CytometryML, Binary Data Standards

Robert C. Leif*

XML_Med, a Division of Newport Instruments, 5648 Toyon Road, San Diego, CA 92115

ABSTRACT

CytometryML is a proposed new Analytical Cytology (Cytomics) data standard, which is based on a common set of XML schemas for encoding flow cytometry and digital microscopy text based data types (metadata). CytometryML schemas reference both DICOM (Digital Imaging and Communications in Medicine) codes and FCS keywords. Flow Cytometry Standard (FCS) list-mode has been mapped to the DICOM Waveform Information Object. The separation of the large binary data objects (list mode and image data) from the XML description of the metadata permits the metadata to be directly displayed, analyzed, and reported with standard commercial software packages; the direct use of XML languages; and direct interfacing with clinical information systems. The separation of the binary data into its own files simplifies parsing because all extraneous header data has been eliminated. The storage of images as two-dimensional arrays without any extraneous data, such as in the Adobe® Photoshop® RAW format, facilitates the development by scientists of their own analysis and visualization software. Adobe Photoshop provided the display infrastructure and the translation facility to interconvert between the image data from commercial formats and RAW format. Similarly, the storage and parsing of list mode binary data type with a group of parameters that are specified at compilation time is straight forward. However when the user is permitted at run-time to select a subset of the parameters and/or specify results of mathematical manipulations, the development of special software was required. The use of CytometryML will permit investigators to be able to create their own interoperable data analysis software and to employ commercially available software to disseminate their data.

Keywords: CytometryML, Cytometry, Cytomics, XML, DICOM, FCS, Schema, Ada, SPARK, Standard

1. INTRODUCTION

The capacity of CytometryML to describe the metadata associated with analytical cytology (cytomics) data acquisition has been described[1,2]. In terms of metadata CytometryML has the following advantages over Flow Cytometry Standard[3,4], FCS: 1. CytometryML is a superset of FCS; 2. CytometryML can be extended by a standard means, the creation of XML schema; 3. CytometryML was derived from existing standards; 4. CytometryML was based on both a requirements document and a hazards analysis. 5. CytometryML is an application that is based on an ubiquitous international standard, XML. The use of XML permits interfacing with the present computer and Internet infrastructure. 6. The use of XML is consistent with the standardization efforts of many governments including the USA[5,6], Europe[7,8], etc. This use includes the selection of XML & XML schema for data interoperability by the U.S. Centers for Medicare & Medicaid Services (CMS)[9].

A major limitation of CytometryML was the lack of any capacity to deal with the binary list-mode data and image data produced by analytical cytology instrumentation. The binary formats associated with CytometryML and their manipulation are the subject of this paper. It should be noted that the separation of the binary data and metadata produced by analytical cytology instrumentation is not a new idea. It has been proposed[10] for an Image Cytometry Standard and was employed for a flow cytometer, the Coulter® Volume, Conductivity, Scatter (VCS) hematology systems produced by Coulter Corp, now Beckman Coulter®.

The separation of the binary data from the metadata eliminates the step of extracting the binary data from a combined file, such as those produced by FCS. Since the binary data is directly available, it can be analyzed with both user programs and be imported into commercially available programs, such as MathCad® (www.mathcad.com). The image data, as will be described below, has been imported and exported into the ubiquitous, Adobe® Photoshop® (www.Adobe.com)

2. METHODS

*rleif@rleif.com; phone 1 619 582-0437; fax 1 619 582-0437; www.newportinstruments.com

2.1. Software Development

Ada[11] was used as the programming language because of: its strong typing including the capacity to duplicate XML schema data-types, readability, standardization, safety, portability, real-time capabilities, stability, availability of a free compiler, and capacity for object oriented design. The Ada constructs were limited to essentially the same subset as SPARK[12] with the addition of generics (templates) and a function to create a tagged type (class). Since neither pointers nor run-time dispatching were employed, the executable occupies space in the stack and does not use the heap. The code developed for this project is compatible with real-time processing including Ada tasking. The environment for execution does not require a garbage-collector, since there were no pointers; hence, no garbage. The GNU Ada (GNAT) compiler was employed because: it came with a reasonable development environment, was free, is a complete implementation of Ada 95, and produces superb error messages. Although the free version of the compiler has not had its conformity assessed formally, the commercial version has been demonstrated to conform to the Ada standard. The software consists of 13 package specifications, 8 package bodies, and one main procedure. This software is supported by a collection of Ada utilities: strings, numeric types, files, time, etc.

