REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
May 2004	Technical Paper/Briefing Slides	May 2004

4. TITLE AND SUBTITLE
Automated Extraction of Data Display Configuration from Telemetry Applications

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Charles H. Jones PhD.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)
412 TW/ENTI
Air Force Flight Test Center (AFFTC)
Edwards AFB, CA 93524

8. PERFORMING ORGANIZATION REPORT NUMBER
PA-04097

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
412 TW/ENTI
Air Force Flight Test Center (AFFTC)
Edwards AFB, CA 93524

N/A

12. DISTRIBUTION / AVAILABILITY STATEMENT
A Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
CC: 012100 CA: Air Force Flight Test Center Edwards AFB

14. ABSTRACT
This paper presents a methodology for a knowledge-based approach to effectively extract configuration data from raster images and source code of display systems. Our approach uses three different means of recognizing display objects from data display images and source code: extracting information from multiple raster images; vectorizing the images and using a rule-based object identification; and parsing the source code to extract graphics, dynamics, and data variables of the display system. No single technique would be able to completely extract display information. For example, it is impossible to completely extract display dynamics even if numerous raster images were provided. To solve this, we apply data fusion techniques in order to decrease the uncertainty inherent in each of the above-mentioned techniques.

15. SUBJECT TERMS
Configuration Files Neutral Format Machine Learning
Data Display Markup Language (DDML) Data Fusion

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Unclassified Unlimited	24	Charles H. Jones PhD.
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			19b. TELEPHONE NUMBER (include area code) 661-275-4492

Automated Extraction of Data Display Configuration from Telemetry Applications

Ronald Fernandes, Michael Graul, Chang-Nien Wu, Burak Meric
Knowledge Based Systems, Inc.
College Station, TX 77840

Charles H. Jones, PhD
412 TW/ENTI
Edwards AFB, CA 93524-8300

ABSTRACT

This paper presents a methodology for a knowledge-based approach to effectively extract configuration data from raster images and source code of display systems. Our approach uses three different means of recognizing display objects from data display images and source code: extracting information from multiple raster images; vectorizing the images and using a rule-based object identification; and parsing the source code to extract graphics, dynamics, and data variables of the display system. No single technique would be able to completely extract display information. For example, it is impossible to completely extract display dynamics even if numerous raster images were provided. To solve this, we apply data fusion techniques in order to decrease the uncertainty inherent in each of the above-mentioned techniques.

KEYWORDS

Data Displays, Configuration Files, Neutral Format, Machine Learning, Data Fusion

INTRODUCTION

The issue of the reuse of data display configurations is important in T&E environments in order to standardize T&E procedures and conduct flight-testing in an environment independent of testing location, data acquisition, and display system. To achieve this, an XML-based neutral format called Data Display Markup Language (DDML) has been developed to serve as the inter-lingua for data displays [1]. DDML eliminates the n^2 problem of having to develop a total of $n(n\text{-}1)$ translators for a set of n data display vendor formats. With DDML, only $2n$ translators have to be developed—each vendor format would need a pair of translators between itself and DDML. In addition, a change in one of the vendor formats would only require the re-coding of the translators between that format and DDML.

DDML has been developed to be generic enough to encompass various vendor-specific data display formats and, at the same time, be unified and not just a loose grouping of vendor formats under the cover of XML. DDML is defined in terms of four layers: graphics components, dynamics, variables, and data sources. And, it closely parallels a typical software–layered architecture composed of graphics resources, visualization and user interfaces, and information management and persistence, respectively. The DDML layers are shown in Figure 1.

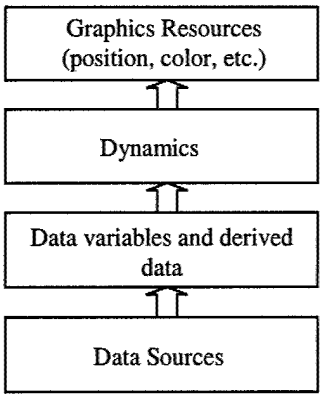


Figure 1: DDML Layers

Translators to and from DDML are relatively easy to develop when the vendor format is formally defined using the Backus-Naur Form (BNF), XML DTD/schema in the case of XML-based configurations, or at least an informal but accurate description of the format. The real problem arises when data display systems lack the ability for end-user configuration. In other words, there is no external configuration specification capability to extract display information to translate to DDML, and at the same time, the need to rapidly duplicate the functionality in another vendor display system is critical. Such cases arise when data display applications are custom-built for specific T&E scenarios without using off-the-shelf data display systems but with the use of standard graphics API definition should go here (APIs) such as OpenGL or Java 2-D/3-D library.

