Advanced Relation Model for Program Visualization (ARM 4 PV)

Brian J. d'Auriol
Department of Computer Science
The University of Texas at El Paso
El Paso, TX, USA 79968
Email: dauriol@acm.org

***Abstract*— The Advanced Relation Model for Program Visualization (ARM 4 PV) is proposed as a framework model for analysis and visualization of computer programs. The ARM 4 PV is composed of two sub-systems. The systems sub-system provides for the identification, extraction, representation and visualization of relational information about the program. The visualization sub-system provides for various different types of visualizations that can be incorporated into the systems sub-system. A detailed overview of the ARM 4 PV is given. The format and representation of relational information is presented. Several visualizations that show the application of the ARM 4 PV are also presented.**

I. Introduction

Visualization is generally defined as the representation of information or data in the form of images or graphics for a better understanding of the data, see for example [1], [2]. Visualization is useful to perceive patterns in the data, especially in large data sets. Visualization has been applied in many fields, including, scientific visualization and program visualization. Program visualization, often considered to be within software visualization and information visualization, facilitates the understanding of program code, see for example [3], [4], [5].

Program understanding is commonly facilitated by good source code layout, good data and procedural abstractions and good internal and external documentation. Many common techniques such as pretty-printing, indentation and the use of extra blank lines provide visual clues about the meaning and purpose of code fragments. Data and procedural abstractions in the code are captured by data structures and the subdivision of operations into tasks, objects and procedures. Other techniques such as commenting, appropriate identifier naming and external documentation provide heightened information about the concepts inherent in the program. However, despite the long history of writing readable code, understanding programs remains a daunting task: large code size, many supported functions and, especially, the additional complexities found in distributed and web-based computing, all contribute to difficulties in understanding programs. The purpose of the Advanced Relation Model for Program Visualization (ARM 4 PV) is to provide a framework model that provides for various types of visualizations which can be universally applied to programs in order to facilitate program understanding.

This paper is organized as follows. The next section, Section II, presents a detailed overview of the ARM 4 PV. Applications of the visualization models within the ARM 4 PV are briefly presented in Section III. Conclusions are given in Section IV.

II. ARM 4 PV Overview

The Advanced Relation Model for Program Visualization, ARM 4 PV, is a framework model for analysis and visualization of computer programs. Figure 1 shows the sub-systems and various models that make up the ARM 4 PV. There are two sub-systems. The systems sub-system, shown in orange-black, provides for the identification, extraction, representation and visualization of relational information about the program. This sub-system is composed of five models: the Static Relation Model (SRM), the Dynamic Relation Model (DRM), the Data Repository (DR), the Relation Reduction Model (RRM) and the Visualization Model (VM). The visualization sub-system, shown in blue-black, provides for various types of visualizations. This sub-system currently has four models: the Geometric Representation of Programs (GRP) model, the Program Scientific Visualization (PSV) model, the Parameterized Iconic Glyph (PIG) model and the Conceptual Crown Visualization (CCV) model. These four models can be 'inserted' into the VM so as to provide various different visualizations of the same program related information. As such, the ARM 4 PV allows for other visualization models to be defined. The remainder of this section describes the various aspects of the ARM 4 PV.

Relations in a hierarchy are defined as the primary datum in the ARM 4 PV. Relations are denoted by R^l where l refers to the level of the relation in the hierarchy. The hierarchy itself is composed of a set of relations $\{R_i^1, R_j^2, \ldots, R_k^l\}$ where the subscripts are integers $i \geq 1, j \geq 0, k \geq 0$ that denote particular relations in the specified level. A relation's level in the hierarchy describes the relative abstractness of that relation. There are two types of relations. *Semantic relations* bind semantics to individual program statements: $R^1 = (S, d)$ where S is a program statement and d is the statement's semantics (e.g. its operational semantics). *Constructive relations* are higher-order relations that extend the semantics by creating more abstract descriptions that apply to the group of participating relations: $R^l(R^i, R^j, d_\alpha)$ where $l > 1$ and $i, j < l$ such that either $i = l - 1$ or $j = l - 1$ and d_α is an abstraction that includes the semantics of both the

