IST-2003-511598 (NoE)

COGAIN

Communication by Gaze Interaction

Network of Excellence
Information Society Technologies

D2.2 Requirements for the Common Format of Eye Movement Data

Due date of deliverable: 31.08.2005
Actual submission date: 05.09.2005

Start date of project: 1.9.2004 Duration: 60 months

De Montfort University

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Bates, R., Istance, H., and Spakov, O. (2005) ***D2.2 Requirements for the Common Format of Eye Movement Data.*** Communication by Gaze Interaction (COGAIN), IST-2003-511598: Deliverable 2.2.
Available at http://www.cogain.org/reports/

Main author: Richard Bates (DMU)

Main contributors: Howell Istance (DMU)
Oleg Spakov (UTA)

Thanks to: Päivi Majaranta (UTA)
John Paulin Hansen (ITU)
Mick Donegan (ACE)
Lisa Oosthuizen (ACE)
The members of WP2

Table of Contents

List of Figures

List of Tables

Executive Summary

This deliverable builds on previous work surveying the 'de-facto' standards in gaze-based communication systems (COGAIN Deliverable 2.1 by Bates et al., 2005) by bringing the diverse and incompatible 'de-facto' standards found in that survey together into outline COGAIN standards for gaze-based communication systems. The main body of work follows the same approach as the survey deliverable by examining the complete chain enabling gaze-based interaction and communication, starting from the hardware and progressing through to the software interface. This approach was adopted to give rapid and simple reference between the two documents (Deliverables D2.1 and D2.2). The standards and requirements outlined in this document are multi-level to include as full a range of eye-tracking systems as possible, with basic standard levels that most systems may adhere to, through to open-ended advanced levels of functionality and standards that may only be applicable to advanced or even future systems. This 'compliance to level' approach allows the uptake of COGAIN standards by nearly all manufacturers, encouraging compliance as far as possible, and also setting out future standards to lead and inspire manufacturers to work toward a common goal of a single high-level standard most suited for the users of gaze-based communication systems.

A single gaze communication standard

The deliverable presents a draft set of COGAIN standard interfaces based on multi-level standards that encompass diverse interfaces from many gaze-tracking systems. This approach of a standardised interface greatly reduces the need for third parties, who wish to develop software to aid gaze-based communication, to develop multiple separate and incompatible software versions each dedicated to a particular system. Instead, this deliverable presents methods to produce a single interface, enabling the development of gaze-based communication in the most effective and flexible way for all users. It is hoped that such a single, simple, open and commonly held standard will give the benefit of furthering the choice, acceptability and uptake of gaze based systems for end users.

Implementation of standards

The next steps from this deliverable are the implementation of examples of these standards in the real world. It is hoped that this task will form the next deliverable from WP2 (COGAIN Work Package 2: “Standardisation”), with the implementation of the basic COGAIN wrapper for gaze-based communication systems.

Future COGAIN standards

This deliverable outlines the requirements for standards to enable gaze-based communication systems to work on a more 'plug and play' approach. This work will be continued with future outline standards for gaze-based environmental control, and gaze-based mobility control. By producing standards in these areas, users of gaze-based interaction may not only use gaze for communication, but also for environmental and mobility control with the same freedoms and ease that is enjoyed by users of more common devices such as hand mice or joysticks – this is the role of COGAIN.

Agreement of WP2 members

The aim of his deliverable was based on meetings with WP2 members. Agreement was found on the content of WP2 as follows:

- Work should be based on data archiving, streaming and API control and interfacing.
- The COGAIN standards should work toward a single common standard.
- The COGAIN standard should exist with increasing levels of compliance.
- The WP2 members should work toward a practical implementation of the standards.
- The practical realisation would form a 'wrapper' or common interface between differing systems and the COGAIN standard interface.
- A research retreat will be held to discuss formally the findings of D2.1 and D2.2, leading to and informing D2.3.

1 Introduction

1.1 The aim of COGAIN WP2[1]

Work Package 2 “Standardisation” within COGAIN is directly aimed at improving the existing gaze-tracking systems' incompatibility situation essentially by introducing standards for eye communication on several levels. These standards will start from basics such as defining the format in which the trackers provide the measured data, to higher level interfacing such as establishing recommendations for the protocol used for interfacing with graphical user interfaces and environmental control systems. By the end of the COGAIN project, it is hoped that these standards will be adopted by suppliers and manufacturers and included as part of their in-house operating standards, thus allowing a mix and match approach to gaze-based interaction system construction, freeing end users to choose any hardware and any software from any manufacturer to form a combination that best suits their needs in the same way that users of more mainstream technology can already do today.

As found in previous work surveying the 'de-facto' standards in gaze-based communication systems (COGAIN Deliverable 2.1 by Bates et al., 2005), at present gaze driven tools that wish to fully integrate with eye-tracking systems must use specially written drivers specific to each eye-tracking system they wish to integrate with. In addition, gaze driven tools may wish to communicate between each other, such as the communication required between a dwell click tool preferred by a user that is from one manufacturer and the selection of keys by dwell on an on-screen keyboard from another manufacturer. This gives rise to multiple communication paths between differing eye-tracking hardware and gaze driven tools, and between differing gaze driven tools, with each new addition requiring a new dedicated and specially written driver to allow communication. This complex situation is illustrated in Figure 1.

Here, for example, gaze driven tool 'A', perhaps an on-screen keyboard, has been designed to operate and enable control over two different eye tracking systems, 'C' and 'D'. To enable this control the gaze driven tool must use two differing interfaces, one compatible with each of the different eye-tracking systems. In addition, the gaze driven tool may also wish to communicate with a second gaze driven tool 'B'. To enable this communication, yet another interface dedicated to that tool is required. Clearly, as the number of proprietary and non-compatible systems increases, so the number and complexity of the drivers required to sustain communication between the many systems and tools increases to an unmanageable level. This is compounded by the high cost of producing many differing interfaces and drivers to enable this communication, leading to manufacturers almost certainly not producing these interfaces. This lack of communication for using eye tracking as an input method is a significant block for implementing gaze-enabled software components that could be widely adopted by the user community.

[1] This first section (section 2.1) is an updated reprise of the introductory notes of COGAIN Deliverable 2.1 (Bates et al., 2005) in order to refresh the reader as to the aims of both that deliverable (D2.1) and this deliverable (D2.2) as both deliverables are closely associated and lead to a final milestone in this area (M2.1 “Draft standards for eye movement data formats and eye tracker APIs”). Throughout the document, D2.1 refers to the Deliverable 2.1 by Bates et al., 2005.

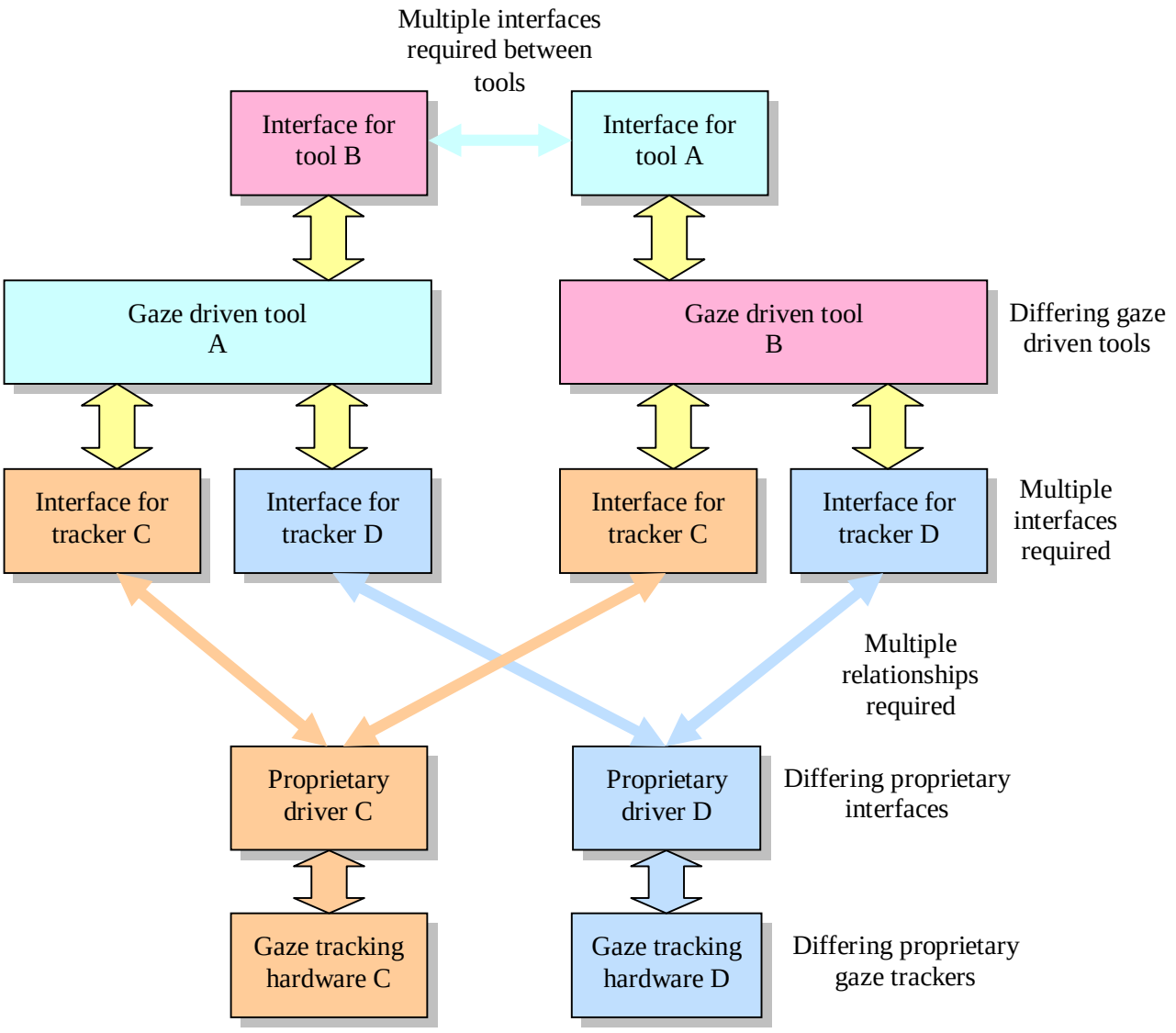

Figure 1. Current complexity of inter-device communication

A far better solution could be provided by adherence or inclusion of a COGAIN 'Virtual Device' standard, with all gaze-tracking hardware and gaze driven tools providing and sharing a single open standard interface (Figure 2). Here all gaze-tracking systems and gaze driven tools share a single common communication standard, with each system simply sharing data with any other system via the common standard. This results in manufacturers being only required to implement a single additional interface and driver (additional to their current in-house standard drivers) that would be compliant with the COGAIN standard. This would allow gaze driven systems, both hardware and software to enjoy the same flexibility of user choice as other systems such as normal desktop mice, making gaze communication as simple as other 'plug and play' devices.