To solve this problem, we have developed a methodology to extract the data display configurations of such applications by using three different methods of recognizing display objects and their properties: use of raster images, use of vector images, and source code parsing. Each of these methods provides imprecise knowledge about the display configurations. Since no single technique can be used to completely extract display information, data fusion techniques have to be employed. Our approach makes the following assumptions. First, there is a well-defined software development standard that is in place and is being enforced. This is required in order to process a number of existing data displays using our methodology, in order to semi-automate the extraction of data display configurations in a cost-effective manner. Second, the source code is available. Finally, it assumes that for each application, numerous screen shots of the display can be saved as raster images while the application is running.

METHODOLOGY

The methodology uses both image recognition and source code parsing to extract display configuration information from the application. Image recognition includes raster processing and vector processing of the run-time displays of the data display application. In order to better extract the graphics and dynamics, multiple images are required.

Raster-based recognition involves pattern and shape recognition techniques in order to identify objects in a data display. An example of using shape recognition is artificial neural networks, where a 2-D region of interest (ROI) of the raster is fed to a classifier neural network, which is basically a multi-layer perceptron that has been previously trained to provide a probability for each of the known shape classes. If the display image is available as a compressed file format such as JPEG, it must first be converted into a bitmap.

The second method involves vectorizing the raster image and then extracting display graphics and partial dynamics from the vector images. A popular vector image is the W3C's Scalable Vector Graphics (SVG) [2], which is also the format used in DDML to represent graphic primitives such as lines and labels. A common method of performing vectorization is the "thinning" method [3], which decides whether or not a pixel belongs to the vector skeleton. Often, post processing is required to recognize the broken line segments, polygons, and oval shapes before we can actually recognize one particular display instrument. The vectorization tool can be extended to include a user to group the graphics primitives of the vector skeleton into high-level objects, which can then be converted to DDML. The IDEFØ activity model of the vector-based object recognition is shown in Figure 2.

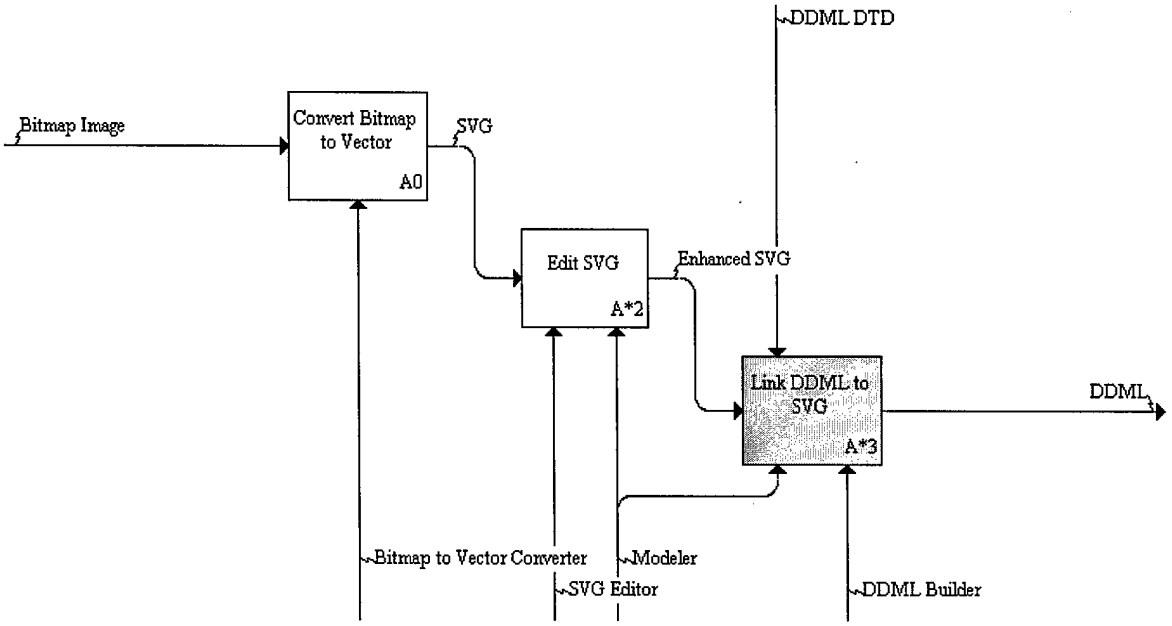


Figure 2: Vector-based Object Recognition

The third method involves the parsing of the source code of the display system to extract knowledge at all four layers of DDML. -As a pre-processing step, it is important to understand the basic structure of the program to select relevant files to parse. In addition, obtaining the programming flow from design documents and the software requirements package plays an important role in selecting appropriate files to extract relevant layers of DDML.