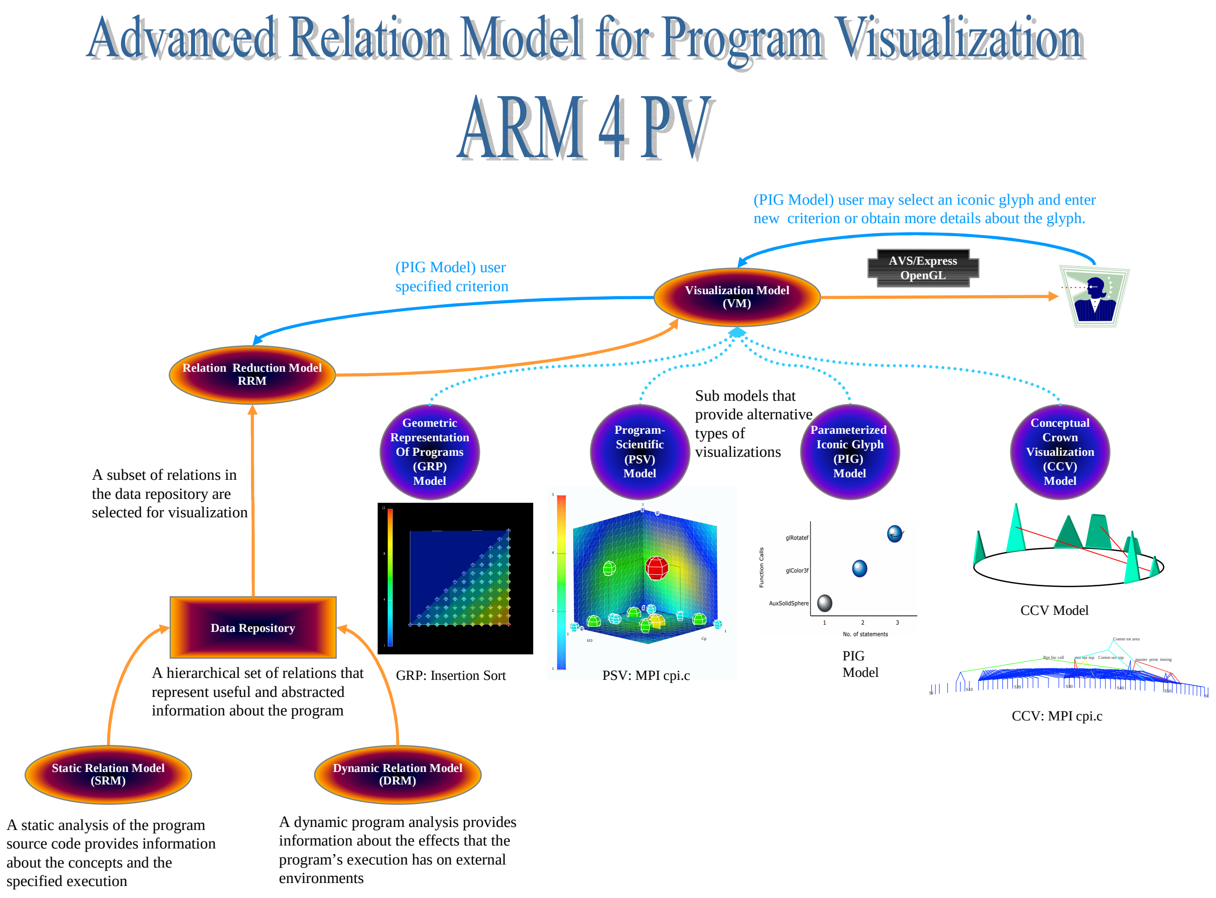
Advanced Relation Model for Program Visualization
ARM 4 PV
(PIG Model) user may select an iconic glyph and enter new criterion or obtain more details about the glyph.
AVS/Express OpenGL
(PIG Model) user specified criterion
Visualization Model (VM)
Relation Reduction Model RRM
Sub models that provide alternative types of visualizations
Geometric Representation Of Programs (GRP) Model
Program-Scientific (PSV) Model
Parameterized Iconic Glyph (PIG) Model
Conceptual Crown Visualization (CCV) Model
A subset of relations in the data repository are selected for visualization
Data Repository
GRP: Insertion Sort
PSV: MPI cpi.c
PIG Model
CCV Model
CCV: MPI cpi.c
A hierarchical set of relations that represent useful and abstracted information about the program
Static Relation Model (SRM)
Dynamic Relation Model (DRM)
A static analysis of the program source code provides information about the concepts and the specified execution
A dynamic program analysis provides information about the effects that the program's execution has on external environments