2.1.1. Data Structures List Mode Files:

The naming convention for this project is the same as that previous employed for CytometryML[1,2]. As much as possible, a one-to-one correspondence was maintained between the XML schema data-types and the Ada data-types. In order to create a demonstration of the software for binary data that is produced by a flow cytometer, a model of the data acquisition system of a flow cytometer had to be created. The model is an updated version of a previously described[13] multiparameter electro-optical prototype. The information about the parameters required to name and control the Ada software is a subset of the CytometyML Parameters schema. Two of the schema elements that were included in the Ada software were the Short_Name and Parameter_Num (parameter number). These provide means to identify the parameter. The Short_Name is a string that contains from 1 to 16 characters, which is initially set to all Nulls. The resolution (Num_Bits_Stored) of the ADCs was set to values that are in the range of current instruments.

As shown in Code Segment 1, each code statement for Ada has the prefix "Ada" and is sequentially numbered; similarly, XML statements have the prefix "XML". All Ada type declarations and methods end in a semicolon. The suffix _Type is added to the name of the object. All unsigned integers have the prefix Uint, which is directly followed by its size in bits (Ada.1). As shown in Code Segment 1, it is a relatively simple matter (Ada.2) to create a record (struct) that includes the data from all of an instrument's parameters. The creation of an array of these records (Ada.3) is straight forward; and most computer languages or their libraries include indexed files and their methods (Ada.4, Ada.5). Objects in Ada methods have direction. They can be **in** (read), **out** (write), and **in out** (read and write). The records from the array and the elements of an indexed file can have a one-to-one correspondence; and the indices of the array and sequential file can be identical. Thus it is straight-forward to read and write individual records directly to either the array or the file. Usually for real-time measurements, the records are written to the array and subsequently to a file.

Code Segment 1

```
Ada.1. Num_Samples : Uint32_Type;

Ada.2. type All_Measurements_Simple_Rec_Type is record
         Time_Stamp_Part                : Time_Stamp_Type                :=
           Time_Stamp_Type (Real_Time.Clock);
         Low_Angle_Scatter_Part         : Low_Angle_Scatter_Type         := 1;
         Forty_Five_Degree_Scatter_Part : Forty_Five_Degree_Scatter_Type := 2;
         Log_Ninty_Degree_Scatter_Part  : Log_Ninty_Degree_Scatter_Type  := 0.0;
         DC_Impediance_Part             : DC_Impediance_Type             := 3;
         RF_Impediance_Part             : RF_Impediance_Type             := 4;
         Fl_1_Part                      : Fl_1_Type                      := 5;
         Fl_2_Part                      : Fl_2_Type                      := 6;
         Calculated_Opacity_Part        : Calculated_Opacity_Type        := 0.5;
       end record;

Ada.3. type List_Mode_Simple_Array_Type is array (1 .. Num_Samples) of
          All_Measurements_Simple_Rec_Type;
```



```
Ada.4. procedure Create (
          List_Mode_File : in out List_Mode_File_Type;
          Mode           : in     File_Mode           := Inout_File;
          Name           : in     String              := File_Name;
          Form           : in     String              := ""          );
Ada.5. procedure Read (
          List_Mode_File                : in     List_Mode_File_Type;
          All_Measurements_Simple_Rec : out All_Measurements_Simple_Rec_Type;
          From                          : in  Positive_Count);
```

Since flow cytometry experiments can include large numbers of cells or particles, the cost of storage can become significant. The obvious solution is to store only the relevant parameters. Unfortunately, the creation of software similar to that above which permits the storage of only a selected subset of the parameters (parts of the record) is not simple. One obvious solution was the use of a variant record where the excluded parameters were represented by nulls. After this solution was partially implemented, it was discovered that the use of a null or similar entities including one with its size specified to be zero always resulted in at least one byte being stored for each of the unused parameters. Besides the wastage of both memory and disc space, the inclusion of extraneous bytes in the stored data and its associated array increases the difficulty of parsing data from different versions of the code or compiled for different CPU architectures. The necessity for extraneous bytes was eliminated by serialization. The data from the selected parameters was transformed sequentially into an array of storage elements (bytes), which is similar to a stream data structure. This had the secondary advantage of permitting the record that contains the outputs from the detectors and calculated values to be an extensible type.