Our preliminary studies showed that the source code parsing approach in the case of non-object--oriented programming paradigm is challenging because of the following nontrivial tasks:

- Recognizing high-level objects when the program architecture is not object oriented. For example, there is not a slider object class or a data structure that gathers primitive objects that are related to a slider. A slider was simply drawn by a number of drawline() statements. Thus, it is not straightforward to identify the slider object from such statements.
- Detecting the data source objects of real-time simulation libraries such as SCRAMNET.
- Ensuring that the current display programs are all consistent. (e.g., programming language, environment, use of graphics libraries, object oriented vs. procedural).

As a result of the challenging nature of this approach, the majority of extractable items are ~~only~~ low-level drawing objects such as rectangles, circles, and polygons can be extracted from C source code files.~~,~~ However,~~although~~ some basic high-level entities such as labels and alphanumeric tables can also be extracted.

Higher level graphic objects can be inferred from these low-level entities using a rule-based engine containing rules relating to proximity on screen, proximity within the source code, and object-specific characteristics. In order to detect object dynamics, we mainly rely on the conditional statements within the C source code. To distinguish between true dynamics and those that are not, we check those variables that are within the conditional statement. If those variables are global variables that are related to some simulation libraries, we conclude that the variables play a part in the dynamics of the object within the conditional statement that is being parsed.

A comparison of the automated translatability of the three methods is shown in Table 1.

Table 1: Comparison of Translatability by the T~~t~~hree ~~m~~Methods

DDML Layer	Raster Graphics	Vector Graphics	Parsing Source Code
Graphics Resources	partial	partial	partial
Dynamics	partial, with numerous samples in color	partial, with numerous samples in color	yes
Data Variables	no	no	yes
Data Sources	no	no	yes

DATA FUSION

While it is obvious that the source code parsing approach is the ideal approach to ensure that all display information is correctly extracted, it is very tedious when the code is not object oriented, making it difficult to recognize high-level objects such as sliders and strip charts. On the other hand, the main drawback of the image processing approach is that it is almost impossible to extract the dynamics of the system. The only possibility would be to have multiple screenshots from the same display to determine the dynamics. But, even with a large number of sample screenshots, this would be a non-trivial problem. Also, the variables and data sources are impossible to extract using the image processing approaches. As a result, it is necessary to fuse the results of the three approaches to produce a coherent extracted display configuration in DDML format. Code parsing would be ideal to get the dynamics as well as some of the primitives of the display, such as labels, tables, and other static objects. Image processing would be more promising for identifying high-level objects and irregular shapes. Use of data fusion techniques such as Bayesian inference, Dempster-Shafer theory of evidence, and fuzzy logic inference techniques [4], [5], [6] are suitable for combining the inferences of each of the methods for the graphics and dynamics. The overall data fusion schema is shown in Figure 3.