Fig. 1. Advanced Relation Model for Program Visualization (ARM 4 PV)

participating relations as discussed subsequently. The most abstract relations, i.e., at the highest level, are called *concepts*.

Let τ denote a semantic operation that combines two semantics and generates an abstraction of those combined: $\tau(d_1, d_2) = d_\alpha$. For example, if d_1 is 'prompt user for input' and d_2 is 'read input value', then $\tau =$ 'interactive user input'. At present, the ARM 4 PV defines the requirements for these semantic operations, however, a semantic engine to accomplish this is still being investigated. The semantic propagation results in more and more abstract descriptions at higher levels of the code fragments.

The relation hierarchy represents the information about the program source code. Since semantic information is bound at the lowest level and is propagated up-wards in more abstract terms, the highest levels of relations consist of very abstract descriptions. If the hierarchy is *complete*, then the abstract descriptions of the highest levels represent the composition of all Level 1 relations, that is, represents the purpose of the program. This condition is reflective of a well developed program where all the statements contribute to a select few concepts about the whole code. A complete hierarchy depends on two things, first, the code is well developed, second, the processes used to construct the hierarchy lead to a complete hierarchy. This provides for a way to measure the hierarchy building processes. Assume a set of well developed codes, then construct the hierarchy, lastly, analyze the hierarchy to ensure that each program statement is recursively participating in the highest level concepts. In some cases, a complete hierarchy is not necessary in which to abstract the program's purposes. A *near complete* hierarchy is one in which sufficient but not necessarily all Level 1 relations participate in the highest level concepts. A *partially complete* hierarchy is one in which highest level concepts are supported by a subset of Level 1 relations. All of the visualization models use the relations in the hierarchy as the input information to visualize. Hence, it is important that the relation hierarchy be complete or near complete. Partially complete hierarchies may be used to highlight specific regions in the code that are of interest; however, the danger is that some useful information about the program may not be available to the viewer.

The processes of identifying relations inherent in a given program source are performed in the SRM and DRM. These models output the multi-level hierarchy of relationships which are then stored in the DR. At present, the SRM is composed of two sub-models, the Cluster Relation Model (CRM) and the Graph Relation Model (GRM). These two sub-models perform, respectively, cluster analysis and graph based analysis, of the input source code. Further preliminary discussion appears in [6]. An implementation of the DRM is presented in [7]: "The main focus of the [DRM implementation] is on analyzing different types of information during program execution". The output of the DRM implementation is suitable to be incorporated into the DR.

A complete or near complete hierarchy will likely have a very large relational set. The RRM acts as a filter between the DR and the VM and selects or reduces the number of relations that are visualized. This is needed so that the visualization models can perform efficiently in real-time.

The VM is a container model. It defines the relational hierarchy as the input to the visualization and the output as suitable data that is rendered by a graphics or visualization tool. Currently, AVS/Express is used, and OpenGL based applications are being considered.