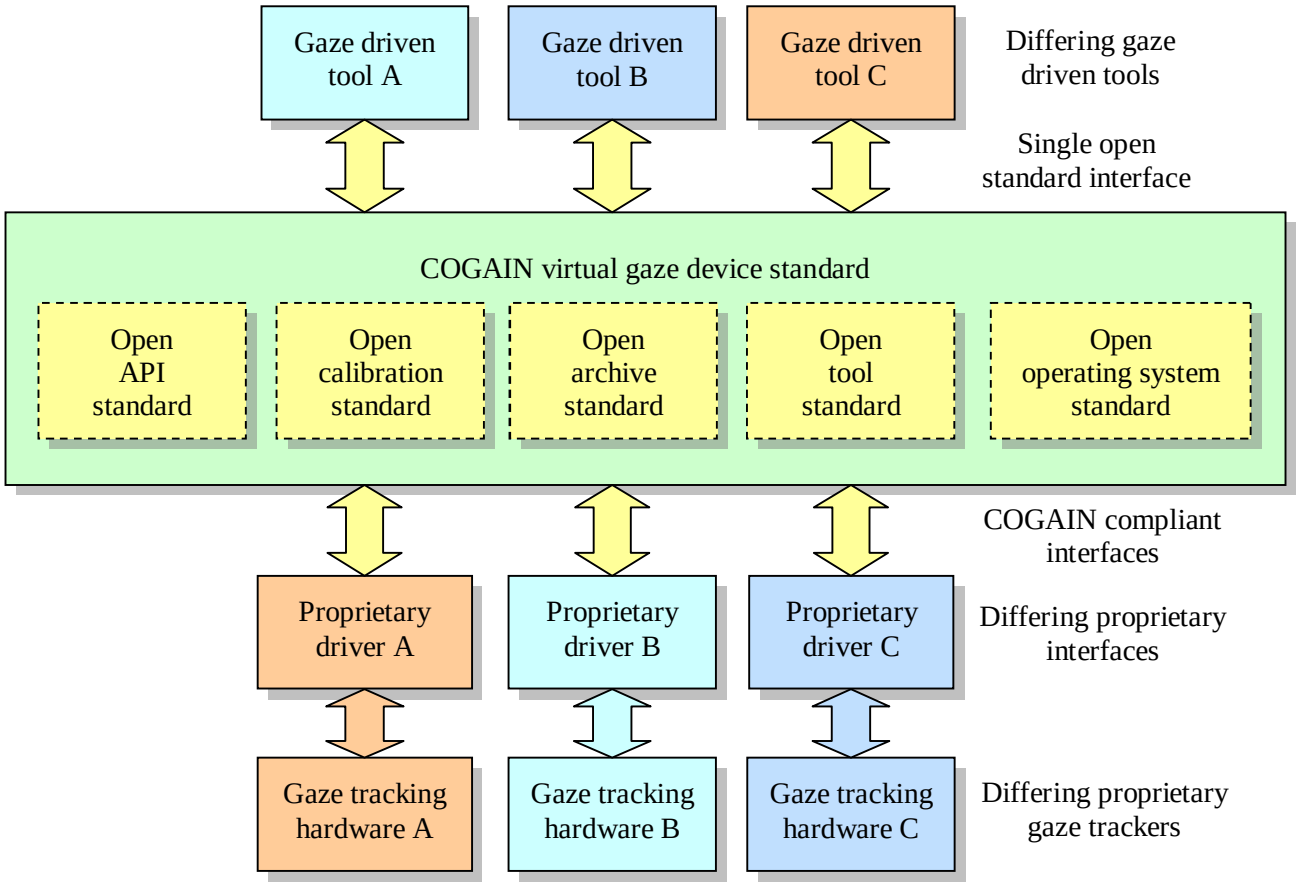

Figure 2. Enabling open communication with a COGAIN standard

1.2 This deliverable

As the previous survey of existing standards found (D2.1), at present choosing and using a gaze communication system does imply commitment to a given eye-tracking device from a certain manufacturer, with gaze tracker and operating software forming a tight partnership that is incompatible with other manufacturers' systems. This incompatibility between systems greatly restricts choice for users of gaze communication system and those who wish to develop systems to aid gaze-based communication, with developers forced to produce new and differing versions of the same application, each being suitable to work with a different manufacturer's system. This approach of producing many dedicated versions of the same application is costly to third party developers and deters the development and uptake of useful software systems suitable for gaze-based communication.

As discussed in the previous survey (D2.1), this existing situation is understandable for several reasons. To date, the main application area of gaze trackers has not been human-computer interaction: their development history origins from medical and psychological diagnostics and experiments (Jacob and Karn, 2003). Consequently, gaze trackers are not traditionally designed for compatibility with other systems and applications. This is an unfortunate situation, as this lack of inter-system compatibility results in end users having only a limited range of choice of systems and applications. To address this situation and give users choice over mixing and matching differing gaze-tracking systems with third party gaze driven applications in ways best suited to their needs, a single commonly held interface standard must be produced.

This deliverable (Deliverable 2.2) presents recommendations for levels of a single commonly held standard to 'wrap' differing manufacturers systems interfaces and translate these differing interfaces into a single COGAIN 'virtual' gaze standard interface as illustrated in Figure 2. This approach allows third parties to develop applications in a single version that conforms to a single COGAIN standard interface, but which may be gaze driven by a wide range of multiple, differing and previously incompatible gaze-tracking systems. By this approach, COGAIN hopes to free third parties to concentrate on development for end users, rather than concentrating on ever changing compatibility problems with many differing manufacturers.

1.2.1 Areas for standards

The deliverable follows the approach of the first deliverable (D2.1) and approaches the generation of standards by breaking down the operation of eye-tracking equipment into six main areas of functionality (Figure 3):

1. Methods of gaze tracking – the hardware type
2. Physical interface – the physical connection between gaze tracker and computer
3. Calibration – method of gaze calibration
4. Data Archiving – format of archived gaze data
5. Data Streaming – format of on-line gaze data
6. Application Program Interface – methods and properties for controlling the system

Figure 3. Areas of gaze tracking system functionality

Each of these six areas was chosen to allow simple examination of any possible standards within given function areas of gaze-tracking systems that can exist separately from the other properties. For example, the standard for gaze tracking (the hardware) can exist separately from the standard of data archiving (saving in a file) of the tracked gaze data – one may be changed without influence on the other.

Each of these six potential standards areas will be addressed in turn, first by a brief description of the area, and then followed by the outline of possible draft standards in that area.

1.2.2 Levels of standards

'Levels' of standard are used in the deliverable to indicate the desirability of compliance to ascending levels of standards. At a basic level, any operational area of eye tracking equipment that complies with, or is denoted with, a level of 'zero' is considered as undesirable for use by the users that the COGAIN network is serving. This may be because the systems are simply unsuitable for use by people with these high levels of disability, or because the systems do not offer the required levels of functionality to make them useful for gaze-based interaction and communication. A categorisation of 'Level 1' denotes basic compliance with a standard that is deemed acceptable for use at a basic level, and is the minimum requirement to be recommended by COGAIN. Increasing levels, such as 'Level 2' or 'Level 3', indicate increasing desirability of functionality as provided by the systems. Hence, the higher the level accorded a system attribute, the higher the level of standard that the system has complied with.

The physical standards section of this document addresses the first three areas for standards (Figure 3) where 'levels' are used to indicate simple steps of desirability with no attempt to indicate detailed standards, as these would be inappropriate. The software standards section of this document addresses the remaining three areas for standards (Figure 3). Here 'levels' are used in some detail and the standards defined for each level of functionality must be complied with to achieve the accreditation of achieving any given level, with any deviation from the standard defined for a level being unacceptable if the standards are to achieve their full usefulness.

1.2.3 Most relevant and important standardisation areas

From the survey identifying the main operational areas of eye-tracking equipment (Figure 3), the following pages of the deliverable will show that the most important areas for standards of these areas are within the software section, the areas 4, 5, and 6: Data Archiving, Data Streaming, and the Application Program Interface (API). The reader is particularly encouraged to examine the sections of this deliverable within the software section that address standardisation of these software and communication based areas.