A major goal of the software design was to make the data-types and methods employed in the main procedure to be very similar to those shown in Code Segment 1.

The All_Measurements_Simple_Rec_Type shown in Ada.2 was replaced (Ada.6) by an array of storage_elements (bytes).

```
Ada.6 type El_Array_Type is array (1 .. Num_Stor_Els) of Storage_Element_Type;
```

Current microprocessors employ a byte as their storage elemente. The El_Array is sequentially filled only with data from previously selected parameters. The information concerning each parameter is stored as an extensible record (class):

Code Segment 2

```
Ada.7. type Generic_Parameter_Rec_Type is tagged record
          Short_Name_Part               : Short_Name_Type := Null_Bd_16;
          Parameter_Num_Part            : Parameter_Num_Type;
          Num_Bits_Stored_Part          : Num_Bits_Stored_Type;
          Stored_Part                   : Boolean                 := True;
          Index_Part                    : Index_Type              := 1;
          Parameter_Part                : Parameter_Generic_Type;
          Num_Stored_Els_Allocated_Part : Positive              :=
            Num_Bits_Allocated/ Storage_Unit_Size;
       end record;
```

The record, Code Segment 2, is described as being tagged (Ada.7), which means that it is a class and more parts can be added. The suffix _Part is used to indicate an element of a record. The terms Short_Name, Num_Bits_Stored, and Num_Bits_Allocated are identical with those with the Parameters.xml document. The Parameter_Num is a synonym for Waveform_Channel_Number. The value of the Stored_Part controls whether the data from a parameter will be included in the El_Array. The Index_Part contains the starting position for the parameter data in the El_Array. The Parameter_Part contains the data from the measurement and the Num_Store_Els_Allocated_Part is calculated by dividing the Num_Bits_Allocated by the number of bits (8) in a storage_unit. As shown in Table 1, If a parameter is stored, then the index of the next parameter is the sum of the Index and the Num_Store_Els_Allocated (Num. Bytes) of this parameter; otherwise the Index remains unchanged. Table 1 shows an example where 5 of the 9 parameters are stored. The total number of stored elements (bytes) for this example is 20. The 4 bytes for Calculated_Opacity start at 17 (the Index) and end at 20.

The Parameter_Part is generic and is instantiated, specified as being the type of each of the parameters shown in Table 1. This results in the Generic_Parameter_Rec_Type being a template for nine record types that only differ in the type of the Parameter_Part and the Number of Stored elements allocated to hold the sequence of bytes that corresponds to the value of each of the data-types.

The nine record_types, which are required to describe each of the parameters, are nine parts of the All_Measurements_Rec_Type. The All_Measurements_Rec_Type also includes a Num_Stor_Els_Part and an Initialized_Part. When an All_Measurements_Rec is created, the indices for the start of each parameter in the El_Array and the size (Num_Stor_Els) of the El_Array to hold the data from the selected parameters are calculated; then, the Initialized_Part is set equal to true.

Table 1: El_Array_Type Layout

Parameter	Stored	Index	Num. Bytes
Time_Stamp	True	1	8
Low_Angle_Scatter	False	9	2
Forty_Five_Degree_Scatter	True	9	2
Log_Ninty_Degree_Scatter	False	11	4
Dc_Impediance	True	11	2
Rf_Impediance	False	13	2
Fl_1	True	13	4
Fl_2	False	17	4
Calculated_Opacity	True	17	4

Following the design principles of "information hiding"[14], the methods, for example (Ada.8), employed for using this application that include the List_Mode_File and the List_Mode_Array include the All_Measurements_Rec.

```
Ada.8. procedure Read (
          List_Mode_File       : in  List_Mode_File_Type;
          All_Measurements_Rec : out All_Measurements_Rec_Type;
          From                 : in  Positive_Count);
```

The List_Mode_File is derived from the indexed sequential file of the Ada library package Ada.Direct_IO[11]. XML descriptions of new metadata required by the Ada program have been added to the multiplex_groups.xsd schema. This data includes for each of the instrument parameters: its number (Parameter_Num), whether it is stored, its Short_Name, and its Index. The multiplex_groups schema also includes an optional Uniform Resource Identifier, URI, that provides the location of the binary data file. The use of a URI is compatible with telemedicine since the data can be retrieved from any location on the Internet.

