PICS un Sistema de Comprensión e Inspección de Programas

Mario M. Berón
Universidad Nacional de San Luis - Departamento de Informática
San Luis - Argentina
mberon@unsl.edu.ar
Pedro R. Henriques
Universidad de Minho - Departamento de Informática
Braga - Portugal
prh@di.iminho.pt
Maria J. Varanda
Instituto Politécnico de Bragança - Departamento de Informática
Bragança - Portugal
mjoao@ipb.pt
Roberto Uzal
Universidad Nacional de San Luis - Departamento de Informática
San Luis - Argentina
ruzal@sinectis.com.ar

Resumen

La Comprensión de Programas (CP) es un área de la Ingeniería del Software que tiene como objetivo el estudio y creación de modelos, métodos, técnicas y herramientas con la finalidad de facilitar el entendimiento de los sistemas de software. La CP es útil para mantenimiento, reingeniería, ingeniería reversa, entre otras tantas aplicaciones.

Para construir herramientas de comprensión de programas es necesario concebir esquemas que permitan extraer y visualizar la información de los sistemas. PICS es una herramienta de CP basada en la presentación de diferentes perspectivas o vistas del sistema de estudio. Estas vistas tienen como propósito interrelacionar distintos dominios útiles para la comprensión. En este contexto, la principal fortaleza de PICS es alcanzar una estrecha relación entre el dominio del programa y el comportamiento del sistema. Esta relación falicita la comprensión porque permite asignar semántica a las componentes del dominio del programa.

En este artículo describimos PICS, sus funcionalidades de inspección, comprensión y las estratégias implementadas en su núcleo que permiten alcanzar la interrelación de dominios. En este último caso, hacemos énfasis en la interrelación entre el dominio del programa y el comportamiento del sistema debido a la gran importancia que tiene esta relación para la CP.

Palabras Claves: Comprensión de Programas, Métodos, Técnicas, Herramientas.

1. INTRODUCCIÓN

La comprensión de programas se traduce en la habilidad de entender una pieza de código escrito en un lenguaje de alto nivel. Un programa no es mas que una secuencia de instrucciones que serán ejecutadas de forma de garantir una determinada funcionalidad. El lector de un programa consigue extraer el significado del mismo cuando comprende de que forma el código cumple con la tarea para la cual fue creado. El área de comprensión de programas es una de las más importantes de la Ingeniería del Software porque es necesaria para tareas de reutilización, inspección, manutención, migración y extensión de sistemas de software. Puede también ser utilizada en áreas como ingeniería reversa o enseñanza de lenguajes de programación. La tarea de comprensión de programas puede tener diferentes significados y puede ser vista desde diferentes perspectivas. El usuario puede estar interesado en como la computadora ejecuta las instrucciones con el objetivo de comprender el flujo de control y de datos, o puede querer verificar los efectos que la ejecución tiene sobre el objeto que esta siendo controlado por el programa. Considerando estos niveles de abstracción, una herramienta versátil de inspección visual de código es crucial en la tarea de comprensión de programas [13].
En este artíulo se presenta PICS (**P**rogram **I**nspection and **C**omprehension **S**ystem) una herramienta destinada a facilitar el proceso de comprensión de programas escritos en lenguage C. PICS aborda este desafío por medio de la presentación de diferentes vistas. Estas vistas representan al sistema de estudio en distintos dominios y en diferentes niveles de abstracción. La presentación de vistas por si sola no es suficiente son necesarios mecanismos de navegación entre ellas. Esto posibilita que el programador pueda acceder a un dominio específico y cuando considere necesario acceder a otro con facilidad. Esta característica facilita el aprendizaje debido a que fomenta la interrelación de conceptos. De se esta manera se reduce la brecha existente entre los conocimientos del programador y los conceptos subyacentes del sistema.
Una de las relaciones más importantes es la que vincula el dominio programa y el dominio del problema. Podemos realizar esta afirmación porque dicha vinculación permite identificar claramente las operaciones realizadas por el sistema y su efecto comportamental. De esta forma la tarea de modificar, documentar o actualizar una funcionalidad es más leve. Esto se debe a la posibilidad acceder solamente a las componentes del sistema utilizadas para producir el comportamiento bajo análisis.
Para alcanzar la interrelación de dominios, PICS utiliza técnicas de compilación que permiten extraer información estática y dinámica. Luego construye vistas textuales y basadas en grafos que representan al sistema o parte de él en otros dominios [3]. Además, PICS implementa una estrategia de relación comportamental-operacional denominada BORS (**B**ehavioral-**O**perational **R**elation **S**trategy) que posibilita encontrar la relación entre las componentes del dominio del programa y el comportamiento del sistema.
Este artículo esta organizado de la siguiente manera. La sección 2 presenta los trabajos relacionados. La sección 3 describe la arquitectura de PICS. La sección 4 explica las técnicas de compilación utilizadas para la extracción de información estática y dinámica. La sección 5 describe brevemente la información disponible en PICS y la forma de administrarla. La sección 6 detalla las vistas y sus funcionalidades. La sección 7 explica BORS. Finalmente, la sección 8 expone la conclusión de este artículo.