2 Draft Physical Standards

2.1 Methods of gaze tracking

From the survey of deliverable D2.1 there were seven main methods of gaze tracking currently found (Figure 4):

1. Video oculography – pupil and corneal reflection
2. Video oculography – pupil only
3. Video oculography – dual Purkinje image corneal reflection
4. Video oculography – limbus, iris-sclera boundary
5. Electro oculography – electro-potential about the eye
6. Electromagnetic – scleral coil in the eye
7. Contact lens – contact lens in the eye

Figure 4. Methods of gaze tracking

2.1.1 A standard for hardware

Examining the survey of D2.1 (D2.1, Table 2, 'survey of gaze-tracking methods') found that of these methods (Figure 4), only two methods were commonly used: video oculography and electro oculography. Examining a method for applying a level of standards to the differing hardware used by the manufacturers of these systems found no practical standard for hardware construction, such as number of cameras, number of light sources, number of electrodes etc, which could be applied between differing equipment that would have any useful meaning. Just because one system used two or three light sources for pupil illumination, would this be better, or comply with a 'higher' level of standard than a system that used only one light source? It was not possible to determine this by examining hardware configuration alone. A better approach was to examine the operating qualities of the systems in terms of the user, and the output, of the systems.

The key factors for the users when operating the hardware were the invasiveness of the systems to the body of the user, and the output of the systems in terms of tolerance to movement of the user during operation, temporal resolution of the systems, and the claimed accuracy of the systems. Here levels of standards could be outlined.

2.1.2 A standard for physical intrusion

Clearly, as discussed above and in D2.1, only minor and preferably no intrusion onto the body of the user is desirable. Here basic levels of standards may be defined. Clearly, any intrusion into the eye of a user, particularly if they are motor disabled and perhaps could not remove the device themselves, is unacceptable. Minor intrusion onto the skin is acceptable and is used in practice. However, no intrusion at all is most desirable. In this case it can be argued that shining moderate (or high) levels of infrared light or other artificial light onto/into the eye of a user is also intrusion as it may dry the eye and cause longer-term discomfort. Hence, perhaps the most desirable level of intrusion is a passive observation of the eye with no artificial illumination, apart from that ambient light present in a room. These basic levels of standard are outlined (Table 1):

Standards for physical intrusion		
Desirability	Standard 'Level'	Description
Acceptable++	3	No physical intrusion, no exposure to IR or artificial light onto/into the eye, ambient light only
Acceptable+	2	No physical intrusion, exposure to IR or artificial light into the eye
Acceptable	1	Intrusion onto the skin/face
Unacceptable	0	Intrusion into the eye

Table 1. Basic standards for intrusion

Examining the impact of the standards on the system types (Figure 4) shows that video oculography pupil only may comply with the highest level standard (provided ambient only light is used), video oculography with corneal reflection can only achieve Level 2 as intrusive light is used to generate the corneal reflection, electro oculography intrudes onto the facial skin and can only achieve Level 1 (still acceptable), but the other methods are deemed unacceptable due to their intrusion.

Examining the survey of gaze-tracking methods used by motor disabled users (from D2.1, Table 2) we find that no systems comply with Level 3, eleven comply with Level 2, and one to Level 3. Thus, the standards categorisation allows these systems to improve and move to higher levels. It is of note that work is ongoing within the COGAIN network to develop Level 3 eye-tracking systems.

2.1.3 A standard for head movement tolerance

Examining the results for tolerance of head movement and working distance (from D2.1, Table 2) found that the gaze-tracking systems had head movement tolerances ranging from approximately a 4 cm volume to a 30 cm volume, with tolerance in the z axis considerably less than the x and y axes. Clearly, it is desirable to have as large a tolerance of head movement/head position as possible as this gives the greatest flexibility for the end user. Although some users may have little or no personal head movement due to disability, the ability to 'roll up and use' the systems is essential. This might mean users are placed at differing head positions between sessions of use, hence to enable the systems to be used without third part assistance the systems must be capable of accommodating differing user head positions. In addition, some users may have involuntary (compulsive) head movements due to the disability (for more information, see COGAIN Deliverable 3.1 by Donegan et al., 2005). These basic levels of standard are outlined in Table 2.

Standards for head movement tolerance		
Desirability	Standard 'Level'	Description (x,y,z volumes)
Acceptable+++	4	Fully free head movement tolerated.
Acceptable++	3	Minor head movement tolerated. Equal or greater than 25cm x 25cm x 10cm volume.
Acceptable+	2	Minor head movement tolerated. Equal or greater than 10cm x 10cm x 5cm volume.
Acceptable	1	Minor head movement tolerated. Equal or greater than 5cm x 5cm x 0cm volume.
Unacceptable	0	No head movement tolerated. Less than 5cm x 5cm x 0cm volume.

Table 2. Basic standards for head movement

Examining the survey of gaze-tracking methods used by motor disabled users (from D2.1, Table 2) we find that no systems comply with Level 4, one complies with Level 3, three with Level 2 and 3 two to Level 3. Thus, the standards categorisation allows these systems to improve and move to higher levels.

2.1.4 A standard for temporal resolution and accuracy

Examining the results for temporal resolution and spatial accuracy (from D2.1, Table 2) found temporal resolutions ranging from 15 Hz to 120 Hz with spatial accuracies ranging from 2.0º to 0.45º visual angle at the eye. Again, it is desirable to maintain as high a temporal resolution and spatial accuracy as possible, as higher performance gives greater fidelity to the actual gaze movement of the user. As before, these basic levels of standard are outlined (Tables 3 and 4):

Standards for temporal resolution		
Desirability	Standard 'Level'	Description (Sample rate Hz)
Acceptable+++	4	Highly responsive, >120Hz
Acceptable++	3	Responsive, 100Hz – 120Hz
Acceptable+	2	Moderately responsive, >30Hz, <100Hz
Acceptable	1	Slowly responsive, 10Hz – 30Hz
Unacceptable	0	Not responsive, < 10Hz

Table 3. Basic standards for temporal resolution

Standards for spatial accuracy		
Desirability	Standard 'Level'	Description (Degrees visual angle)
Acceptable+++	4	Highly accurate <0.25º
Acceptable++	3	Accurate 0.25º - 0.5º
Acceptable+	2	Moderately accurate 0.5º - 1.0º
Acceptable	1	Slightly accurate 1.0º - 2.0º
Unacceptable	0	Not accurate >2.0º

Table 4. Basic standards for spatial accuracy

Again, examining the survey of gaze-tracking methods used by motor disabled users (from D2.1, Table 2) we find that no systems complied with Level 4 on both temporal and special standards, thus as before the standards categorisation allows these systems to improve and move to higher levels. Further, one system complies with Level 3 on both counts, with most of the remaining systems complying with Level 2, and two complying with Level 1.

2.2 The physical interface

From the survey of deliverable D2.1, there were five methods of physical link between gaze-tracking equipment and a standard personal computer currently found in use by eye gaze system manufacturers (Figure 5).

1. Serial port (RS232C standard port)
2. Universal serial bus (USB I or II standards)
3. Firewire (IEEE-1394 serial port)
4. Analogue signal (simple plugs and sockets)
5. Dedicated card or box (dedicated card within PC)

Figure 5. Methods of physical link

Each of these methods of linking gaze tracker to personal computer may carry data that is proprietary to the manufacturer of the gaze tracker (for example in-house binary data formats) and require a dedicated manufacturer's driver and hence be inaccessible for use by a third party system, or of a standard open type (for example ASCII codes) and hence the link data could be accessible by a third party system (Figure 6).

1. Closed (requires proprietary driver to access the data)
2. Open (data is of a standard form and can be accessed)

Figure 6. Methods of physical link data

2.2.1 A standard for the physical interface

It is desirable for the both the physical link and the data it carries to be as universally open, or accessible, as possible. This may be defined as the freedom and independence of the link and data from any manufacturer's dedicated in-house standards. So for example a physical link that comprises a commonly used format (such as USB or serial port for example) must be more desirable as it can be connected to any format of user's computer, and must be regarded as more useful and flexible than any link that requires the use of a proprietary

card or interface that may or may not be easily fitted to any user's computer. In addition, the data is most universally available if the data is in a commonly accepted format (such as ASCII for example) rather than in a manufacturer's own standard that may not easily be read and utilised. This approach is reflected in the standard for the physical interface (Table 5).

Standards for the physical interface		
Desirability	Standard 'Level'	Description (Link and data type)
Acceptable+	2	Open: standard commonly used link, open data format
Acceptable	1	Partially open: manufacturer's dedicated card or link, open data format
Unacceptable	0	Closed: manufacturer's dedicated card or link, closed data format

Table 5. Basic standards for physical interface

The results of the survey (D2.1, Table 2) indicated that industry-wide only two systems achieved a Level 2 standard, with all other systems exhibiting closed data formats (requiring a driver from the manufacturer to decode the data) and so being deemed unacceptable. If the manufacturers could make available 'open' data formats and physical connections for these systems by following the lead of COGAIN (Work Package 2), then third party manufacturers could access these systems, giving more choice to end users.