2.1.2. Date Structures Image Files:

Microscope image data in standard formats, such as TIFF[15], JPEG[16], or RAW can be imported, processed, and stored with commercially available software products. Analysis of the data with custom software is simplified by providing the image data separate from the XML metadata. In fact since the Adobe® Photoshop® RAW format when a header is not used is a simple two dimensional array. Software development for it required no specialized knowledge of the data format except for the type of pixels employed, the image's height and width in pixels and the number and order of storage of the color planes. This ability to directly parse the data greatly facilitated the development of fluorescence to absorbance image transformation software. Recently, Adobe has introduced a Digital Negative (DNG) Specification[17]. Unfortunately, this specification of metadata does not describe the binary image data; and since it is directed to commercial photography is much less relevant than DICOM to Cytometry metadata.

The gray scale pixels are unsigned integers (Ada.9, Ada.10, Ada.11). `Uint32` (Ada.11) has been included for completeness. The `Pixel_RGB_Type` (Ada.12)is used to create color images.

```
Ada.9. subtype Pixel_8_Type is Uint8_Type;
Ada.10. subtype Pixel_16_Type is Uint16_Type;
Ada.11. subtype Pixel_32_Type is Uint32_type;
Ada.12. type Pixel_RGB_Type is record
          R_Part : Pixel_8_Type := 0;
          G_Part : Pixel_8_Type := 0;
          B_Part : Pixel_8_Type := 0;
      end record;
```

A generalized two-dimensional image type is described in Ada.13. Specific array types are created by replacing the two maximum dimensions by constants and replacing the `Pixel_Type` with one of the four types described in Ada.9 through Ada.12.

```
Ada.13. type Raw_Array_Type is array (1 .. Height_Max, 1 .. Width_Max) of
          Pixel_Type;
```

Raw_Arrays can be stored as a file (Ada.14) and created from a file (Ada.15).

```
Ada.14. procedure Put_Raw_Array (
          File_Name : in     String;
          Raw_Array : in     Raw_Array_Type);
Ada.15. function Create_Raw_Array (
          Pixel_File_Name : String)
        return Raw_Array_Type;
```

The image manipulation methods take arrays that contain monochrome and RGB pixels.

The data-types in the schema (arrays.xsd) that declares one, two, and three dimensional arrays has been updated to include the XML versions of these Ada data-types. The type declaration in XML.1 is semantically identical with that of Ada.9. The prefix nums: is an abbreviation of the name of the schema where `UInt8_Type` was declared. The XML descriptions of the metadata for two and three dimensional binary files consist of sequences, which are the equivalent of records or structs. Similarly to the list-mode data, each of these two sequences includes an optional element that provides an optional Uniform Resource Identifier, URI, that provides the location of the binary data file. The use of a URI is compatible with telemedicine since the data can be retrieved from any location on the Internet.

```
XML.1.<simpleType name="Pixel_8_Type">
        <restriction base="nums:UInt8_Type"/>
      </simpleType>
```

2.2. S Phase Cell Sample Preparation and Imaging

As previously described[18], 5-BrdU labeled tissue culture cells were labeled with a Europium(III) macrocycle (Quantum Dye®) conjugate of anti-5-BrdU. Subsequently, the cellular DNA was stained with 4',6-Diamidino-2-phenylindole (DAPI). The protocol[19] of the Phoenix Flow Systems (San Diego, CA) ABSO-LUTE-S™ kit was followed with the substitution of the EuMac-Anti-5-BrdU for the fluorescein labeled antibody. This direct staining procedure was based on the SBIP™ (Strand Break Induced Photolysis) technique[20]. Centrifugal cytology preparations[21] were made and the cells were allowed to air-dry from the ethanol, because the low surface tension of ethanol produces minimal morphological distortion. The dry cells were covered with Clearium Mounting Medium and the solvent was removed from the Clearium by mild heat generated with a heat gun.

As previously described[18], the cells were illuminated with a 100 watt Hg-Xe Arc and observed with a Ploem-Pak cube UV DAPI, equipped with a 365 nm narrow-band-width excitation filter (Omega 365HT25) and a 400 nm Beamsplitter (Omega 400DCLP02). The CCD optical path was optionally equipped with either a 619 nm narrow-band emission filter

(Chroma Technology D620/20m) or a standard DAPI 450 nm emission filter (Omega 450DF65). The narrow band 620 nm filter was used to image the narrow red emission from the Quantum Dye® and the standard DAPI filter was employed for DAPI emission. The two emissions were separately imaged to 12 bit gray scale monochrome images, which were stored as TIFF files. These 16 bit gray scale files were opened with Adobe Photoshop, converted to 8 bit gray scale images, and stored as RAW files. These RAW files lack a header and thus can be read in as a two-dimensional array of 8 bit pixels. Three color, RGB, files with each pixel consisting of a red, a green, and a blue byte can also be opened and saved.