2. Trabajos Relacionados

Existen diversas herramientas destinadas a inspeccionar y comprender programas. En nuestras investigaciones hicimos énfasis en aquellas destinadas a estudiar programas escritos en lenguaje C. Esto se debe al deseo de proponer estrategias para estudiar software de base (normalmente escrito en C) y sistemas legado.

En este contexto podemos decir que Understand C/C++ [5] es una herramienta que permite la exploración de sistemas a través de diferentes vistas interesantes. Por ejemplo: visualización de la

relación de llamadas a funciones como un árbol, formularios de presentación de la información de los identificadores, cómputo y visualización de métricas, construcción de diagramas de flujo del programa, etc.

CodeSurfer [6] es otra aplicación que posee características similares a Understand C/C++. La extracción de la informacón es llevada a cabo usando una representación del sistema basada en grafos. Dicha representación permite extraer con facilidad información estática. Una característica saliente de esta aproximación es la simplicidad con la que se pueden computar sclicing de programas.

Imagix 4D [7] realiza el mismo tipo de tareas que las herramientas anteriores pero presenta innovaciones en la visualización del programas a través del uso de representaciones gráficas en tres dimensiones.

CScope [8] es una aplicación antigua que también es útil para inspeccionar programas. La interfaz de este software es muy simple debido a la ausencia de visualizaciones. Sin embargo, las funciones de búsqueda y navegación la hacen atractiva para estudiar programas escritos en C. La información que se puede recuperar consiste en los atributos estáticos de cada entidad del sistema de estudio.

SHriMP (Simple Hierachical Multi-Perspective) [10] es un ambiente para explorar grandes volumenes de información. Esta herramienta basa su potencia en la presentación de las relaciones entre las entidades del sistema utilizando grafos anidados. Además tiene incorporadas atractivas funciones de navegación. No obstante, recupera la misma clase de información que las herramientas descriptas previamente.

JGrasp [9], es otro software destinado a facilitar la comprensión y exploración de programas. Esta aplicación esta implementada en Java y puede ser aplicada a diferentes lenguajes como C, C++, Java, Ada, VHDL, etc. La innovación presentada en este caso esta en la visualización de diagramas de flujo junto con el código fuente. Por otra parte, dicho software permite la construcción de ambientes personalidados en donde se puede observar información estática y dinámica del sistema.

Alma [4] es una aplicación cuyo objetivo es facilitar el aprendizaje de la programación a través de la comprensión de las característcas escenciales de los lenguajes de programación. Este software alcanza su objetivo a través de la simulación del programa de estudio. Alma es aplicable a cualquier lenguaje de programación. Para lograr este objetivo, es necesario la construcción de un front-end del lenguaje destino. Además de recuperar la información de las entidades del programa, Alma permite visualizar la ejecución del programa utilizando el árbol de sintáxis abstracta y las modificaciones realizadas sobre esa estructura de datos a medida que el programa ejecuta.

SeeSoft [4], es una herramienta antigua pero muy interesante debido a que sus visualizaciones hacen un fuerte uso de los atributos del texto del programa. El objetivo principal es facilitar la comprensión a través de una clara presentación del código fuente.

Luego del estudio de estas aplicaciones observamos que: i) Muchas de ellas se basan en la recuperación de la información estática del los sistemas, ii) Presentan vistas muy útiles para la comprensión, iii) No se pudo encontrar una explicación clara acerca de las componentes necesarias para la construcción de herramientas de comprensión, iv) En general y como punto más saliente presentan escasas estrategias para relacionar el domino del problema y el dominio del programa. Esta característica es relevante debido a que la misma es considerada el gran desafio en Comprensión de Programas.