2.3 Calibration

From the survey of deliverable D2.1 the methods of calibration currently found in use by the manufacturers of eye gaze systems were broken down into categories dependant upon the possibility of varying the number of targets on the screen from the manufacturers pre-set number (Figure 7), and the possibility of varying the positions of the calibration targets on the screen from the manufacturers pre-set locations (Figure 8):

1. Fixed (A single number of targets on the host personal computer screen)
2. Variable (A range of target numbers on the host personal computer screen)

Figure 7. Number of calibration points

1. Fixed (Targets are pre-set and fixed on the host personal computer screen)
2. Variable (Targets are pre-set but can be changed on the host personal computer screen)

Figure 8. Layout of calibration points

2.3.1 A standard for calibration layout

The possibility of greater flexibility in the number and layout of calibration targets on the screen allows more, or less, accurate mapping of gaze to screen vector over the whole area of the screen. Thus fewer targets give a poorer mapping but more rapid and easier calibration, and greater numbers of targets giving a higher quality of mapping but with slower and more demanding (of the user) calibration. Each has its own advantages and disadvantages, depending on the capabilities of the end user and the requirements of the task to be driven by gaze. The ability to alter calibration target numbers and positions is valuable to users who may have some difficulty using the complete screen, or have difficulty calibrating a particular portion of the screen. Such

problems can be addressed by adding, moving or clustering calibration points to such problem areas on the host computer screen, hence the greatest flexibility in placement and number of points is most desirable. This approach is reflected in the standard for the physical interface (Table 6).

Standards for the calibration screen		
Desirability	Standard 'Level'	Description (Points number and layout)
Acceptable+	2	Variable layout, variable number
Acceptable	1	Variable layout, fixed number OR fixed layout, variable number
Unacceptable	0	Fixed layout, fixed number

Table 6. Basic standards for the calibration screen

The results of the survey (D2.1, Table 2) indicated that four systems achieved a Level 2 standard, with all other systems achieving only an unacceptable standard. It is hoped that manufacturers will be encouraged to give greater flexibility in calibrations in the future.

3 Draft Software Standards

3.1 A language for gaze data

Software standards allow a great flexibility for defining standards for gaze-based systems. Unlike hardware based standards, or perhaps recommendations for compliance and functionality, software standards can use greater flexibility and data manipulation methods to bridge the standards gaps between differing incompatible systems. This section continues the 'levels' compliance based approach applied to hardware, but extends this into a language for gaze-based data exchange. Briefly introducing the three fields within this section of data archiving, data streaming, and interaction with differing application programming interfaces, these three fields have the commonality of all being software based systems that carry data and it's associated methods together.

This suggests that a method for bringing differing software systems together would be common data structures and an associated descriptive language for the data. The approach adopted is one of using an existing and popular standard that carries and enables data exchange between differing platforms. Methods for achieving this for fixation based gaze data archiving and analysis have suggested an entity (what is the data) and data (the data) approach[1,2].

The language for gaze data proposed here is based on this approach with a more formal structure based on XML (Extensible Markup Language) as defined by the World Wide Web Consortium (W3C)[3]. A definition and possible example are shown (Figures 9 and 10):

Definitions[1]:

```
String – a string of characters in ASCII
Integer - an integer in ASCII
Data - the data to be conveyed, in either integer or string
// Comments
<> entity identifier
</> end of the scope of an entity
{a, b, c} a list of valid data values
```

Figure 9. Definitions of COGAIN gaze language

Example:

```
// a gaze sample {x,y} screen coordinates for the left eye:
<eye>left</eye>
<screen>
<x> 255 </x> <y> 255 </y>
</screen>
```

Figure 10. Possible example of COGAIN gaze language

[1] http://www.cs.uoregon.edu/research/cm-hci/VizFix/
[2] http://www.cs.uta.fi/~oleg/light/icomp.htm
[3] http://www.w3.org/TR/2004/REC-xml-20040204/

As can be seen from the definitions and example, the language is small, simple, extendable and flexible. It would also easily allow translation software to be written to convert existing formats into this common format via simple parsing utilities. Where elements are not present in the data, these can simply be ignored as they will not then be parsed, likewise, if newer elements are introduced, older software will simply not parse these new elements, thus preserving forward and backward compatibility.

XML is based on ISO 8879 and was originally designed to meet the challenges of large-scale electronic publishing. However, it is playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere, where it has proved to be both flexible and widely applicable. XML files and documents are made up of storage units called entities (which can be used to define the following data), which contain either parsed or unparsed data (the gaze data we wish to convey). The parsed data is made up of characters, some of which form character data, and some of which form markup data (which can be used to define the data structure or storage layout). COGAIN suggested standards for 'COGAIN XML extensions for gaze' are thus defined where appropriate in the following sections.

3.2 Data archiving

Data archiving is defined as the method and formats by which gaze-tracking systems record and save the data captured by the hardware of the system for later examination and analysis. Data archiving is an important function of gaze-tracking systems when used as input pointing devices to systems as it allows later 'off-line' and repeatable examination of how the system performs with an end user; giving insight into the suitability of the system for that user.

As before, from the survey of deliverable D2.1 it was found that some systems did not offer any data archiving capabilities, and of those that did there were two methods or formats of archived gaze data used by gaze tracking system manufacturers (Figure 11). These were archiving data in a closed form using a manufacturers propriety standard such that the data could only be read by that manufacturers own software and applications and was closed from third party software access, and archiving data in an open form that could be read, such as a text file, by any third party software.

1. None (No data archive capability)
2. Closed (Binary manufacturers propriety standard)
3. Open (Text ASCII public standard)

Figure 11. Methods of data archiving

3.2.1 A standard for data archiving availability

Clearly, the desirability of the system features is related to the availability of the data, with more available archived data being more desirable. This can be summarised in standards for data archive availability (Table 7).

Standards for data archive availability		
Desirability	Standard 'Level'	Description ('Openness' of data system)
Acceptable+	2	Open, ASCII text or similar format data
Acceptable	1	Closed, binary, manufacturer's propriety standard
Unacceptable	0	No data archive capability

Table 7. Basic standards for data archive availability

The methods of data archiving can be further broken down by the data rate, or sampling rate, used to record the system data (Figure 12).

1. Archive data rate (samples archived per second)
2. Fixed data rate (invariant sample rate)
3. Variable data rate (variant with time)

Figure 12. Methods of archive samples

3.2.2 A standard for data archiving sample rate

Examining the archive sample rates, it is desirable to obtain as high a rate as possible of data for archiving, thus allowing the maximum possible detail of analysis to take place on the data during any post-recording processing. A possible excess of data samples can be regarded as non-harmful as no real-time processing would occur with a data archive, with post-processing possible in non-time critical conditions. The desire for high data rates is also reflected in standards for temporal resolution (Table 3).

Time variant data sampling and recording, or variation in the data rate from a fixed interval may cause processing overheads, and difficulties in producing software and analysis tools that could easily step through data in equal and even intervals. Even with time stamps on samples, time variant recording is not as desirable as time invariant sampling and recording. These findings are reflected in draft standards (Table 8).

Standards for data archive availability		
Desirability	Standard 'Level'	Description (Variance of data rate and data rate)
Acceptable+++	4	Time invariant data rate, Standard Level 4 temporal resolution, >120Hz
Acceptable++	3	Time invariant data rate, Standard Level 3 temporal resolution, 100Hz – 120Hz
Acceptable+	2	Time invariant data rate, Standard Level 2 temporal resolution, >30Hz, <100Hz
Acceptable	1	Time invariant data rate, Standard Level 1 temporal resolution, 10Hz – 30Hz
Unacceptable	0	Time variant data rate

Table 8. Basic standards for data archive sample rates

Finally, the survey of the preceding deliverable (D2.1 Table 6) found that methods of data archiving were broken down into the types and ranges of data archived (Figure 13).

1. Index number (count)
2. Time stamp (ms)
3. Gaze (on screen coordinates x, y)
4. Pupil (size or image x, y coordinates)
5. Corneal reflection (size or image x, y coordinates)
6. Head position (x, y coordinates)
7. Other data (specified)

Figure 13. Methods of sample types

3.2.3 A standard for data archiving methods

A basic standard of levels of desirability for archiving methods would be complex, as once the basic gaze coordinates on the viewed screen are obtained, it would be difficult to determine which of any other data available would add more to the desirability of the system than any other additional data available. Thus, the basic standard has two levels (Table 9).

Standards for data archive methods		
Desirability	Standard 'Level'	Description (Data archived)
Acceptable+	2	Additional data - screen gaze point x,y coordinates, time stamp, index number, pupil size, head position etc.
Acceptable	1	Minimal data required - screen gaze point x,y coordinates

Table 9. Basic standards for data archive methods

Typically, each of the data archiving properties (Table 9) forms a header line in a text file containing the data samples, with the sample data then placed in columns and rows under each data header (Figure 14).

```
Sample#     PupilX      PupilY      ScreenX     ScreenY     PupilSize
1           200         250         400         545         23
2           201         246         402         544         23
3           204         240         412         538         21
```

Figure 14. Typical data archive format

This approach gives rise to incompatibilities between differing systems, as not all systems can record the same types of data with the same types and order of headers. In addition, there is little chance of adding or manipulating additional data into this form of data archive. For example, the file does not define the sample rate, and it cannot support binocular tracking, which is becoming more commonplace. Defining a highly rigid format for archive data files such as this header and rows format is undesirable. A more effective approach is to use a data archiving language based on a COGAIN XML extension within a standard ASCII text file.

3.2.4 A standard for data archiving protocols

Based on the methods found from the Deliverable 2.1 survey (D2.1, Table 6), the methods used for data archive by differing manufacturers were summarised (This deliverable, Figure 13). A XML type entity and data type description of these methods can easily be produced as follows (Figure 15).