The Geometric Representation of Programs (GRP) model provides for initial program specification in a geometric framework. The basic premise is that programming may be considered to consist of two fundamental actions: (a) the grouping of program statements, and (b) the forming of relationships within a group or between groups. Statements in a group are mapped to points (usually integer) in an n-dimensional metric space; the grouping itself is formed as a convex polytope of such points. Relationships between the computations in and between groups are represented as graphs over the internal vertices of the polytope. GRP is defined as a stand-alone model, however, it can be incorporated into the ARM 4 PV as follows. Let Level 1 relations be mapped to points inside the polytope. Select the highest level concept that forms a complete or near complete hierarchy of all the relations that are mapped to points; i.e., this concept becomes the abstraction associated with the polytope, and hence, with the group of statements. Early work on the model is reported in [8], [9].

The PSV model uses traditional scientific visualization techniques like contouring to provide insight into program related information. Each relation in the hierarchy is augmented to include an attribute vector v: $R^l = (\dots, v)$. Although v could be associated with relations in the DR, it is better to augment the DR's relation representation to include v. For Level 1 relations, the values in v can be determined by the SRM and DRM analyses processes. For higher level relations, a combining operation can be defined to propagate the values up-wards in the hierarchy. The combining operation used in the applications in this paper is averaging:

Given

$$R^1_1 = (S_1, d_1, (e_{1,1}, e_{1,2}, e_{1,3})$$
$$R^1_2 = (S_2, d_2, (e_{2,1}, e_{2,2}, e_{2,3})$$

then

$$R^2_1 = (R^1_1, R^1_2, d_\alpha, \left(\frac{(e_{1,1}+e_{2,1})}{2}, \frac{(e_{1,2}+e_{2,2})}{2}, \frac{(e_{1,3}+e_{2,3})}{2}\right) \quad (1)$$

where d_α represents the semantic abstraction of d_1 and d_2. Further details are described in [10].

The Parameterized Iconic Glyph (PIG) model defines a visual entity composed of an icon and a glyph. The icon is a picture representing a high level concept. The icon is 'painted' on a glyph, say a rectangle, where attributes associated with the concept are mapped to the glyph attributes, in this example, say the concept's level is associated with the height of the rectangle. Iconic glyphs can be graphed in 2-D or 3-D plots that associate concepts together. The PIG model may also allow for interactive selection of relations, that is, may data mine the DR through the RRM. Further discussion is found in [6].

The CCV model provides concept visualizations to facilitate the viewer's better understanding of the concepts inherent in the programming. There are two basic visualizations that are defined: a line structure and a space structure visualization. The former renders selected relations in the relation hierarchy as either single vertical lines (Level 1) or dual piece-wise single point-connected lines (higher levels) whereas, the latter renders concepts as a convex hull of the participating lower-level relations. Relations are graphed in the $x-y$ plane. The linear x-axis is continuous. In this paper, Level 1 relations are represented as vertical lines, $\overline{(S_i, 0), (S_i, 1)}$. This means that program statements are mapped to integer coordinates in their lexicographic order. A higher level relation, $R^l = (R^i, R^j, d)$ for $l > 1$, is represented as: $\overline{(x_1, i)(x_2, l)}$, $\overline{(x_2, l)(x_3, j)}$ where (x_1, i) and (x_3, j) are points for relations R^i and R^j, respectively; and, x_2 is the midpoint of the minimum and maximum extents of all the participating Level 1 relations in the R^l. For perceptive reasons, the x-axis is mapped to a circle thereby creating a cylinder shaped visual object. This enables: (a) immersive visualization by allowing the viewpoint to be placed in the center of the circle, (b) a zooming operation by providing greater forefront focus while compressing the information in the peripheral area, and (c) greater information density by providing up to twice the amount of information displayed on the screen. Further details are described in [11].