3. ARQUITECTURA DE PICS

La Figura 1 muestra la aquitectura de PICS. El lector puede observar que la herramienta posee las siguientes componentes:

Módulos para extracción de la información: Estos módulos extraen información estática y dinámica del sistema de estudio. Para la extracción de la información estática, se utilizan técnicas de compilación tradicionales. Para la recuperación de la información dinámica, se instrumenta el código fuente con funciones de inspección. Este esquema de instrumentación será descripto brevemente en la sección 4.

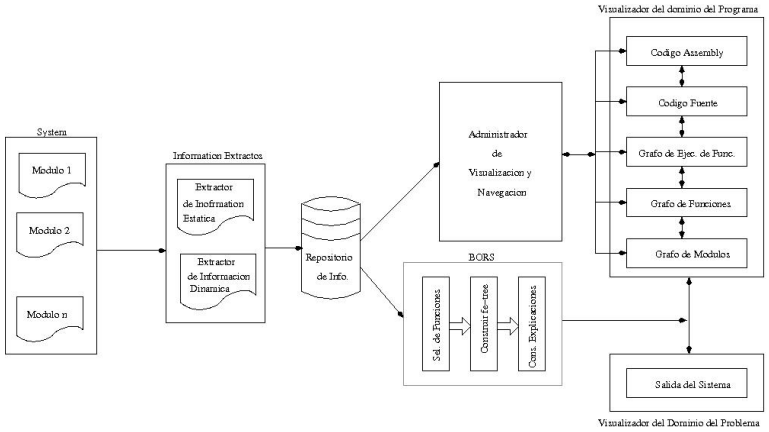

Figura 1: Arquitectura de PICS

Repositorio de Información: Esta componente posibilita el almacenamiento de toda la información extraída. Los módulos principales que acceden a este repositorio son el: *Administrador de Visualización y Navegación* y *BORS*. Esta componente será presentada brevemente en la sección 5.

Administrador de Visualización y Navegación: La función principal de esta componente (ver sección 5) es la de posibilitar la construcción de las vistas disponibles en PICS. Además permite la navegación entre ellas.

Visualizadores: consta de varias componentes agrupadas en dos categorías (Visualizaciones del Dominio del Programa y Visualizaciones del Dominio del Problema). Cada componente implementa las distintas vistas disponibles en PICS. Cada una de ellas, además de construir la vista apropiada usando la información provista por el *Administrador de Visualización y Navegación*, presentan al usuario una visión gráfica o textual dependiendo de la clase de perspectiva que se utilize. Las vistas provistas por PICS serán descriptas en sección 6.

BORS: Este módulo es el encargado de implementar la estrategia de relación comportamental-operacional. En otras palabras, permite relacionar el dominio del problema con el dominio del programa. La estrategia *BORS* es compleja y será descripta en la sección 7.

4. EXTRACCIÓN DE LA INFORMACIÓN

Para la extracción de la información estática PICS utiliza técnicas de compilación tradicionales. Es decir, la herramienta tiene implementado un analizador sintáctico de ANSI-C con las acciones semánticas necesarias para la recuperación de: funciones, datos, tipos, etc.
Para el caso de la recuperación de la información dinámica [12] [15] [2] se utilizó *instrumentación de código*. Esta técnica consiste en insertar sentencias útiles dentro del código fuente del sistema. Esto tiene por objetivo extraer la información deseada a medida que el sistema se ejecuta.
El esquema utilizado para este caso es simple: *se insertan funciones de inspección en el comienzo y fin de las funciones*. Estas funciones de inspección tienen como tarea imprimir el nombre de la función ejecutada y alguna otra información que el usuario desea capturar. Este esquema se debe completar con una estrategia para controlar el número de funciones recuperadas por las funciones de inspección. Esto se debe a que las funciones pueden ser invocadas dentro de iteraciones. Las iteraciones pueden repetirse muchas veces porque normalmente son utilizadas para inicializar estructuras de datos, o porque forman parte de un algoritmo complejo. En este caso, las funciones del sistema se invocaran en

