver interface.
| rdriver.c | It contains all function used by manager to call
|         | the recogniser drivers (REC driver).
|         |
+-----+-----+
| ddriver.h | Digital to Analog driver interface.
| ddriver.c | It contains all function used by manager to call
|         | the DAC drivers.
|         |
+-----+-----+
|         |
| var.h     |
| gest_win.h | General functions to manage screen outputs
| gest_win.c |
|         |
+-----+-----+
|         |
| check.h   | Functions designed check files existence before
| check.c   | start a test
|         |
+-----+-----+
|         |
| train.h   | Function designed train a recogniser
| train.c   |
|         |
+-----+-----+
|         |
| i_test.h  | Function designed test a recogniser in isolated
```

```

| i_test.c | mode
|         |
+-----+-----+
|         |
| c_test.h | Function designed test a recogniser in continuous
| c_test.c | mode
|         |
+-----+-----+
|         |
| timer.h  | Function designed to manage a more speed time.
| timer.c  |
|         | This functions were originally inside the recogniser
|         | drivers, but this will bring a lot of troubles; for
|         | example if users remove the recogniser driver from
|         | memory the timer tick had to be slow down using an
|         | appropriate program. Now all this works are performed
|         | automatically by manager itself.
|         |
+-----+-----+
|         |
| var.h    | Main module of MANAGER
| cmd.h    |
| trn_tst.h | It call all the other modules based on the mnemonics
| manager.c | read in the manager control file
|         |
+-----+-----+

```

I think that this restructuration of modules allows users to understand much better its functionality. The Vermont Views calls are hidden inside the gest_win.c file, so we can change that calls with other ones to free the software from rights problems. All VV functions are in the directories \SAMPAC\MANAGER\INCLUDE and \SAMPAC\MANAGER\LIB and they are not used in any other module, so we don't need to extend their visibility using the environment variables.

IAU2DRV

The TSR module have been unified with the one used in the recogniser driver; now both driver use the same routines in TSR.C contained in SAMPAC.LIB. Further research could bring us towards a better tsr routines more efficient, and both recognizer and dac driver will gain a lot of from them.

Some bugs have been fixed, but the click problems have not been solved yet; in fact it is very difficult to solve, because it does not depend by a bug in the software but, as explained in [4], it depends only on the value of the first and last samples sent to the I/O device.

When we play a token, usually the first sample is different from zero, so the signal level rise up from zero generating the attach click; conversely when we stop playing an utterance the signal level drop down to zero producing the detach click. The OROS driver tries to avoid the detach click, retaining the last sample of an utterance, but unfortunately the OROS firmware reset to zero the A/D converter before playing the next token.

The first solution proposed in [4] have been investigated, fading the signal by using a ramp for example. The signal rise slowly from zero to the level of the first sample and drop down slowly from the level of the last sample to zero. Unfortunately this produces other extra noises because we introduce other signals at high frequency. I tried several times by using different rising signal (linear, logarithmic, hamming windows, ...) but without appreciable results. A more complex signal analysis is needed to solve the problem using this technique.

The second solution proposed in [4] is to re-label the speech database to find the zero-crossing point near the original end-points. This solutions have not been investigate, but it is more simple (maybe except for speech databases with a strong offset).

TOOLS

Some missing tools now have been developed. All these tool present the same user interface, they run directly from DOS command line and they use wildcard characters to read label files plus a new facility to run using a list of files as input (indicated by the prefix char '@'). The software is written in C language and it can easily updated. Command line help allows naive users to get in touch easily with them.

a) extract

The first tool allow users to generate test and training files reading directly speech database label files without any DBMS. A lot of command line

switches (documented as comment rows in the generated file) allows to generate files in several different ways.

b) scopy

Another pieces of software allows users to copy speech files from a disk to another (usually from a network disk to an local hard disk); it reads training/test files and copies only speech files really used in them. At end of tests it can also delete files from hard disk and load some other ones.

c) sconvert

A tools have been developed to convert speech files between some format (PCM 16 bits, mu-law 8 bit, A-law 8 bit)

d) analyze

Finally another scoring tools have been developed. Advantages in respect to SAM_SCORE, that is still a valid tool, are the direct result file scoring, the generation of scoring matrixes (min, ave, and max scores), and the capabilities to generate new test files based on results of previous test (for example including all utterances with wrong results)

NEW FUNCTION IN SAMPAC V 4.30

a) MOD: mnemonic

The MOD: mnemonic now appear only in the control file (*.CTL) and a new function - set_rec_par() - have been added to communicate some parameters to recogniser drivers; currently it pass the MOD: mnemonic and its value to the recognize drive. This free users to write the same configuration files both for continuous mode test and isolated mode test.