III. Visualization Model Applications

This section briefly describes several applications. The first part show cases a very simple file processing program together with a partially complete relation hierarchy. A PSV visualization output is also described. The second part consists of a CCV visualization of a different program as illustration of different aspects of the ARM 4 PV model.

A. File Processing Example [10]

The program in Figure 2 is selected as a simple example of how the ARM 4 PV may be applied. This program consists of seven statements such that the `while` loop in Statement 3

```
1.  fileopen("foo.bar",read-acCess)
2.  printf("some column headings")
3.  while not eof on file do
4.     readfile
5.     printf(detail report)
6.  end-while
7.  close file
```

Fig. 2. File processing pseudocode program.

TABLE I

TABLE FOR PROGRAM STATEMENT RELATIONS

Level1	**File I/O**	**User Output**	**Comp**
(R^1_1,S2,print column headings)	0.0	1.0	0.0
(R^1_2,S5,print data in columns)	0.0	1.0	0.0
(R^1_3,S3,eof query on file)	0.25	0.0	0.0
(R^1_4,S3,repeat loop-body)	0.0	0.0	0.1
(R^1_5,S4,read a record from a file)	1.0	0.0	0.0
Level2			
(R^2_1,R^1_1,R^1_2,print report)	0.0	1.0	0.0
(R^2_2,R^1_3,R^1_4,repeat eof query)	0.125	0.0	0.05
(R^2_3,R^1_4,R^1_5,repeat file read)	0.5	0.0	0.05
Level3			
(R^3_1,R^2_2,R^2_3,process every record in file)	0.31	0.0	0.05
Level4			
(R^4_1,R^2_1,R^3_1,general file report)	0.15	0.5	0.025

contains double semantics, i.e., this statement is both a repetition as well as a file input/output. Only static analysis is considered in this example. The SRM (which is responsible for generating a relation hierarchy) is applied directly on this program. Currently, the SRM is not implemented, hence, the relation hierarchy is hand-crafted.

The relation hierarchy is updated with $v = (\text{file I/O, user output, computation})$ and is shown in Table I. A pictorial representation of the hierarchy is shown in Figure 3.

Figure 4 shows a PSV model visualization for this program. The color of a glyph is mapped to the level of the corresponding relation in the hierarchy. The concentration of the glyphs on the left hand side of the figure indicates that file I/O constitutes the bulk of the *processing* in this program. This observation is different than the high level abstraction of the program shown by the red glyph representing R^4_1. The most abstract description of this program is the significant user output operations combined with some file input/output and computational processing. In other words, the purpose of this program is to print a report based on information in a file; the most important abstraction is the report generation, but the bulk of the processing requires file input.

B. Swap Program Example

This example illustrates a CCV visualization. A small C program that performs an exchange of two user inputs is shown in Figure 5. Figures 6 and 7 show CCV line structure and space structure, respectively, visualizations of this program.

Some summary comments regarding this program are:

- there is a distinct identification of the abstractions related to each function as well as that for the whole program,

$$\begin{tikzcd}
\text{Level 1} & \text{Level 2} & \text{Level 3} & \text{Level 4} \\
R^1_1 \arrow[r] & R^2_1 \arrow[rrd] & & \\
R^1_2 \arrow[ru] & & & R^4_1 \\
R^1_3 \arrow[r] & R^2_2 \arrow[r] & R^3_1 \arrow[ru] & \\
R^1_4 \arrow[ru] \arrow[r] & R^2_3 \arrow[ru] & & \\
R^1_5 \arrow[ru] & & &
\end{tikzcd}$$

Fig. 3. Pictorial display of relation hierarchy for Figure 2.