forma repetitiva y las funciones de inspección registrarán este hecho. Por esta razón, la información recuperada será enorme. Para resolver este problema las iteraciones son controladas insertando código antes dentro y después de ellas. El código preliminar a las iteraciones indica, a las funciones de inspección, el número de veces que las funciones invocadas dentro del cuerpo de la iteración deben ser reportadas. Este número es almacenado en una pila debido a que las iteraciones pueden estar anidadas. El código dentro de las iteraciones tiene como objetivo decrementar el número que está en el tope de la pila (número de veces que las funciones dentro de la iteración pueden mostrarse). Cuando este valor llega a cero las funciones de inspección no recuperan los datos de las funciones invocadas. Finalmente, el código insertado después de las iteraciones tiene como finalidad recuperar el número de veces que las funciones de la iteración anterior debe mostrarse. Dicho valor se encuentra en el tope de la pila de control de ciclos. La Figura 2.a muestra el esquema para las funciones de inspección y la Figura 2.b muestra el esquema descripto para las iteraciones. El lector interesado en una descripción detallada de esta aproximación puede leer [12].
Para finalizar esta sección es importante notar que toda la información (estática y dinámica) es almacenada en el *Repositorio de Información*. Dicha componente es accedida por otros módulos del sistema para la construcción de vistas y otras operaciones.

```
void f (int a, int b)
  {
    INSPECTOR_ENTRADA("f");
    ........
    INSPECTOR_SALIDA("f");
    return;
  }
```

(a)

```
push(pila,N)
for(i=0;i<TAM;i++)
 {
  /* acciones del loop */
  .....
  decrementarTope(pila);
 }
pop(pila);
```

(b)

Figura 2: Instrumentación de Funciones y Control de Ciclos

5. ADMINISTRACIÓN DE LA INFORMACIÓN

Esta componente almacena información de: tipos, módulos, variables, funciones y relaciones existentes entre ellas. Entre las relaciones más destacadas encontramos: llamadas a funciones, comunicación y dependencia de módulos, dependencia de tipos, definiciones de variables clasificadas por funciones y módulos, etc.
Por otra parte, esta componente administra la información dinámica del sistema. Como por ejemplo las funciones y tipos usados en tiempo de ejecución.
Tanto la información estática como la dinámica son administradas usando un modelo relacional de base de datos. La parte más compleja de este módulo radica en la administración de la información dinámica. Esta complejidad se debe a la enorme cantidad de información recuperada por las técnicas de extracción de información dinámica. Este problema ha conducido al uso de organizaciones de archivos sofisticadas para la implementación de esta componente.

6. VISTAS

Una vista es una perspectiva del sistema de estudio que permite resaltar algún aspecto del mismo. Las vistas son importantes porque actúan como facilitadores del aprendizaje y por lo tanto ayudan en la tarea de CP. En las subsecciones siguientes se describen los objetos de interés para visualizar e inspecionar [11], las vistas y sus funcionalidades.

6.1. OBJETOS VISUALIZADOS POR PICS

Entre los objetos que pueden ser visualizados por PICS se pueden mencionar:

Código Fuente y Objeto: los códigos fuente y objeto son las unidades básicas de inspección. La generación estrategias que faciliten la lectura y navegación en ambos códigos facilita la comprensión e inspección.

Datos: los datos son una fuente importante de información. A través del análisis de los datos el usuario puede:

- extraer el mapa de tipos (un grafo que muestra los tipos y las relaciones entre ellos).
- Relacionar las funciones con los datos. Esta operación es muy compleja cuando el sistema de estudio esta implementado usando el paradigma imperativo.
- Visualizar la tabla de símbolos del sistema.
- etc.

Grafo de Llamadas a Funciones: permite ver, en forma estática, como las funciones del sistema están interconectadas. Esta perspectiva es tradicional en CP y es muy útil para detectar, mediante operadores de grafos, funciones que son puntos de entrada o salida de un sistema, funciones escenciales etc. También es muy adecuado para usarlo en combinación con técnicas de análisis dinámico.

Funciones de Tiempo de Ejecución: consta de las funciones recuperadas por la técnica de instrumentación de código descripta en la sección 4. Conocer las funciones usadas por el sistema para alcanzar un objetivo simplifica el proceso de inspección y comprensión.

Grafo de Comunicaciones de Módulos: posibilita visualizar, en forma estática, como los módulos del sistema están interconectados. Esta perspectiva, al igual que el grafo de llamadas a funciones, es tradicional en CP. Sobre ella se pueden aplicar operadores de grafo para detectar módulos con características particulares.

Salida del Sistema: útil para entender las funcionalidades del sistema. El mecnanismo de inspección dinámica implementado en PICS posibilita relacionar la salida del sistema con las componentes utilizadas por el programa. De esta manera se logra una visión integrada entre el comportamiento y la operación del programa.