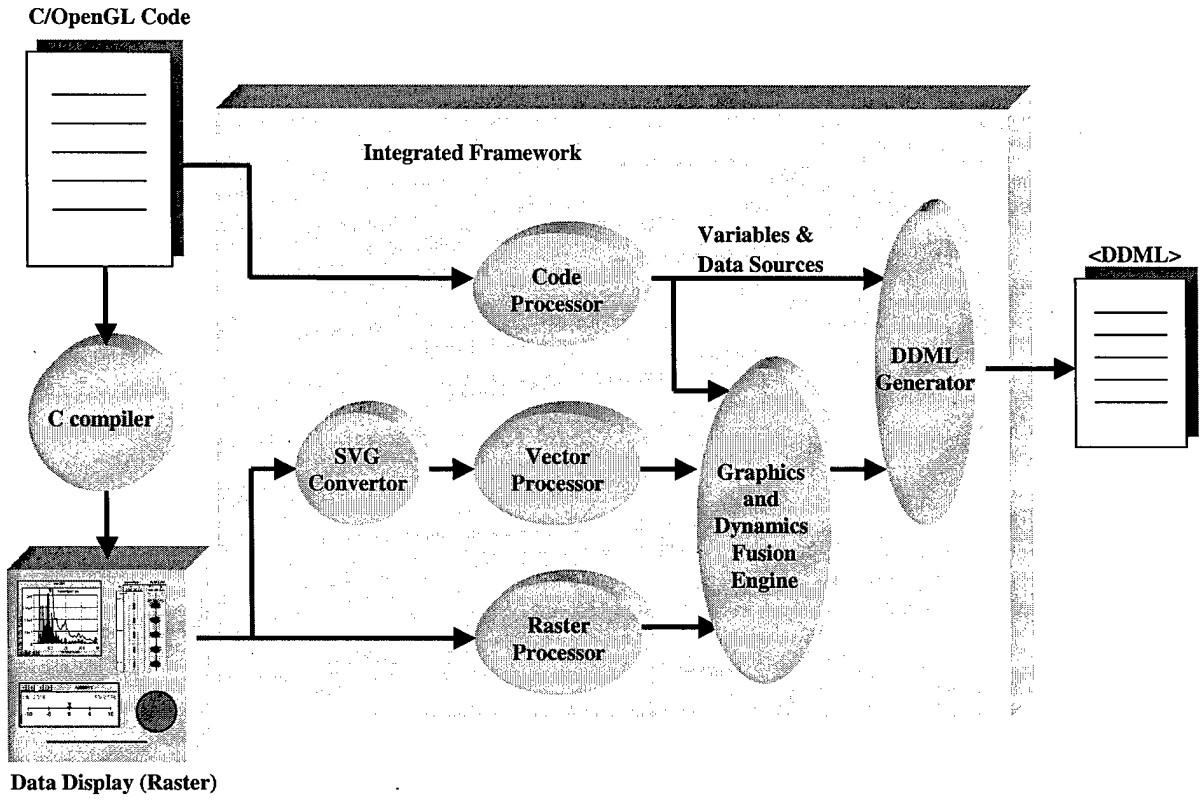


Figure 3: Data Fusion for Combining Data Configuration Extraction Techniques

SUMMARY

We present a methodology for extracting data display configuration from a data display system based on available raster images and source code. Our approach uses raster processing, vector processing, and source code processing techniques to extract configuration information at all four layers of DDML. The use of data fusion techniques helps provide a coherent DDML file, which can then be used in standard T&E procedures.

REFERENCES

[1] B. Meric, M. Graul, R. Fernandes, and C. Jones, "Design of an Interlingua for Data Displays," ITC Proceedings, 2003.

[2] Scalable Vector Graphics (SVG) 1.1 Specification, http://www.w3.org/TR/SVG11/.

[3] L. Lam, S.W. Lee, and C.-Y. Suen. "Thinning methodologies - a comprehensive survey." IEEE Trans. Pattern Analysis and Machine Intelligence, 14(9): 1992, pp. 869-885.

[4] Gros, X.E., NDT Data Fusion, Arnold Publishers, London, 1997.

[5] Hall and Llinas, Editors, Handbook of Multisensor Data Fusion, CRC Press, Boca Raton, FL, 2001.

[6] Unal, G., and Meric, B., "On the Uncertainty of Propositions in the Combined Body of Evidences," Bull. Tech. Univ. Istanbul, Vol. 49, 1996, pp. 365 – 374.

Automated Extraction of Data Display Configuration from Telemetry Applications

Ronald Fernandes, Mike Graul, Chang-Nien Wu, Burak Meric
Knowledge Based Systems, Inc.

Charles Jones, PhD
Edwards Air Force Base

8th Annual ITEA Test Instrumentation Workshop
May 5, 2004

Agenda

- Motivation
- Data Display Markup Language (DDML)
- Methodology for Automated Extraction
- Data Fusion
- Applications
- Conclusions

Why Automated Extraction?