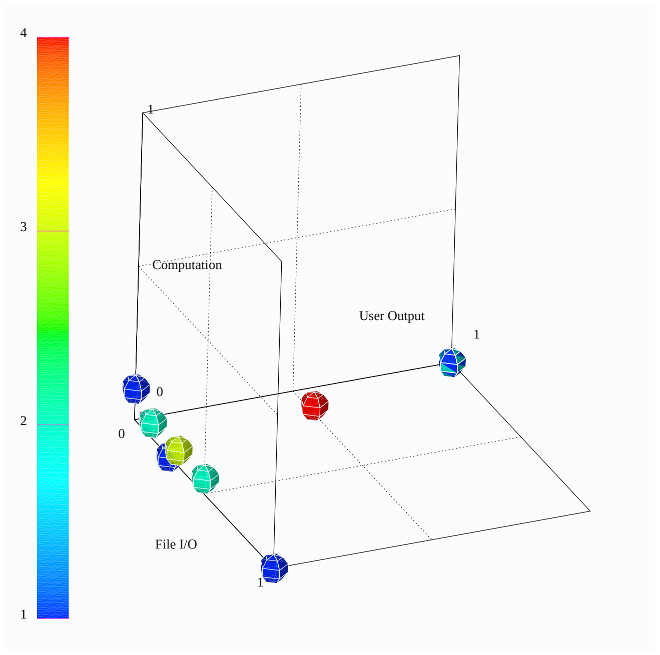

Fig. 4. PSV visualization of the file processing pseudocode program.

```
#include <stdio.h>
void swap(float *x, float *y)
{
        float t;
        t = *x;
        *x = *y;
        *y = t;
        printf("Values in swap %f, %f\n", *x, *y);
}
main()
{
        float x, y;
        printf("Please input 1st value: ");
        scanf("%f", &x);
        printf("Please input 2nd value: ");
        scanf("%f", &y);
        printf("Values BEFORE swap %f, %f\n", x, y);
        swap(&x, &y);
        printf("Values AFTER swap %f, %f\n", x, y);
        return 0;
}
```

Fig. 5. Swap program example.

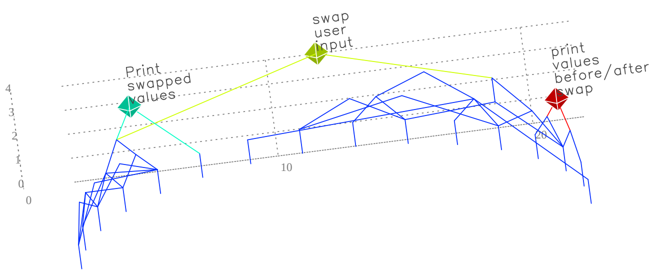

Fig. 6. CCV line structure visualization of swap program.

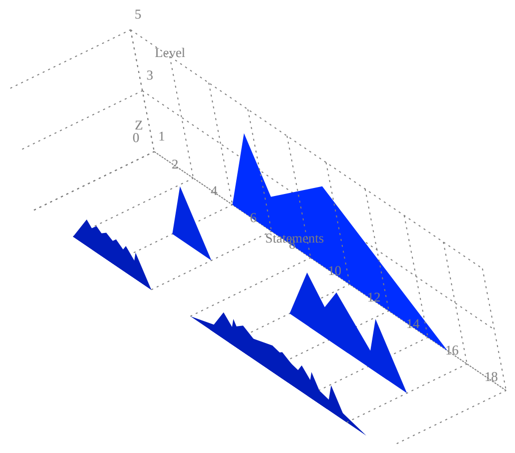

Fig. 7. CCV space structure visualization of swap program.

- there is a distinct separation in the abstractions related to the two functions, and
- there is an assessment regarding the formation of these abstractions, and hence, the program: the abstractions of the swap function as well as that for the main function are not well formulated (the first is due to the duplicated `printf`, Statement 7 while the latter may be due to the fact that a partially complete hierarchy has been used).