3. RESULTS

3.1 List Mode Data

The capacity of the Ada software to produce list-mode files and the effect of changing the composition of the stored parameters was tested. Two sets of data were produced. For the first set of measurements, all 9 of the parameters were stored, which resulted in the El_Array consisting of 32 storage elements (bytes). For the second set of measurements, only 1 parameter, Low_Ang_Scat, was stored, which resulted in the El_Array consisting of only 2 bytes (Table 1). Figure 1 shows plots of the calculated number of bytes vs. the size of the file reported by Windows XP Professional. The calculated value is the product of the size of the array of storage elements (`Num_Stor_Els`) and the number of cells or particles stored (`Num_Samples`). The file sizes were obtained with a simple BAT file that transferred the names and sizes of all of the list-mode files to a text file, which was subsequently imported into Microsoft® Excel, analyzed, and graphed. The equations for the lines with 2 bytes and 32 bytes are respectively y = x+ 561.66 and y = 0.9999x + 646.15. The values for R^2 were both 1. Thus the slopes for both were very close to 1 and the intercept, which is the overhead associated with storing a file is about 600 bytes.

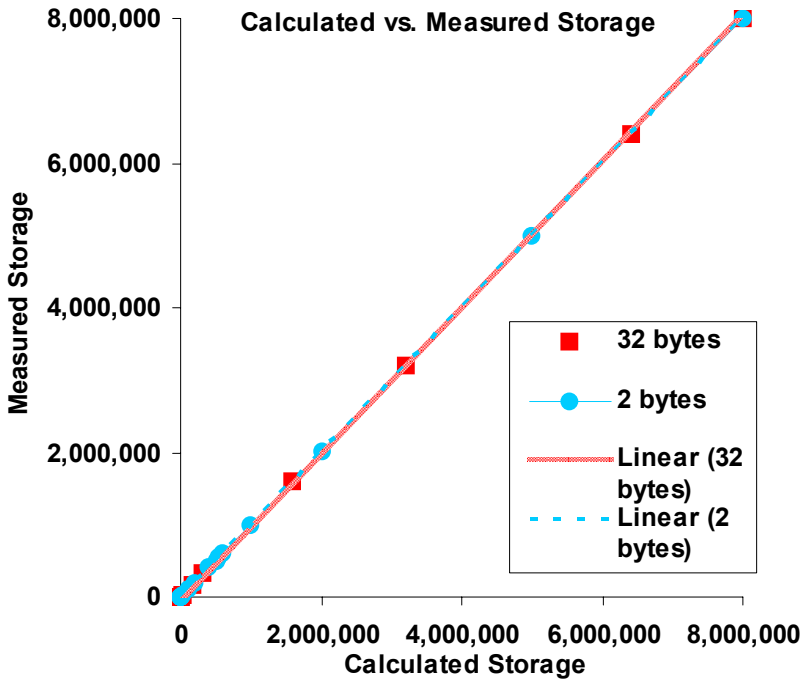

Figure 1 consists of two graphs: The data points for the 2 byte and 32 byte storage element arrays are shown respectively as circles with a dashed trend-line and squares with a cross-hatched trend-line.

3.2. Image Data

A Fluorescence_To_Absorbance program has been written. This program aligns the gray scale images and then produces a psuedocolor absorbance image. In the case of the combined Europium Quantum Dye and DAPI images, the red and blue RAW image files were each converted into a two dimensional array of unsigned 8 bit integers, Uint8s. The red and blue arrays were then aligned by sliding the red array against the blue array and selecting the relative position that maximized the sum of the product of the red and blue pixels (cross-correlation). An RGB array with the same height and width as the red and blue array was created. Then each RGB pixel in the RGB array was set equal to a calculated RGB pixel. The value

of each of the three parts of the RGB pixel was calculated by subtracting from a white (255) Third_Pixel a Decrement, which was calculated as shown in Equation 1.

$$\text{Decrement} = \text{Weighing\_Factor}(\text{First\_Pixel} + \text{Second\_Pixel}) \text{ and } 0 <= \text{Decrement} <= 255 \quad (1)$$

Preliminary studies indicated that a weighting factor of 0.7 produced visually pleasing results. The value of the green pixel used for the subtraction was 0. For instance, from an original green pixel (255), the sum of 70% of the corresponding pixel in the blue and red arrays was subtracted. After the red, green, and blue decrements were completed, the resulting RGB array was saved as a RAW RGB file and opened in Photoshop.