6.2. AMBIENTES DE VISUALIZACIÓN Y FUNCIONALIDADES DE PICS

En esta subsección se presentan las interfaces que implementan distintas vistas útiles para la comprensión e inspección de programas. Estas vistas posibilitan visualizar las componentes descriptas en la sección 6.1. Además se explica la forma de navegación y las funcionalidades provistas por cada una de ellas.

6.2.1. Interfaz de preprocesamiento e instrumentación de código

PICS posee un ambiente para visualizar el código fuente, preprocesarlo, aplicar el analizador sintáctico y visualizar el código instrumentado. Esta interfaz puede ser vista en la Figura 3. La interfaz esta compuesta de dos ventanas. En la ventana superior, el usuario puede visualizar el código original, preprocesado o instrumentado. En la ventana inferior se pueden ver los resultados de cada una de las operaciones. La visualización del código original puede llevarse a cabo con una simple operación abrir asociada al primer botón de la barra de herramientas. El preprocesamiento se lleva a cabo presionando el segundo botón. El resultado de esta operación se muestra automáticamente en la ventana superior. Finalmente, luego del preprocesamiento, el código se puede instrumentar presionando

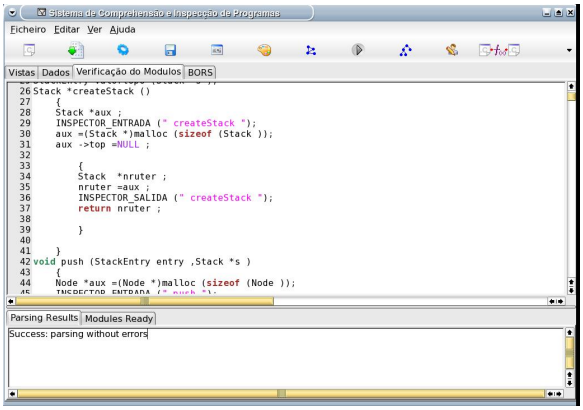

Figura 3: Visualizador de Código Original, Preprocesado o Instrumentado

el tercer botón del ratón. En la figura 3, el lector puede ver el código de un sistema instrumentado con las funciones de inspección.

6.2.2. Visualización de la tabla de símbolos

La figura 4 muestra la interfaz para visualizar la tabla de símbolos provista por PICS. Esta interfaz

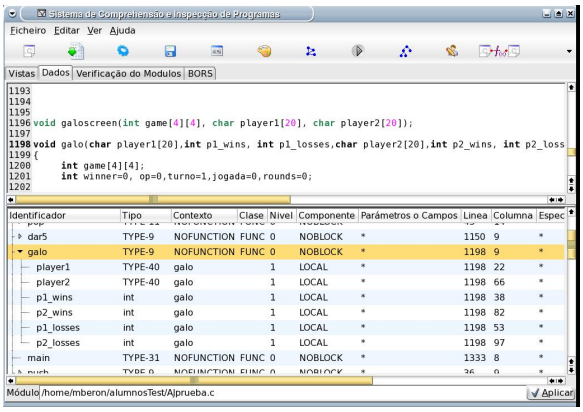

Figura 4: Visualizador de la Tabla de Símbolos

consta de dos ventanas. La superior contiene el código fuente bajo análisis. En la inferior se puede observar cada uno de los identificadores definidos en el sistema. Estos identificadores son clasificados en *Tipos Anónimos*, *Primitivos*, *Variables*, *Funciones* y *Estructuras*. Los primeros describen los tipos anónimos generados por el analizador sintáctico. Los segundos son los tipos predefinidos por el lenguaje usado para implementar el sistema, en este caso C. Los terceros referencian a las variables. Finalmente, los últimos identifican a las funciones del sistema y a sus parámetros como asi también a las estructuras y sus campos.

El mecanismo de navegación tiene dos modos de operación: *automático* y *búsqueda*. En el modo *automático*, el usuario sólo necesita posicionar el puntero del ratón sobre el identificador deseado. El sistema muestra, en forma automática en la ventana superior, el lugar donde aparece ese identificador en el código fuente. El modo *búsqueda* se utiliza cuando el usuario desea buscar un identificador. En este caso es necesario estar posicionado en la ventana inferior y luego escribir las letras que componen al identificador deseado. En ese momento se activa una búsqueda fonética que va seleccionando el identificador que contiene las letras escritas. Luego de realizada esta tarea solamente es necesario

presionar la tecla *enter* para que el sistema muestre el identificador en la ventana superior.