Gaze data method
Index number

COGAIN entity
```
<index> data{integer} </index>
```

Gaze data method
Time stamp

COGAIN entity
```
<timescale> {min,sec,ms} </timescale>
      //measure in minutes, seconds, milliseconds etc, set once.
<timestamp> data{integer} </timestamp>
      // time stamp, must match the format of the timescale set.
```

Gaze data method
Gaze (on screen)

COGAIN entity
```
<screen>                          // set mapping reference to screen
<gaze>                            // set data source
<x> data{integer} </x>            // x,y coordinates
<y> data{integer} </y>
</gaze>                           // end of scope of data source
</screen>                         // end of scope of mapping reference
```

Gaze data method
Pupil location (on video image)

COGAIN entity
```
<video>                           // set mapping reference to video
<pupil.location>                  // set data source
<x> data{integer} </x>            // x,y coordinates
<y> data{integer} </y>
</pupil>                          // end of scope of data source
</video>                          // end of scope of mapping reference
```

Gaze data method

Corneal reflection location (on video image)

COGAIN entity

```
<video>                              // set mapping reference to video
<corneal.reflection>                 // set data source
<x> data{integer} </x>               // x,y coordinates
<y> data{integer} </y>
</corneal.reflection>                // end of scope of data source
</video>                             // end of scope of mapping reference
```

Gaze data method

Pupil size (on video image)

COGAIN entity

```
<video>                              // set mapping reference to video
<pupil.size>                         // set data source
<x> data{integer} </x>               // x,y coordinates
<y> data{integer} </y>
<d> data{integer} </d>               // or diameter…
</pupil.size>                        // end of scope of data source
</video>                             // end of scope of mapping reference
```

Gaze data method

Head position (in space)

COGAIN entity

```
<world*>                             // set mapping reference to world
<head.position>                      // set data source
<x> data{integer} </x>               // x,y,z coordinates
<y> data{integer} </y>
<z> data{integer} </z>
<r> data{integer} </r>               // roll,pitch,yaw coordinates
<p> data{integer} </p>
<y> data{integer} </y>
</head.position>                     // end of scope of data source
</world*>                            // end of scope of mapping reference
```

*Here for example the mapping reference could be relative to the world, the screen, the eye tracker etc, without changing the following language.

Gaze data method

```
Other data
```

COGAIN entity

```
<mapping reference>                  // set mapping reference
<data source>                        // set data source
<data1> data{integer} </data1>        // data
<data2> data{string} </data2>
</data source>                       // end of scope of data source
</mapping reference>                 // end of scope of mapping reference
```

```
Note: Here the flexibility of the language system shows that any data sources and types could be represented in any mapping reference and still be understood by any third part software reading the file.
```

Figure 15. Example data archiving protocols

An excerpt from a sample file to illustrate the operation of the standard can easily be generated (Figure 16).

```
<start>                              // start of file
<ident>1</ident>                     // Participant number
<date>31/08/2005</date>              // Date of the experiment
<cond>Dwell Click</cond>             // Experimental condition information
<cal%>82</cal%>                      // Calibration accuracy
<timescale>ms</timescale>            // measure in milliseconds
<time>01:30:04</time>                // Starting time of the experiment

// Here comes the data…

<screen>                             // set mapping reference
<gaze>                               // set data source
      <timestamp>0</timestamp>       // time stamp
      <x>200</x>                     // x coordinate
      <y>150</y>                     // y coordinate
      <timestamp>50</timestamp>
      <x>204</x>
      <y>152</y>
      <timestamp>100</timestamp>
      <x>208</x>
      <y>154</y>
</gaze>                              // end of scope of data source
</screen>                            // end of scope of mapping reference

// end of the data

</start>                             // end of the file
```

Figure 16. Example data archive file

3.2.5 Benefits and costs of the proposed archive standard

Comparing the two file formats (Figures 14 and 16) clearly shows the contrast between the two approaches. The currently existing format (Figure 14) is fixed and highly inflexible, but has the benefit of requiring little overhead of additional metadata (the column headings) to separate differing data. The proposed preliminary COGAIN standard (Figure 16) is considerable more verbose and less concise than the original format, requiring considerable metadata (the element identifiers), however, contemporary computing systems should have no problems dealing with this additional information.

The proposed format has the greatest benefit in its extreme flexibility, with any data being easily represented. In addition, this approach allows for complete extensibility and backward and sideways compatibility as additional future data types added will simply be ignored by older versions of parsing software reading the file, and any software that does not require all of the data can simply read the file and locate the elements the software requires whilst not requiring any knowledge of any other data elements.

3.2.6 Examples of useful extensions

The extensibility of the format can be illustrated by listing various extensions to the basic format that might be added at any time. If the file is written following the basic format and protocol illustrated (Figure 15), any additional data can be added (Figure 17).

```
<ident> integer </ident>      // Participant number
<date> string </date>         // Date of the experiment
<time> string </time>         // Starting time of the experiment
<cond> string </cond>         // Any experimental condition information
<name> string </name>         // For example, name
<age> integer </age>          // Age etc
<rate> integer </rate>        // Tracker sampling rate
<tracker> string </tracker>   // Tracker make/model/version
<cal%> integer </cal%>        // Calibration accuracy
<left> string </left>         // Left eye data
<right> string </right>       // Right eye data
```

Figure 17. Examples of useful extensions

3.2.7 Levels of archive standards

The proposed standard file format can be applied in increasing levels of standard compliance. Over and above the basic need for a file format to comply with the basic 'entity and data' format illustrated (Figure 15), the range of data the file carries can be subject to levels of standard as discussed previously (Table 9). In this case, using the proposed file format would give four initial levels of standard (Table 10).

Standards for gaze archive files		
Desirability	Standard 'Level'	Description (Data archived)
Acceptable++	3	Further archive data added that is not currently recorded
Acceptable+	2	Additional data that is currently recorded
Acceptable	1	Minimal data required - screen gaze point x,y coordinates
Unacceptable	0	File not conforming to COGAIN proposed standard

Table 10. Standard levels for COGAIN gaze archive files

Here (Table 10) a file conforming to the proposed standard that contains only mapped screen coordinates of gaze at a time invariant sample rate (the basic minimum required for meaningful data) would be regarded as a Level 1 file. By adding all of the additional data recorded by manufacturers at this time (as gained from the survey of D2.1 Table 6, and illustrated in this deliverable Figure 13) the file would be regarded as a Level 2 file. By adding additional useful data that would enhance the understanding of the content of the file, such as user/subject information as suggested (Figure 17), the file would be regarded as a Level 3 file as it contains a maximal amount of desirable information – regardless of whether any end-user of the file requires this additional data or not.

3.2.8 Summary of archive standards

This section has proposed a flexible XML type approach to data archiving, with a basic standard format for recording data into a file that can be read by any software, and that allows backward and sideways compatibility, together with unlimited possibilities for extension. The section also suggested the promotion of increasing levels of file content, from a basic form of data to a highly rich form of data, with the aim of encouraging the recording of the maximum amount of data in standard open-format files that may be accessed and used by any interested party.

These software standards could be easily implemented into the proposed COGAIN wrapper, which would then parse manufacturers archive files and convert them into the proposed COGAIN format. Finalising the archive standard and implementing an example of the wrapper archiving standardised files will be the subject of the next deliverable D2.3.

3.3 Data Streaming

Data streaming is defined as the method and formats by which gaze-tracking systems output and stream real time data captured by the hardware of the system for real time examination and analysis, and can be thought of as the 'real time' equivalent of data archiving (as discussed in Section 4.2).

From the survey of deliverable D2.1 (D2.1, Table 7) it was found that there were three methods or formats of streaming gaze data used by gaze-tracking system manufacturers (Figure 18). These were streaming data by mouse emulation (moving the cursor on the host personal computer to emulate a hand mouse function), and streaming data either via a software link (such as via a manufacturers provided dynamic link library or via a pipe or messaging) or via a hardware link.

1. Emulate mouse (move the mouse cursor directly on the host personal computer)
2. Software link (via an application internal software link on the same computer)
3. Hardware link (via a hardware external link from a different computer)

Figure 18. Methods of data streaming

3.3.1 A standard for data streaming availability

Clearly, as with data archiving, the desirability of the system features is related to the availability of the data with more available streamed data being more desirable. This can be summarised in standards for data stream availability (Table 11).

Standards for data stream availability		
Desirability	Standard 'Level'	Description ('Openness' of data system)
Acceptable+	2	Open, either hardware or software link
Acceptable	1	Restricted, mouse emulation only
Unacceptable	0	No data stream capability

Table 11. Basic standards for data stream availability

The survey of D2.1 (D2.1, Table 7) allowed the properties of the streaming links to be broken down into the methods of the data rate (gaze samples per second actually sent, not the actual baud rate of the link) (Figure 19). The data rate along the link could be either fixed and invariant in time, or variable in time due to the level of processing required by the system delaying the availability of samples.

1. Stream data rate (samples archived per second)
2. Fixed data rate (invariant sample rate)
3. Variable data rate (variant with time)

Figure 19. Methods of streaming sample rates

3.3.2 A standard for data streaming sample rate

Examining the streaming sample rates in the same way as data archiving sample rates, it is again desirable to obtain as high a rate as possible of data for the stream, thus allowing lowest possible latency of to be present in the system that would be utilising the streamed data. As with data archiving, any possible excess of data samples can be regarded as non-harmful as unwanted samples can simply be ignored. The desire for high data rates is reflected in standards for temporal resolution (Table 3).

Time variant data sampling and recording, or variation in the data rate from a fixed interval may cause processing overheads, and difficulties in producing software that could easily utilise the data in equal and even intervals. Even with time stamps on samples, time variant recording is not as desirable as time invariant sampling and streaming.

Finally, there are considerable benefits in adopting the same standards for data streaming as data archiving (Section 4.2), thus requiring few changes to third party software and applications that would either use data archives or data streams as their gaze input methods. This compatibility between archiving and streaming is also useful for trialling and prototyping gaze driven utilities or analysis tools, such that they can first be trialled with archived data, and then be used on streamed real-time data without altering the internal data handling procedures of the software.