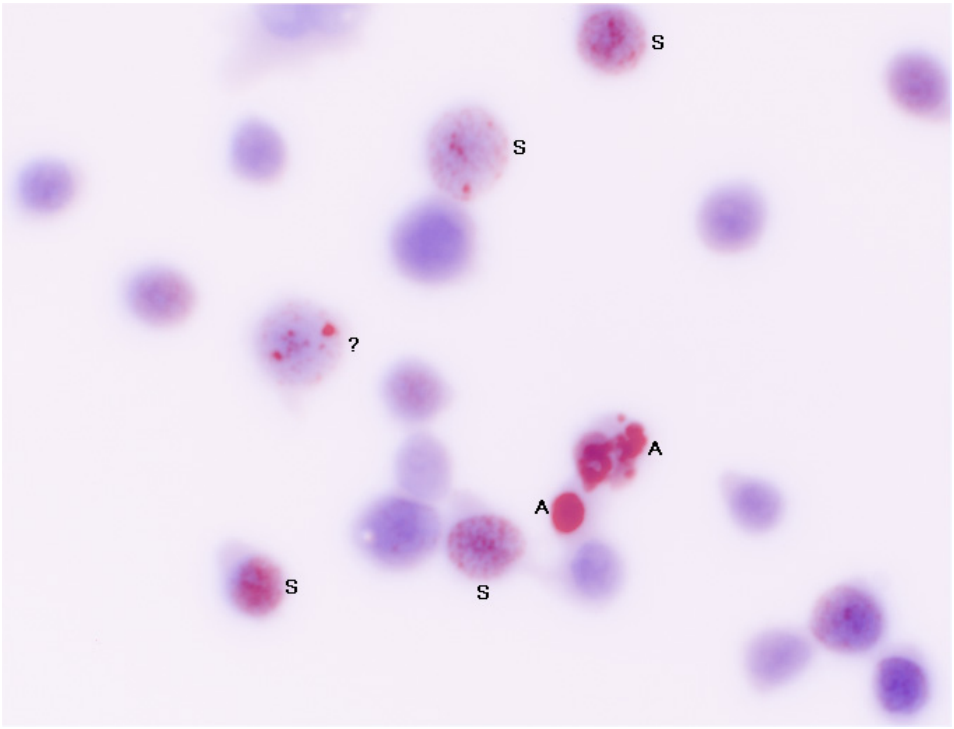

Figure 2. Software generated pseudo-color image of anti-5-BrdU and DAPI labeled growing tissue culture cells. This color image was created by subtracting the red (europium) image from the blue and green image planes and the blue DAPI image from the green and red image planes. Both the blue and red images were obtained with a 60 x oil immersion lens. The continuous excitation was at 365 nm. Both images were binned to 680 x 518 pixels. The cells (S) have a punctate staining pattern, which shows small islands of DNA synthesis. The cells (A) have large densely stained areas, which are the result of apoptosis induced strand breakage. The cell (?) has both types of staining.

4. PORTABILITY AND CONFORMATY

An obvious requirement for a program that reads and writes binary data is that files written with one program are able to be read by another program. Two major questions concerning this interoperability are: 1) What actions are necessary to achieve this requirement? 2) What are the tests required in a conformity assessment to provide reasonable assurance that this requirement has been met by a specific program?

The following actions have been taken to facilitate interoperability: 1) The formats for the binary data from both the list-mode and image files have been limited to only include the data. In the case of binary image data in a standard format, such as JPEG2000[16] or TIFF, the capacity of the reused software component or new program to accurately produce RAW images and the capacity of the program to convert RAW images would have to be tested. This study problem was circumvented by using Photoshop for these conversions. 2) The metadata for the binary formats is available in XML format. 3) Simple reference files have been and can be created that could be used for testing, and, if need be, decoding. In the case of list-mode data, the array and subsequently the file can have identical elements with the data from all of the parameters in an element being unique. An example of this is shown in the declaration of the All_Measurements_Simple_Rec_Type

(Code Segment 1, Ada.2). Since this constant element for a reference file can also be provided in XML, the values from each of the parameters can be converted to their bit pattern and their starting position (index) in the element (El_Array) retrieved from the array can be determined. Validation consists of checking that these constant elements match the corresponding instance of the XML ComplexType.