b) New test modality

The isolated test now can be run in two different way. In the old mode the recogniser were wake-up to recognize before that the DAC driver started to play (I called that mode as NARROW ISOLATED MODE)

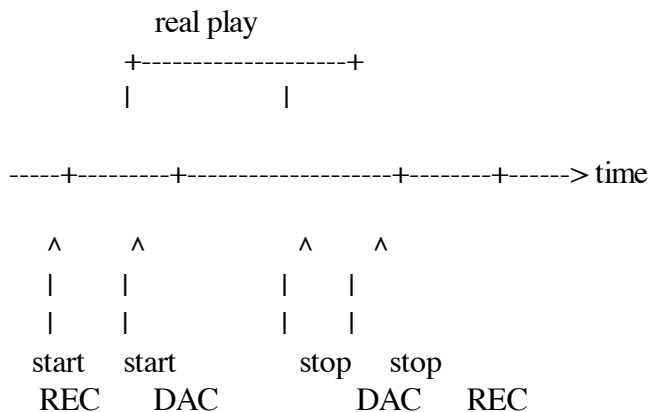


fig. 2 - Narrow Isolated Test

In the new mode (called BROAD ISOLATED TEST) before start the play, and after the recogniser. This allow avoid that artificial silence will help the end point module, but we need enough real silence before the utterances in order to don't cut the stimuli.

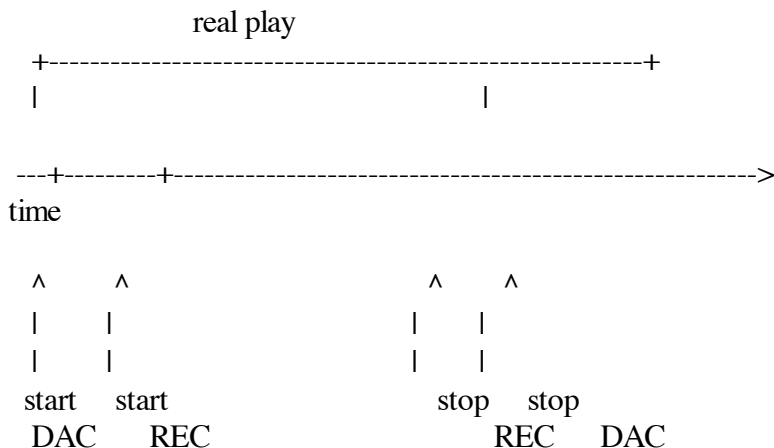


fig. 3 - Broad Isolated Test

Using this new test modality we simulate much more better that before the real use of isolated words speech recogniser.

Finally because the BROAD/NARROW switch is independent from the MOD switch user can have both the original NARROW CONTINUOUS TEST and the BROAD CONTINUOUS TEST.

The words NARROW and BROAD derives directly from the segmentation type of the labels files, but if more appropriate other definitions can be used (suggestions are welcomed).

c) Timer ticks managed directly by manager

With the 4.00 version of SamPac the timer tick functions have been moved from the recogniser driver to the manager program. This simplify developers works and make environment more robust. Now recogniser drivers have to ask to manager the right address of memory word were is the speed-up timer by using the new function put_time_counter_addr() completely contained in the driver() function of rdkern.c. Users are no more be worried about it that it is completely transparent.

d) Batch mode

Now manager can be included in batch files specifying the switch -p at run time. This switch disables the "Press any key" at end of a single run.

e) Dummy training

For recogniser that can't be trained the switch -t allows user to avoid play during training; actually recogniser drivers need training of the labels in order to use every kind of case in tests (both lower, upper or mixed case).

This is not a big improvement but speed up the developing time!

f) Adaptive recogniser testing

The test_recognizer() now pass to the recogniser driver also the label of the utterance under test; this allows to test adaptative recognisers.

THIS PAGE LEFT INTENTIONALLY BLANK

Appendix 3- Modifications by Alcatel - Milan

REQUESTED INFORMATION HAS NOT BEEN SUPPLIED

THIS PAGE LEFT INTENTIONALLY BLANK

Initial Distribution

- 1 Speechdat project managment
- 2 File - DRA/CIS/374/016/02

THIS PAGE LEFT INTENTIONALLY BLANK

Report documentation page

DRA Reference: DRA/CIS/374/016/1.0	Originator: D.N.L.HOWELL,	MOD Contract:	MOD Customer:
Classification: Unlimited	Issue Date: October 1995	Total Pages: Title + x + 49	References: 7
Title: The future of the SAM tools			
Title Classification: Unlimited	Abstract Classification: Unlimited	Conference: N/A	Translation: N/A
Author: D.N.L.HOWELL		Key Words:	
Abstract: A review of the current state of the SAM multi-lingual speech input/output assessment tools and ways of supporting them in the future. The Speechdat corpora and the SAM tools are fundamental and essential resources for those working to keep European speech and language technology advancing ahead of its competitors. The future of the SAM tools is vital to the successful use of the Speechdat corpora and to the promotion of European standards of assessment.			