These findings are reflected in draft standards (Table 12) that are the same as the standards for data archiving.

Standards for data streaming availability		
Desirability	Standard 'Level'	Description (Variance of data rate and data rate)
Acceptable+++	4	Time invariant data rate, Standard Level 4 temporal resolution, >120Hz
Acceptable++	3	Time invariant data rate, Standard Level 3 temporal resolution, 100Hz – 120Hz
Acceptable+	2	Time invariant data rate, Standard Level 2 temporal resolution, >30Hz, <100Hz
Acceptable	1	Time invariant data rate, Standard Level 1 temporal resolution, 10Hz – 30Hz
Unacceptable	0	Time variant data rate

Table 12. Basic standards for data streaming sample rates

Finally, the survey of the preceding deliverable (D2.1, Table 7) found that the methods of data streaming are broken down into the types and ranges of data streamed. In addition to the on-screen gaze coordinates of the end user, there may be additional data sent, as discussed earlier, such as pupil size, or the ability to start and stop the stream (Figure 20).

1. Index number (count)
2. Time stamp (ms)
3. Gaze (on screen coordinates x, y)
4. Pupil (size or image x, y coordinates)
5. Corneal reflection (size or image x, y coordinates)
6. Head position (x, y coordinates)
7. Start/stop stream (ability to start and stop data stream)
8. Other data (specified)

Figure 20. Methods of sample types

3.3.3 A standard for data streaming methods

In the same way as discussed previously for data archiving (Section 4.2), and to maintain compatibility with data archiving methods, only two levels of the basic standard for data streaming methods were adopted (Table 13).

Standards for data streaming methods		
Desirability	Standard 'Level'	Description (Data streamed)
Acceptable+	2	Additional data – start/stop stream, screen gaze point x,y coordinates, time stamp, index number, pupil size, head position etc.
Acceptable	1	Minimal data required - screen gaze point x,y coordinates

Table 13. Basic standards for data streaming methods

Examining how this data is transmitted along the data stream, typically each of the data streaming properties (Table 13) forms a single byte or word in the data stream, with the data stream typically using a header byte to indicate the start of the data packet. The gaze and other data then follow in an order typically particular to individual manufacturers.

An example of such as data stream is shown (Figure 21)[1].

```
Byte Description
1     Status (0 = normal, >0 = error condition)
2     Pupil diameter, most significant byte (0=loss)
3     Pupil diameter, least significant byte
4     Reserved by manufacturer
5     Point of gaze horizontal coordinate MSB
6     Point of gaze horizontal coordinate LSB
7     Point of gaze vertical coordinate MSB
8     Point of gaze vertical coordinate LSB
```

Figure 21. Typical data stream format

[1] From ASL Model 5000 data stream

As with data archiving (Section 4.2), this approach gives rise to incompatibilities between differing systems, as not all systems stream the same types of data with the same order of data bytes. Again, defining a highly rigid format for streaming data such as the illustrated (Figure 21) header byte followed by a defined order of data bytes representing differing metrics is undesirable.

A more effective approach is to use exactly the same approach as for data archiving, with a language based on a COGAIN XML extension within a standard ASCII data stream or ASCII data pipe. Note that the nature of the carrier of the data stream is not important if it can carry ASCII characters.

3.3.4 A standard for data streaming protocols

Based on the methods found from the Deliverable 2.1 survey (D2.1, Table 8), the methods used for data streaming by differing manufacturers were summarised (This deliverable, Figure 20). A XML type entity and data type description of these methods can easily be produced as follows (Figure 22). The methods are the same as those used for data archiving, but with the addition of a data stream start/stop command, and a data packet indicator, this is used to indicate the start and end of a set of data, and wraps all other entities. An illustration of how this method operates is shown in Figure 23.

Gaze data method

```
Index number
```

COGAIN entity

```
<index> data{integer} </index>
```

Gaze data method

```
Time stamp
```

COGAIN entity

```
<timescale> {min,sec,ms} </timescale>
      //measure in minutes, seconds, milliseconds etc, set once.
<timestamp> data{integer} </timestamp>
      // time stamp, must match the format of the timescale set.
```

Gaze data method

Gaze (on screen)

COGAIN entity

```
<screen>                          // set mapping reference to screen
<gaze>                            // set data source
<x> data{integer} </x>            // x,y coordinates
<y> data{integer} </y>
</gaze>                           // end of scope of data source
</screen>                         // end of scope of mapping reference
```

Gaze data method

Pupil location (on video image)

COGAIN entity

```
<video>                           // set mapping reference to video
<pupil.location>                  // set data source
<x> data{integer} </x>            // x,y coordinates
<y> data{integer} </y>
</pupil>                          // end of scope of data source
</video>                          // end of scope of mapping reference
```

Gaze data method

Corneal reflection location (on video image)

COGAIN entity

```
<video>                           // set mapping reference to video
<corneal.reflection>              // set data source
<x> data{integer} </x>            // x,y coordinates
<y> data{integer} </y>
</corneal.reflection>             // end of scope of data source
</video>                          // end of scope of mapping reference
```

Gaze data method

Pupil size (on video image)

COGAIN entity

```
<video>                          // set mapping reference to video
<pupil.size>                     // set data source
<x> data{integer} </x>           // x,y coordinates
<y> data{integer} </y>
<d> data{integer} </d>           // or diameter…
</pupil.size>                    // end of scope of data source
</video>                         // end of scope of mapping reference
```

Gaze data method

Head position (in space)

COGAIN entity

```
<world*>                         // set mapping reference to world
<head.position>                  // set data source
<x> data{integer} </x>           // x,y,z coordinates
<y> data{integer} </y>
<z> data{integer} </z>
<r> data{integer} </r>           // roll,pitch,yaw coordinates
<p> data{integer} </p>
<y> data{integer} </y>
</head.position>                 // end of scope of data source
</world*>                        // end of scope of mapping reference
```

*Here for example the mapping reference could be relative to the world, the screen, the eye tracker etc, without changing the following language.

Gaze data method
Other data

COGAIN entity

```
<mapping reference>                 // set mapping reference
<data source>                       // set data source
<data1> data{integer} </data1>      // data
<data2> data{string} </data2>
</data source>                      // end of scope of data source
</mapping reference>                // end of scope of mapping reference
```

Note: Here the flexibility of the language system shows that any data sources and types could be represented in any mapping reference and still be understood by any third part software reading the file.

Gaze data method
Start/stop stream

COGAIN entity

```
<stream.start>                      // start stream
</stream.start>                     // end stream
```

Gaze data method
Start/end data packet

COGAIN entity

```
<data>                              // start of data packet
</data>                             // end of data packet
```

Figure 22. Example data streaming protocols

From these definitions, an excerpt from a sample file to illustrate the operation of the standard and the data packet start and end indicators can easily be generated (Figure 23).

```
<stream.start>    // start the stream
                  // (sent to the originating eye tracker)

<data>      // Start of the data packet
            // (received from the originating eye tracker)

      <screen>                      // set mapping reference
      <gaze>                        // set data source
            <timestamp>0</timestamp>      // time stamp
            <x>200</x>                    // x coordinate
            <y>150</y>                    // y coordinate
      </gaze>                       // end of scope of data source
      </screen>                     // end of scope of mapping reference
</data>     // end of the data packet

<data>      // Start of the data packet
      <screen>                      // set mapping reference
      <gaze>                        // set data source
            <timestamp>50</timestamp>     // time stamp
            <x>205</x>                    // x coordinate
            <y>156</y>                    // y coordinate
      </gaze>                       // end of scope of data source
      </screen>                     // end of scope of mapping reference
</data>     // end of the data packet

</stream.start >  // end the stream
                        // (sent to the originating eye tracker)
```

Figure 23. Example data stream

Note that there is one difference in how the data is packaged between data archiving (Figure 16) and data streaming (Figure 23). During data archiving, the file is in a defined order with a defined start point in the file when data recording started. When opening the file, all of the required data is contained within the file hence the scope of the various entities describing the data will be read in the correct order. This is not so for data streams, where it is possible that the receiving software may start reading the stream after the stream has started. This is particularly true for eye tracking equipment that has no start/stop stream functions, and that continually streams. In these cases, it is necessary to indicate the logical entry points into the data stream where the scope of entities and their data may be followed correctly.

To overcome this problem, an additional entity '<data>' is used (Figures 22 and 23) to indicate the start and end points of logical data packets. Thus, any receiving software may simply wait until a '<data>' entity is found and then start reading the data within the packet.

3.3.5 Benefits and costs of the proposed streaming standard

Comparing the two streaming formats (Figures 21 and 22) clearly shows the contrast between the two approaches. The currently existing format (Figure 21) is fixed and highly inflexible, but has the benefit of requiring little overhead of additional metadata (the packet headings) to separate differing data. The proposed preliminary COGAIN standard (Figure 22) is considerable more verbose and less concise than the original format, requiring considerable metadata (the packet headers and element identifiers), however, contemporary computing systems should have no problems dealing with this additional information at the data rates currently generated by contemporary eye trackers.

As with data archiving, the proposed format has the greatest benefit in its extreme flexibility, with any data being easily represented. In addition, this approach allows for complete extensibility and backward and sideways compatibility as additional future data types added will simply be ignored by older versions of parsing software streaming the file, and any software that does not require all of the data can simply read the stream and locate the elements the software requires whilst not requiring any knowledge of any other data elements.