Reference image files can be used to determine interoperability between programs that read and store images. Simple reference images are alternating columns (vertical stripes) or rows (horizontal stripes). The number of pixels for each stripe should be constant. The new DICOM Grayscale Standard Display Function[22] provides solutions to the very difficult problem of obtaining reproducible image display from binary grayscale image data.

5. CONCLUSIONS

Binary data and file formats for flow-cytometry list-mode data and digital microscopy two dimensional image data have been developed and tested. List-mode binary data have been efficiently represented in memory and as files. For the sake of size efficiency, the data structure for memory is a single dimensional array of an array of bytes (storage elements). This array of bytes includes the data from each of the parameters that have been selected for storage. In terms of the application programming interface, methods are executed on a record that includes all of the parameter data and relevant control data. The array of bytes need not be directly accessed. This single dimensional array is stored as an indexed sequential file, where the data describing the parameters, which is located in records, can be stored and retrieved by their position in a file. Single dimensional arrays of positive integers (indices) can be stored as indexed sequential files. These arrays of indices can be used to describe the positions (indices) of cells from a subset. The use of these indices will eliminate the time required to read the subset field of the data structure that describes each cell. Instead, the data from only the cells that are members of a specific subset need to be retrieved.

The use of standard image formats, such as TIFF and JPEG 2000[16], for digital microscopy is the ultimate example of standards' paucity; nothing has been created. The special instance of the Adobe RAW format has the virtue of simplicity and ease of use. The reduction in software development effort by the use of the RAW format was demonstrated in the Fluorescence_To_Absorbance program. The development of this program involved less effort that the creation of a program to parse a TIFF file. Another interesting example of the possibilities of the use of simple binary formats is the future development of a means to interface the RAW image data and the new list-mode data formats to commercially available applied mathematics packages, such as Mathsoft™ MathCad. This would remove the need to employ domain specific software for the analysis of analytical cytology data. Since the list-mode binary software and the CytometryML schema are based on the DICOM Waveform[23], they can be modified and extended for use with other modalities that produce waveforms and include future relevant improvements from the development of DICOM. Thus in terms of the storage of both metadata and binary data, CytometryML should serve as an open format that is designed to interface with any XML based system. In short, CytometryMl has now made FCS obsolete.

ACKNOWLEDGEMENTS

The generous support of Newport Instruments is greatly appreciated. The comments and advice of Suzanne B. Leif, Stephanie H. Leif, and David M. Coder are greatly appreciated.

REFERENCES

1. R. C. Leif, S. B. Leif, and S. H. Leif, "CytometryML, An XML Format based on DICOM for Analytical Cytology Data", *Cytometry* **54A** pp. 56-65, 2003.

2. R.C. Leif, S.H. Leif, S.B. Leif, "CytometryML, a markup language for analytical cytology", in *Manipulation and Analysis of Biomolecules, Cells and Tissues*, D. V. Nicolau, J. Enderlein, and R. C. Leif, Editors, SPIE Proceedings **4962,** pp 288-297, 2003.

3. L. Seamer, B. Bagwell, L. Barden, M. Christofferson, L. E. Magruder, G. Malachowski, R. F. Murphy, D. Redelman, G. C. Salzman, and J. C. S. Wood. "Data File Standard for Flow Cytometry, Version FCS3.0," *Data File Standards Committee of the International Society for Analytical Cytology (ISAC)*, http://www.isac-net.org/ 1996.

4. L. C. Seamer, C. B. Bagwell, L. Barden, D. Redelman, G. C. Salzman, J. C. S. Wood, and R.F. Murphy, "Proposed New Data File Standard for Flow Cytometry, Version FCS 3.0.", *Cytometry* **28**, 118-122, 1997.

5. K. Sall, "How the US Federal Government is Using XML: One Year Later", http://www.silosmashers.com/pdfDocu-

ments/How-US-Govt-Using-XML-1YL.pdf (Last visited 29 Dec. 2004).