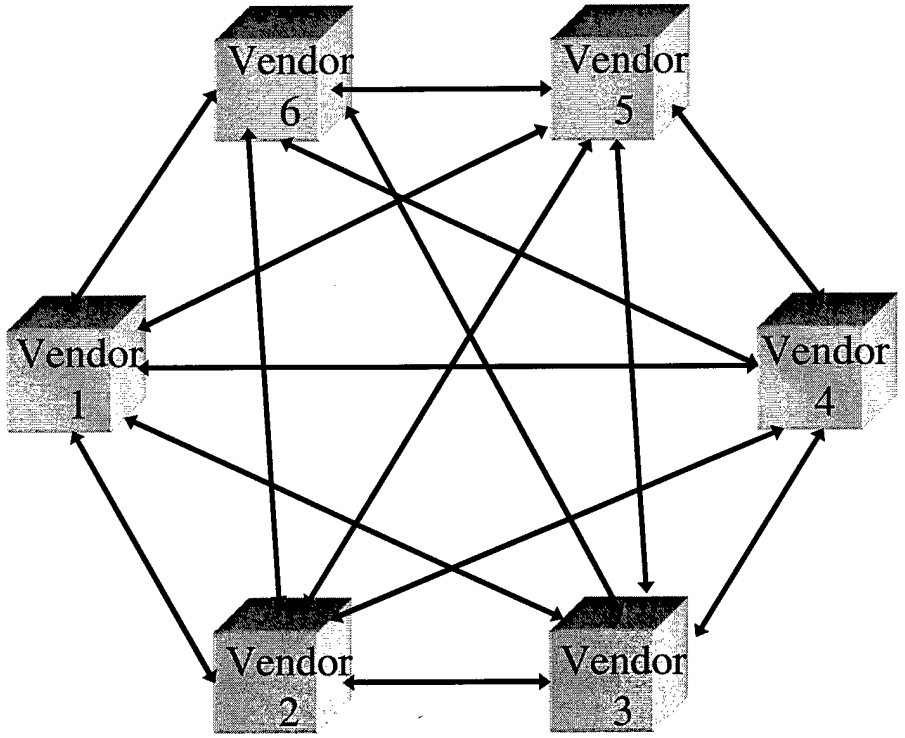
- Data display setup is a lengthy procedure
- Diverse data display systems
- Test and Evaluation (T&E) are performed at different locations
- Need reuse of displays for joint service T&E missions
  - Joint Strike Fighter (JSF)
  - Joint Air-to-Surface Standoff Missile (JASSM)

Why Automated Extraction?

- Many T&E applications are custom developed
- A portable display configuration requires
  - Hardware / OS platform independence
  - Generic Instrumentation Support Systems
  - Neutral language to represent display configuration
  - Tools to translate among different display configuration systems

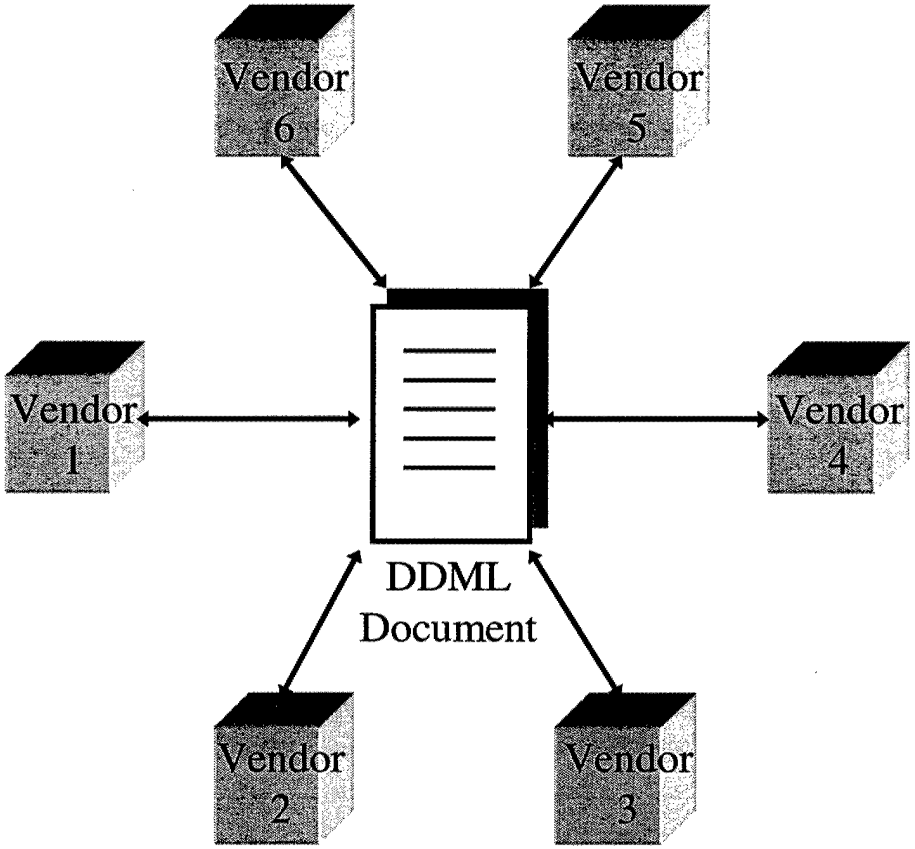
Why Neutral Language?

- Currently, displays are created manually
- Develop translators without neutral language



Why Neutral Language?