Finally, in addition to the existing data elements, any other elements may be added that are deemed to be useful. Examples of these have already been discussed for data archiving (Figure 17).

3.3.6 Levels of streaming standards

As with data archiving, the proposed standard streaming format can be applied in increasing levels of standard compliance. Over and above the basic need for a stream format to comply with the basic 'entity and data' format illustrated (Figure 23), the range of data the stream carries can be subject to levels of standard as discussed previously (Table 13). In this case, using the proposed streaming format would give four initial levels of standard (Table 14).

Standards for gaze streaming		
Desirability	Standard 'Level'	Description (Data streamed)
Acceptable++	3	Further streamed data added that is not currently streamed
Acceptable+	2	Additional data that is currently streamed
Acceptable	1	Minimal data required - screen gaze point x,y coordinates
Unacceptable	0	Stream not conforming to COGAIN proposed standard

Table 14. Standard levels for COGAIN gaze streaming

Here (Table 14) a stream conforming to the proposed standard that contains only mapped screen coordinates of gaze at a time invariant sample rate (the basic minimum required for meaningful data) would be regarded as a Level 1 stream. By adding all of the additional data recorded by manufacturers at this time (as gained from the survey of D2.1 Table 8, and illustrated in this deliverable Figure 20) the stream would be regarded as a Level 2 stream. By adding additional useful data that would enhance the understanding of the content of the stream, such as user/subject information as suggested (Figure 17), the stream would be regarded as a Level 3 stream as it contains a maximal amount of desirable information – regardless of whether any end-user of the stream requires this additional data or not.

3.3.7 Summary of streaming standards

In the same method as adopted for data archiving, this section has proposed a flexible XML type approach to data streaming, with a basic standard format for streaming data that can be read by any software, and that allows backward and sideways compatibility, together with unlimited possibilities for extension. The section also suggested the promotion of increasing levels of stream content, from a basic form of data to a highly rich form of data, with the aim of encouraging the streaming of the maximum amount of data in standard open-format streams that may be accessed and used by any interested party.

As with the data archiving formats, these software standards could be easily implemented into the proposed COGAIN wrapper, which would then parse manufacturer's streams in real time and convert them into the proposed COGAIN format. Finalising the stream standard and implementing an example of the wrapper streaming standardised data will be the subject of the next deliverable D2.3.

3.4 The Application Program Interface

The application programming interface (API) is defined as a software interface comprising a set of definitions of the ways in which one piece of computer software may communicate with another. One of the primary purposes of an API is to provide a set of commonly-used functions, or methods and properties for controlling the system.

An API consists basically of methods (what it can do) and the data those methods act upon (such as setting a calibration point). Typically, by defining what methods are available, the data that those methods operate on is also defined. As before, from the survey of deliverable D2.1, it was found that the methods of the API can be broken down into two areas: calibration methods and operational or running methods. These findings are summarised (Figures 24 and 25).

1. Start calibration (invoke the start of a calibration routine)
2. Accept calibration point (force the acceptance of the calibration for a point)
3. Stop calibration (halt the calibration)
4. Set number of calibration points (set how many calibration points are used)
5. Set calibration point position (set the screen location of a calibration point)
6. Set calibration type (set the calibration type/accuracy requirement used)
7. Load calibration (load a saved a calibration)
8. Save calibration (save a calibration)

Figure 24. Methods of calibration API

1. Start/stop data stream (start or stop data streaming from the gaze tracker)
2. Rate of data stream (change the sample rate of the data stream)
3. System state (get the current configuration of the gaze tracker system)
4. Camera tracking (enable/disable camera tracking/head tracking of the user)
5. Threshold of tracking (change the sensitivity of pupil/corneal reflection detection)
6. Mouse emulation (enable/disable hand mouse emulation)
7. Move/close application window (move/control the position of the gaze tracker window)
8. Other (other commands defined)

Figure 25. Methods of operation API

3.4.1 A standard for the API

Clearly, the desirability of the system features is related to the availability of control over the eye-tracking system via the API, with more control being more desirable (irrespective of whether individual applications use that enhanced control). This can be summarised in standards for eye tracker APIs (Table 15).

Standard levels for eye tracker APIs		
Desirability	Standard 'Level'	Description (Level of functionality)
Acceptable++	3	Enhanced functions (Greater than those in Figures 24 and 25)
Acceptable+	2	Basic functions (From Figures 24 and 25)
Acceptable	1	Minimal functionality – start/stop calibration and data streaming
Unacceptable	0	No API capability

Table 15. Basic standard levels for eye tracker APIs

Here (Table 15), any system that has no usable API can be regarded as unacceptable and rated at a zero level of standard. An API that provides the minimum useful functionality is rated as Level 1, and an API giving the current most common functionality (as summarised in Figures 24 and 25; from the Deliverable D2.1, Tables 9 and 10) can be regarded as acceptable and rated as a Level 2. Finally, any system with an API that exceeds the commonly found levels of functionality with additional features and functions can be regarded as a Level 3 API. Thus, the rating scheme of standards encourages manufacturers to produce APIs that provide the greatest levels of open control over their systems whenever possible.

3.4.2 Standards for calibration and control APIs

In the same way as discussed previously for data archiving and data streaming (Sections 4.2 and 4.3), only two levels of the basic standard for calibration and control API functionality were adopted (Tables 16 and 17).

Here (Table 16) an API that can invoke and cancel a calibration is rated as the minimum acceptable Level 1 of functionality, with an API that gives the basic set of calibration features (from Figure 24) a Level 2 standard, and an API with an enhanced set of functions a Level 3 standard. Again, for control (Table 17) an API that offers basic control is rated as the minimum acceptable Level 1 of functionality, with an API that gives the basic set of control features (from Figure 25) a Level 2 standard, and an API with an enhanced set of functions a Level 3 standard.

Standard levels for calibration APIs		
Desirability	Standard 'Level'	Description (Level of functionality)
Acceptable++	3	Enhanced functions – quality of calibration, drift re-centre, change targets to animations, calibration algorithms etc
Acceptable+	2	Basic functions – invoke/start/stop, accept point, set number of points, set point position, set calibration type, load/save calibration
Acceptable	1	Minimal functionality – invoke/start/stop calibration

Table 16. Basic standards for calibration APIs

Standard levels for control APIs		
Desirability	Standard 'Level'	Description (Level of functionality)
Acceptable++	3	Enhanced functions – full control over the tracker, control cameras, light output levels, tracking algorithms etc
Acceptable+	2	Basic functions – Start/stop mouse emulation, change data rate, get system state, on/off camera tracking, change tracking threshold, manipulate gaze tracker software window
Acceptable	1	Minimal functionality – start/stop gaze data stream

Table 17. Basic standards for control APIs

Examining how this data is transmitted via the API found that typically this was achieved by calling directly functions with the API and also by messaging via the operating system. In these methods, the API is included into or linked to the third party application, and functions are directly called and messages are passed back and form between the API and third party applications via specific handling functions and shared memory in the API and third party software. This is an effective approach but the function types, data types and message types are all specific to manufacturers, with no systems using the same function names, methods and data handling techniques.

An example of controlling via an API is shown (Figure 26)[1].

Retrieving a data sample:

```
EyeLinkGetSample(void Buffer*)   // Function to get gaze data
Buffer is an integer (16 bit)    // Pointer to memory
                                // buffer holding the data
Function returns 0 or 1                 // 1 indicates success
```

Setting a parameter:

```
EyeCmD_Printf("calibration_type=HV9"); // set calibration type
Type =                                          // choose type
      H3    // 3 point
      HV3   // 3 point poor linearization
      HV5   // 5 point
      HV9   // 9 point
Function does not return a value
```

Figure 26. Example API command formats

[1] From SMI model IViewX API handling

Clearly, designing and rewriting applications every time a new eye-tracking system emerges or is changed is unacceptable for third party manufacturers of gaze driven applications. Hence a standardised approach is required that would enable third party applications to only include and provide functions and messaging for one standard API.

3.4.3 A standard for API protocols

Based on the API commands and methods found from the Deliverable 2.1 survey (D2.1, Tables 9 and 10) it is possible to define draft standards for a COGAIN API that would 'wrap' existing differing manufacturer's APIs with a single simplified interface suitable for the needs of most third party application developers.

It is clear that at this stage, addressing the basic methods found in use from the survey of D2.1 (and summarised in Figures 24 and 25, and the data streaming requirements of Figure 20) is sufficient for the first COGAIN API wrapper. This would result in the wrapper achieving and passing on a Level 2 “Acceptable Level+” of functionality (from Tables 13 and 15) to third parties. As before with data archiving and data streaming, some effort is made to make the standardised API functions appear similar to the XML style language adopted in this deliverable for archiving and streaming, thus giving a high level of commonality and requiring minimal effort from third parties to learn and understand the wrapper interface[1].

From examining the required functionality (Figures 24 and 25), three basic data constructs are needed to fulfil a Level 2 (Table 15) API wrapper and the functionality it requires. These are to set a parameter on the API (such as the number of calibration points to show), to send a command to the API (such as to start calibration) and finally to receive data and messages from the API (such as gaze screen coordinates for example). These three required data constructs are outlined (Figure 27).

```
1.  Set/Get Parameter on API (string or integer data)
2.  Send Command to API (string or integer data)
3.  Get Message from API (pointer to data structure)
```

Figure 27. Required data constructs for the API

3.4.4 The API interface

When deciding on the API interface, looking at the required commands and parameters found from the survey of deliverable D2.1 and summarised (Figures 20, 24 and 25), it is clear that the parameter and command names for the API interface should be adopted from these summarised names, so for example, to start calibration, the command should be named “start.calibration” (from Figure 24), or to start data streaming the command should be named “start.stream” (from Figure 20). Thus, the API commands are most similar to the previous data archiving and data streaming syntax adopted by this deliverable, making the understanding of the API interface as simple as possible.