IV. Conclusions

This paper presents the Advanced Relation Model for Program Visualization, ARM 4 PV as a framework model in which various visualization models can be defined. The proposed model is based on the approach that a set of relations and higher-order relations that describe levels of abstractions of programs can be visually presented to a viewer in order to facilitate better program comprehension. This paper also includes two small examples of the applications of the ARM 4 PV that illustrate the Program Scientific Visualization (PSV) and the Conceptual Crown Visualization (CCV) models. Concept visualizations based on the ARM 4 PV are further discussed in [12] while a detailed case study of the visualization of an MPI parallel program appears in [13].

The ARM 4 PV is currently and specifically tailored for program visualization, hence, the full title of the proposed model. Since the ARM 'part' of the model abstracts the processes inherent in information identification and analysis, it should be applicable to information domains other than program visualization. For this reasons, the term 'advanced' is meaningfully descriptive of the proposed model.

There are currently several critical issues that are being investigated regarding the ARM 4 PV: first, the SRM and DRM together with the DR need to be automated, in particular to also include a semantic engine that will perform the functions of τ; second, the visualization models need to be further explored and enhanced.

References

[1] L. Birdsong, G. Helms, and G. Program, "Chapter ii ,visualization and scientific visualization defined," Jul 2000, http://edmall.gsfc.nasa.gov/99invest.Site/VISUALIZATION/visualization.html#2

[2] G. Owen, "Definitions and rationale for visualization," 1999, http://www.siggraph.org/education/materials/HyperVis/visgoals/definiti.htm

[3] G.-C. Roman and K. C. Cox, "Program visualization: The art of mapping programs to pictures," in *14th International Conference on Software Engineering*, A. Press, Ed., Melbourne, Australia, May 1992, pp. 412–420.

[4] J. Stasko, J. Domingue, M. H. Brown, and B. A. Price, Eds., *Software Visualization, Programming as a Multimedia Experience*. The MIT Press, 1998.

[5] S. Tilley and H. Shihong, "On selecting software visualization tools for program understanding in an industrial context," in *Proc. of the 10th International Workshop on Program Comprehension*, June 2002, pp. 285–288.

[6] E. W. Y. Tai, L. Thompson, R. Smith, and B. J. d'Auriol, "Program visualization using avs/express," 2001, http://www.cra.org/Activities/craw/creu/crewReports/2002/texas_final.html

[7] R. Seth, "Dynamic relation model for program visualization," Master's thesis, Department of Computer Science, The University of Texas at El Paso, December 2003.

[8] B. J. d'Auriol, "Expressing parallel programs using geometric representation: Case studies," in *Proc. of the IASTED International Conference Parallel and Distributed Computing and Systems (PDCS'99)*, Cambridge, MA, USA, Nov. 1999, pp. 985–990.

[9] ——, "A geometric semantics for program representation in the polytope model," in *Proc. of the Twelfth International Workshop on Languages and Compilers for Parallel Computing LCPC'99, Aug. 4-6, The University of California, San Diego, La Jolla CA USA*, ser. Lecture Notes in Computer Science, Vol. 1863. Springer-Verlag, 1999, pp. 451–454.

[10] R. Policherla, "Scientific visualization techniques for program visualization," Master's thesis, Department of Computer Science, The University of Texas at El Paso, May 2004.

[11] A. Gajjala, "A model for visualization of program conceptual information," Master's thesis, Department of Computer Science, The University of Texas at El Paso, May 2004.

[12] B. J. d'Auriol, "Concept visualizations of computer programs," in *Proc. of The Conference on Advances in Internet Technologies and Applications (CAITA 2004)*, Purdue University, West Lafayette, Indiana, USA, July 2004, (to appear).

[13] ——, "A concept visulaization study of a parallel computing program," in *Proc. of the 6th International Workshop on High Performance Scientific and Engineering Computing with Applications (HPSEC-04) in conjunction with the International Conference on Parallel Processing (ICPP-2004)*, Montreal, Quebec, Canada, Aug. 2004, (To appear).