- Develop translators with neutral language

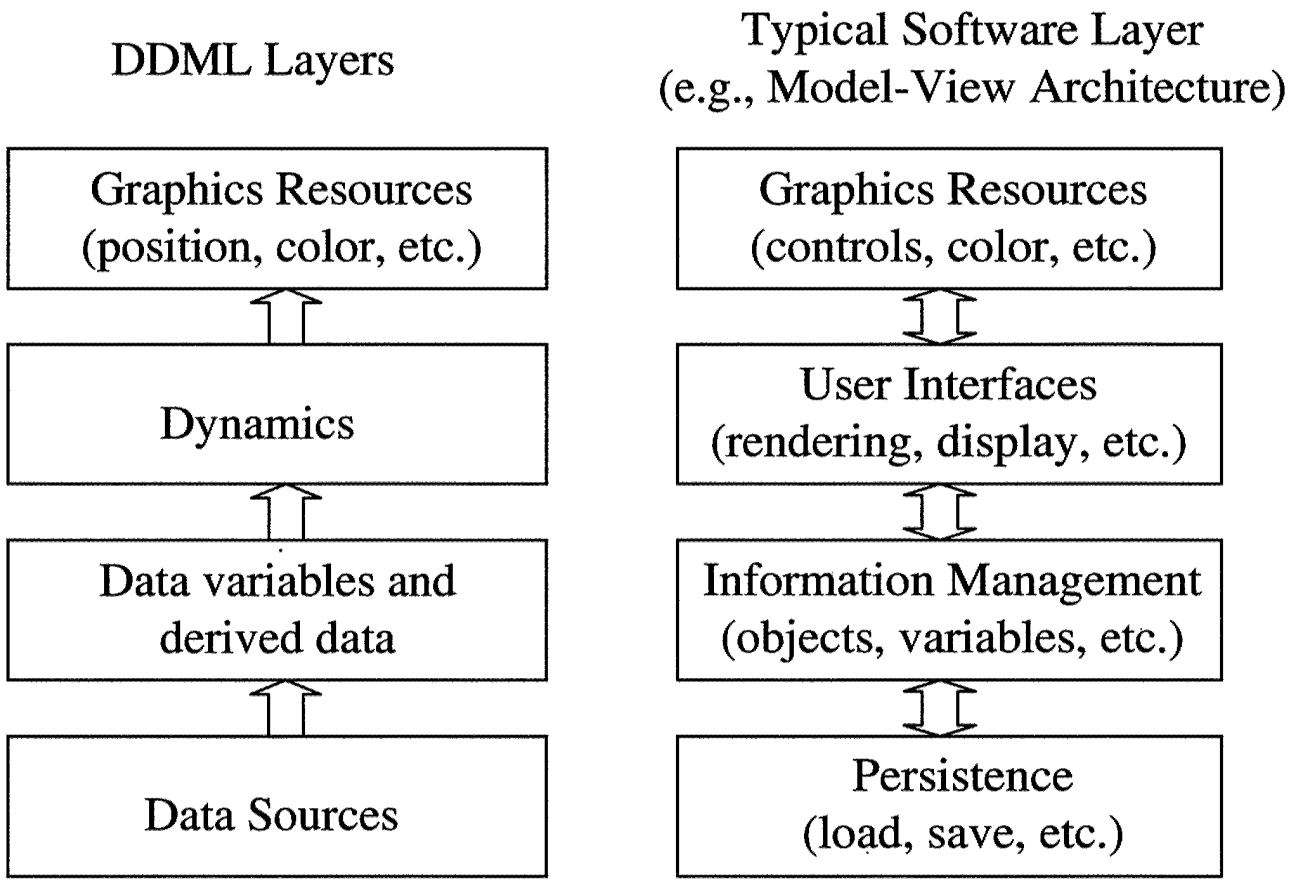


Data Display Markup Language

- eXtensible Markup Language (XML)-based
- Four logical layers: graphics resources, dynamics, variables and data sources
- Includes most T&E display objects
  - plotters, meters, strip charts, etc.
- Primitive graphics resources based on Scalable Vector Graphics (SVG)
  - rectangles, lines, etc.
- Generic to support nonstandard objects