The outline format for the first two data constructs (Figure 27) is simple, and only requires naming and defining the parameter and command names, together with the allowed ranges of string and integer data and any associated constants that are sent to and valid within the API.

The third (Figure 27) data construct dealing with receiving data from the API is more complex as the API is required to send data to a third party application. Typically, this is achieved by passing pointers to shared data structures (the memory location of the data structure). This allows any number or size of data to be transferred easily, and this is the approach adopted. In order to maintain the approach of a single commonly held format

[1] Note that this deliverable is concerned with outlining the basic format of the API functions to wrap differing manufacturers APIs. The actual realisation of these standards for the API in terms of code is the task of the next deliverable, D2.3.

for archiving, streaming and the API interface as far as possible, the data constructs that are passed from the API can be made as close as possible in name and form to those used for data archiving (as illustrated in Figure 15) and data streaming (as illustrated in Figure 22). Hence, for example, the data structure for passing the gaze point on the screen would consist of a call to "screen.gaze" with a return structure consisting of two (x and y) integers (from Figure 22).

Assuming this approach, the following figures show the suggested basic API interface methods and data structures for data streaming, calibration and operation (Figures 28, 29 and 30).

Data streaming API

Gaze data method

```
Index number
```

COGAIN entity

```
set.index.number(integer)          // set starting index number
                                        // parameter=index number
                                        // returns 1=success, 0=failure
```

Gaze data method

```
Time stamp
```

COGAIN entity

```
set.time.scale(string)             // set time scale
                                        // parameter=min, sec, ms
                                        // returns 1=success, 0=failure
```

Gaze data method

```
Gaze on screen
```

COGAIN entity

```
screen.gaze(pointer)      // get screen gaze coordinates
                               // pointer to data structure
                               // returns 1=success, 0=failure
```

Gaze data method
Pupil location on video

COGAIN entity
```
video.pupil(pointer)      // get video pupil coordinates
                              // pointer to data structure
                              // returns 1=success, 0=failure
```

Gaze data method
Corneal reflection location on video

COGAIN entity
```
video.corneal.reflection(pointer)   // get video CR coordinates
                                        // pointer to data structure
                                        // returns 1=success, 0=failure
```

Gaze data method
Pupil size on video

COGAIN entity
```
video.pupil.size(pointer)      // get video pupil size
                                   // pointer to data structure
                                   // returns 1=success, 0=failure
```

Gaze data method
Head position in space

COGAIN entity
```
world.head.position(pointer)   // get head position coordinates
                                   // pointer to data structure
                                   // returns 1=success, 0=failure
```

Figure 28. The basic data streaming API interface

Calibration API

Gaze data method
Start calibration

COGAIN entity
```
calibration.start()          // start calibration
                             // no parameter
                             // returns 1=success, 0=failure
```

Gaze data method
Accept calibration point

COGAIN entity
```
calibration.point.accept(integer)   // start calibration
                                          // parameter=point number
                                          // returns 1=success, 0=failure
```

Gaze data method
Stop calibration

COGAIN entity
```
calibration.stop()           // stop calibration
                             // no parameter
                             // returns 1=success, 0=failure
```

Gaze data method
Set number of calibration points

COGAIN entity
```
calibration.point.number(integer)    // set calibration points
                                           // parameter=number of points
                                           // returns 1=success, 0=failure
```

Gaze data method

```
Set calibration point position
```

COGAIN entity

```
calibration.point.position(integer, integer, integer)
                  // parameter=point number, x on screen, y on screen
                   // returns 1=success, 0=failure
```

Gaze data method

```
Set calibration type
```

COGAIN entity

```
calibration.type(string)       // calibration type
                                     // parameter=type
                                     // returns 1=success, 0=failure
```

Gaze data method

```
Load calibration
```

COGAIN entity

```
calibration.load(string)       // load calibration
                               // parameter=string filename and path
                               // returns 1=success, 0=failure
```

Gaze data method

```
Save calibration
```

COGAIN entity

```
calibration.save(string)       // save calibration
                               // parameter=string filename and path
                               // returns 1=success, 0=failure
```

Figure 29. The basic calibration API interface

Operation API

Gaze data method
`Start data stream`

COGAIN entity
```
stream.start(string)     // set start stream
                               // parameter=string (port or handle)
                               // returns 1=success, 0=failure
```

Gaze data method
`Stop data stream`

COGAIN entity
```
stream.stop(string)            // set stop stream
                               // parameter=string (port or handle)
                               // returns 1=success, 0=failure
```

Gaze data method
`Rate of data stream`

COGAIN entity
```
stream.rate(integer)     // set stream rate
                               // parameter=integer (samples/sec)
                               // returns 1=success, 0=failure
```

Gaze data method
`System state`

COGAIN entity
```
system.state(pointer)     // get system state
                               // pointer to data structure
                               // returns 1=success, 0=failure
```

Gaze data method
Camera tracking

COGAIN entity
```
camera.tracking(integer)       // enable/disable camera tracking
                               // parameter=integer (1,0)
                               // returns 1=success, 0=failure
```

Gaze data method
Threshold of camera tracking

COGAIN entity
```
camera.tracking.threshold(integer)  // camera threshold
                                          // parameter=integer (threshold)
                                          // returns 1=success, 0=failure
```

Gaze data method
Mouse emulation

COGAIN entity
```
mouse.emulation(integer, integer, integer)
                           // enable/disable mouse emulation
                           // parameter=(1=on,0=off),
                        // x ratio, y ratio
                           // returns 1=success, 0=failure
```

Gaze data method
Move application window

COGAIN entity
```
window.move(string, integer, integer)
                     // parameter=window application title (string),
                    // new location top left corner
                    // (screen x, y)
                      // returns 1=success, 0=failure
```

Gaze data method

```
Close application window
```

COGAIN entity

```
window.close(string)
                    // parameter=window application title (string)
                     // returns 1=success, 0=failure
```

Figure 30. The basic operation API interface

3.4.5 Summary of API standards

In the same method as adopted for data archiving and data streaming this section has proposed a flexible approach to interfacing with a standardised COGAIN API that can be utilised by any software, together with unlimited possibilities for extension. The section also suggested the promotion of increasing levels of API functionality, from a basic form to a highly rich form, with the aim of encouraging the availability of the maximum amount of API functionality in a standard open-format interface that may be accessed and used by any interested party.

As with the data archiving and data streaming formats, these software standards could be easily implemented into the proposed COGAIN wrapper, which would then transpose manufacturer's own API interface functions in real time and present them in the proposed COGAIN format. Finalising the API standard and implementing an example of the wrapper API will be the subject of the next deliverable D2.3.

4 Discussion

This deliverable gives a brief overview of draft standards, levels and interface methods for data archiving, data streaming, and application programming interfaces for the COGAIN standard wrapper and virtual device standards.

This deliverable takes the survey of de-facto standards (COGAIN Deliverable 2.1) and outlines standards and levels for all of the commonly found existing systems and interfaces found in the survey of this first deliverable. The aim of the deliverable is to provide a path between the survey of existing systems and the implementation of a standard system by providing a basis for design and basic implementation of the COGAIN device interface.

4.1 Standards Research Retreat

Having defined the basic outline for interfaces in the areas of data archiving and data streaming and API where COGAIN may have most impact in forming common standards for gaze-based interaction 'plug 'n' play', the next step in Work Package 2 is to hold a research retreat to discuss in detail, modify and formalise these draft standards and start an implementation of the COGAIN virtual device wrapper. This will be done by meetings to bring together the members of COGAIN to discuss and formulate commonly agreed standards and implementations for a COGAIN virtual gaze tracker interface. This is the subject of the next deliverable in Work Package 2 'D2.3 Implementation of COGAIN Gaze Tracking Standards'.

4.2 The Importance of COGAIN

The deliverable found that a commonality can be found between the basic and required functions found in the software aspects of the systems. By offering a commonly held standard interface to a range of systems, COGAIN offers the chance of compatibility in firstly data archiving where a common file format may easily be achieved so that files may be read by any compliant application, and secondly data streaming where data streams may also be read by any compliant application, and finally in the API where a common interface may be used for differing gaze tracking systems. This is the importance of COGAIN. COGAIN still offers the best opportunity to date to produce common 'COGAIN virtual device standards' in addition to manufacturer's proprietary standards, allowing third party manufacturers and developers easier access and development of gaze-based systems.

References

Bates, R., Istance, H., Oosthuizen, L. and Majaranta, P. (2005) *D2.1 Survey of De-Facto Standards in Eye Tracking*. Communication by Gaze Interaction (COGAIN), IST-2003-511598: Deliverable 2.1. Available at http://www.cogain.org/results/reports/COGAIN-D2.1.pdf

Donegan, M., Oosthuizen, L., Bates, R., Daunys, G., Hansen, J.P., Joos, M., Majaranta, P., and Signorile, I. (2005) *D3.1 User requirements report with observations of difficulties users are experiencing.* Communication by Gaze Interaction (COGAIN), IST-2003-511598: Deliverable 3.1. Available at http://www.cogain.org/results/reports/COGAIN-D3.1.pdf

Jacob, R.J.K & Karn, K.S. (2003). Eye tracking in human-computer interaction and usability research: Ready to deliver the promises (Section commentary) In Hyönä, J., Radach, R. & Deubel, H. (eds.) *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, 573-605. Amsterdam, Elsevier Science, 2003.