6. Anonymous, “XML.Gov “, http://xml.gov/ (Last visited 29 Dec. 2004).

7. Anonymous, “European Interoperability Framework, for Pan-european Egovernment Services”, Version 1.0, November 2004, http://europa.eu.int/ida/en/document/3473, 2004.

8. Anonymous, “UN/CEFACT XML Naming and Design Rules Technical Specification”, Draft 1.0, 3 August 2004, http://www.disa.org/cefact-groups/atg/downloads/index.cfm, 2004.

9. Chief Information Officer, Office of Information Services, Centers for Medicare & Medicaid Services (CMS), “CMS Target Architecture, *U.S. Department of Health and Human Services*, http://www.cms.hhs.gov/it/enterprisearchitecture/cms_target_architecture.pdf, 2004.

10. P. Dean, L. Mascio, D. Ow, D. Sudar, and J. Mullikin, “Proposed standard for image cytometry data files”, *Cytometry* **11** pp. 561-569, 1990.

11. R. Brukardt (Editor) “Consolidated Ada Reference Manual, consisting of the international standard (ISO/IEC 8652:1995): Information Technology -- Programming Languages -- Ada, as updated by changes from Technical Corrigendum 1 (ISO/IEC 8652:1995:TC1:2000)”, http://www.adaic.org/standards/ada95.html (Last visited 30 Dec. 2004)

12. A. Hall and R. Chapman, “Correctness by construction: developing a commercial secure system”, *Software, IEEE* **19**, pp. 18-25, 2002.

13. R. C. Leif, M. L. Cayer, W. Dailey, T. Stribling, and K. Gordon, “The use of a Spherical Multiparameter Transducer for Flow Cytometry”. *Cytometry* **20**, pp 185-190, 1995.

14. D. L. Parnas, P. C. Clements, and D. M. Weiss, “Software reusability: **Vol. 1**, concepts and models table of contents, pp. 141 - 157 ISBN:0-201-08017-6 Publisher ACM Press, 1989.

15. http://partners.adobe.com/public/developer/tiff/index.html

16.. Anonymous, “JPEG 2000, Our New Standard!”, http://www.jpeg.org/jpeg2000/index.html?langsel=en (Last visited 30 Dec. 2004).

17. Adobe Digital Negative (DNG) Specification, Version 1.0.0.0, September 2004 http://www.adobe.com/products/dng/pdfs/dng_spec.pdf

18. R. C. Leif, M. C. Becker, A. Bromm Jr., N. Chen, A. E. Cowan, L. M. Vallarino, S. Yang, and R. M. Zucker, “Lanthanide Enhanced Luminescence (LEL) with one and two photon excitation of Quantum Dyes® Lanthanide(III)-Macrocycles”, in *Manipulation and Analysis of Biomolecules, Cells, and Tissues*, D. V. Nicolau, J. Enderlein, R. C. Leif, and D. Farkas, Editors, SPIE Proceedings **5322** pp. 187-199, 2004.

19 “ABSOLUTE-S^{TM} Protocol”, Phoenix Flow Systems, 6790 Top Gun St., Suite 1, San Diego, CA 92121-4121, Tel. (858) 453-5095; http://www.phnxflow.com.

20. X. Li, F. Traganos, M. R. Melamed, and Z. Darzynkiewicz, “Detection of 5-bromo-2-deoxyuridine incorporated into DNA by labeling strand breaks induced by photolysis (SBIP)”, *Int. J. Oncol*., **4**, pp. 1157-1161, 1994.

21. R. C. Leif, “Methods for Preparing Sorted Cells as Monolayer Specimens”, *In Living Color, Protocols in Flow Cytometry and Cell Sorting*, R. A. Diamond and S. DeMaggio, Editors, Springer, ISBN 3-540-65149-7, pp. 592-619, 2000.

22. “Part 14: Grayscale Standard Display Function”, *Digital Imaging and Communications in Medicine (DICOM) PS 3.14 -2004, Digital Imaging and Communications in Medicine (DICOM)*, National Electrical Manufacturers Association, VA, USA, 2004. http://medical.nema.org/dicom/2004/04_14PU.PDF, 2004.

23. “A.34 Waveform Information Object Definitions, p. 157 and C.10.8 Waveform Identification Module” pp. 699-708“, *Digital Imaging and Communications in Medicine (DICOM) PS 3.3-2004*, National Electrical Manufacturers Association, VA, USA, 2004. http://medical.nema.org/dicom/2004/04_03PU.PDF, 2004.