Data Display Markup Language

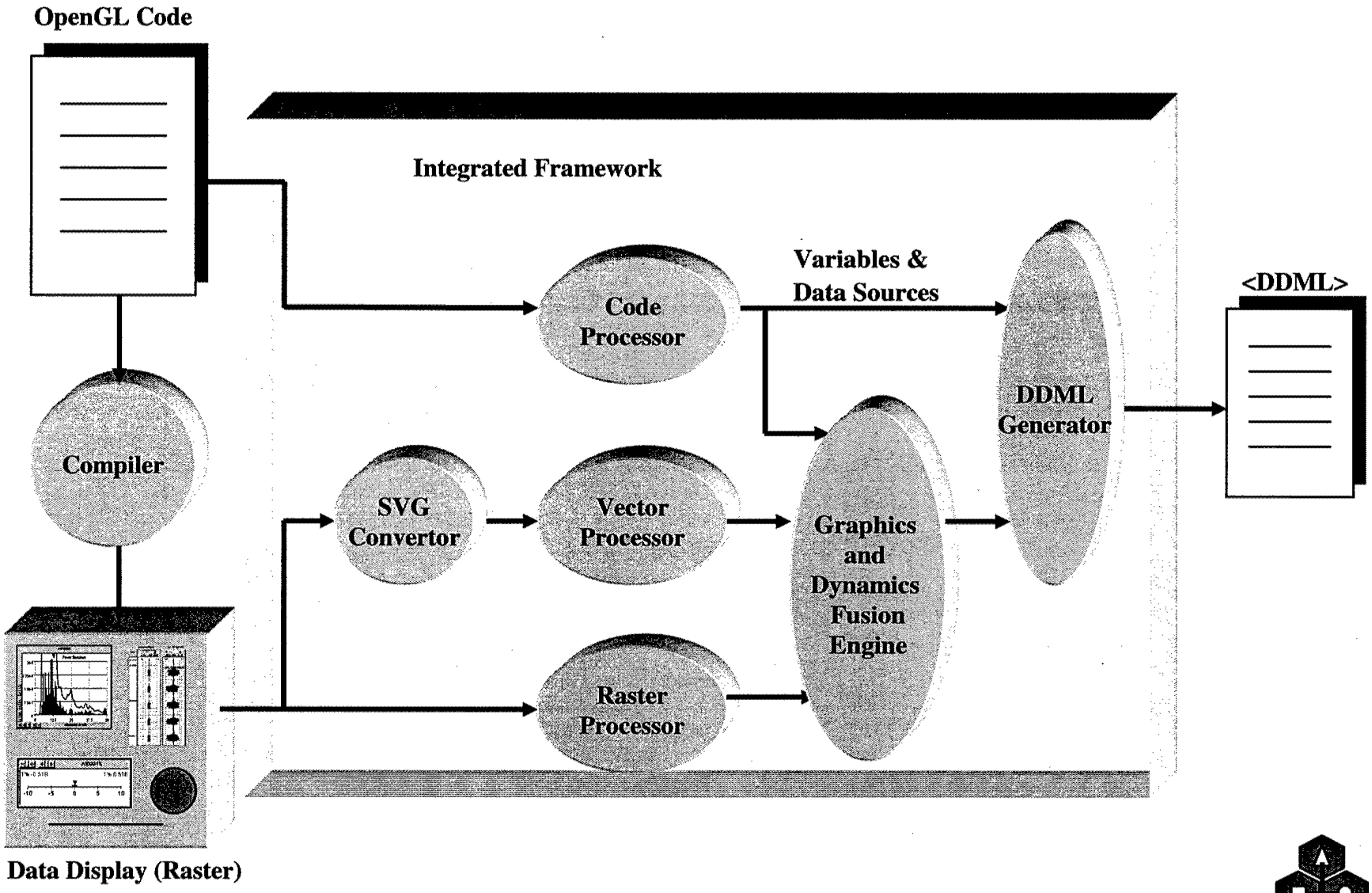
- Layers similar to typical software architecture



Automated Extraction of Display Configurations

- Different approaches
  - Work off raster images
  - Vectorize raster and work off vector graphics
  - Parse source code and extract display configuration
- Strength and weakness for each approach
- Need to fuse results

Overview



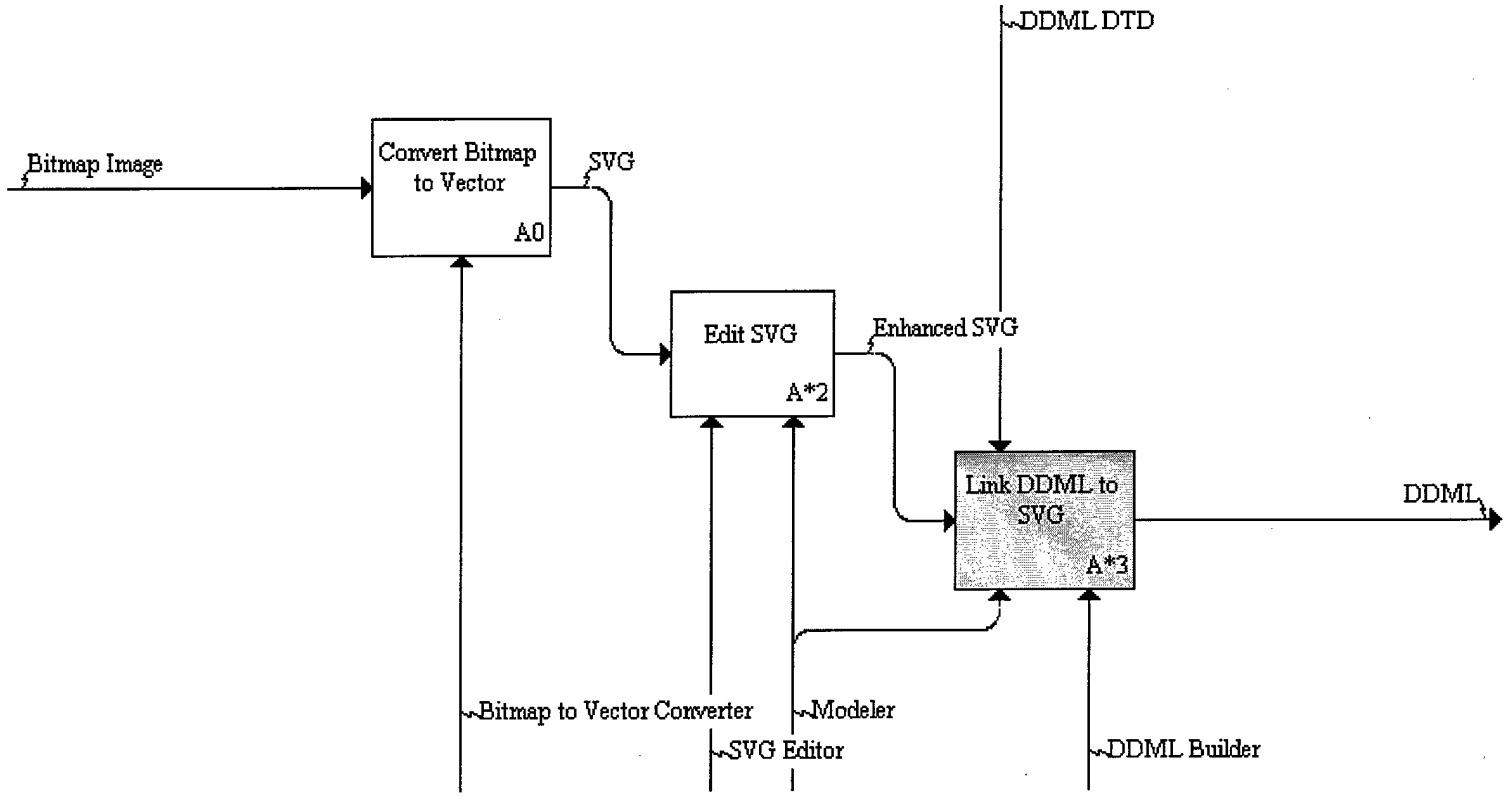
Comparison of Translatability

DDML layer	Raster Graphics	Vector Graphics	Parsing Source Code
Graphics Resources	partial	partial	partial
Dynamics	partial, with numerous samples in color	partial, with numerous samples in color	yes
Data Variables	no	no	yes
Data Sources	no	no	yes

Raster-based Object Recognition

- Raster-based shape recognition
  - Used for high-level objects, e.g., slider
- Image processing toolbox
  - e.g., Matlab Image Processing Toolbox
  - e.g., Manto
- Use of Neural Networks
  - e.g., NeuroSolutions

Vector-based Object Recognition



Code Parsing

- Assumptions
  - Well-defined and structured code with certain style
  - Human assisted program flow understanding
  - Easier to parse object-oriented code
  - Examples of places to look for clues
    - Make use of "#include" header and Makefiles
    - Defined variables become variables in DDML
    - Look for dynamics in if-then-else statements involving variable use in the antecedents (if-part)

Comparison of benefits and risks

DDML layer	Raster Graphics	Vector Graphics	Parsing Source Code
Ease	moderate	moderate	Most difficult
Risks	Relatively low	Relatively low	High; depends on assumptions

Model Fusion

- Human assisted
  - Get graphics from vector/raster and variables/data sources from code parsing
  - Modify DDML accordingly
- Bayesian techniques
- Dempster-Shafer theory of evidence

Benefits

- Supports interoperability of display systems
- Can be used to automate conversion to object-oriented programming
  - Improvement/modification of a high-level display object does not affect rest of code
- Uses beyond data display systems
  - Code re-engineering
  - Legacy code conversion

Conclusions

- Automated extraction of data display configuration from applications
- Represent in DDML, a generic XML-based neutral language for data displays
- Use stored configuration in DDML for other T&E environments and centers