6.2.3. Visualización de tipos de datos abstractos

La figura 5 muestra el visualizador de *Tipos de Datos Abstractos* (TDAS) detectados por una heurística de detección de tipos incorporada en PICS. Este visualizador consta de un comando y

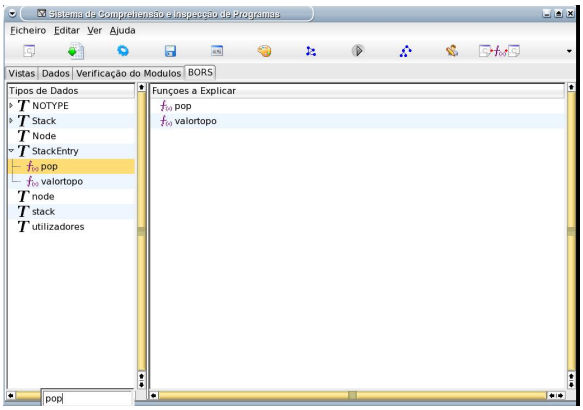

Figura 5: Visualizador de TDAS

dos ventanas principales. El comando esta disponible en la barra de herramientas. Basicamente este comando permite cargar, en la ventana principal izquiera, el archivo que contiene los TDAs del sistema. Esta ventana organiza la información en forma de árbol con tres niveles. La raíz (nivel 0) vincula a todos los nombres de los TDAS (nivel 1). Cada nombre de TDA contiene las funciones asociadas por la heurística de dectección de TDAS (nivel 2).
Cada nodo del árbol de tipos tiene asociado una operación de inserción. Esta operación permite colocar funciones de la ventana de la izquierda en la ventana de la derecha. Esta última ventana contiene las funciones que el usuario desea explicar y que sirven como entrada a la estrategia de relación comportamental-operacional denominada BORS. Los elementos incorporados en la ventana principal derecha tienen una función asociada que posibilita realizar eliminaciones. Esto tiene como objetivo permitirle al usuario personalizar las funciones que desea explicar. Finalmente, ambas ventanas poseen funciones de búsqueda fonética para facilitar la interacción.

6.2.4. Visualizaciones basadas en grafos

Para la visualización de grafos (de funciones, módulos o tipos) PICS provee un visualizador con las siguientes funcionalidades:

- Visualización del grafo correspondiente.
- Operaciones de zoom y movimiento del grafo.
- Cuando se dispone de información de las funciones usadas en tiempo de ejecución permite realizar animaciones a nivel funciones y módulos. Esas animaciones consisten en resaltar, con un color determidado, las funciones o módulos usados con un intérvalo de tiempo de demora.
- Animación paso a paso usando la información dinámica.
- Inserción y eliminación de nodos y arcos

La figura 6 muestra el grafo de llamadas a funciones para una aplicación. Como se puede observar, el ambiente tiene un sector de visualización donde se dibuja el grafo recuperado. Luego en la parte

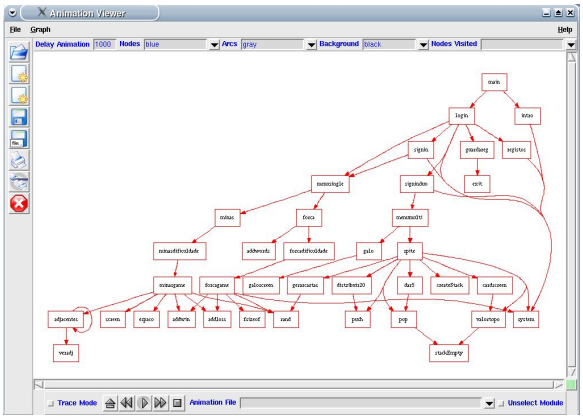

Figura 6: Ambiente de visualización de grafos

superior se encuentran diferentes entradas que perminten personalizar el ambiente. Estas entradas posibilitan cambiar de color a los nodos y a los arcos y establecer la demora para el modo animación. En la parte inferior se encuentra un sector de comandos que posibilita la ejecución de la animación o realizar dicha tarea paso a paso. También en la parte inferior, se localiza una entrada que indica el archivo que contiene las funciones de tiempo de ejecución para realizar la animación. Con este ambiente es posible navegar entre el grafo de módulos, el grafo de funciones y el código fuente. Estas operaciones pueden ser realizadas facilmente haciendo click con el ratón en el objeto deseado.

6.2.5. Inspección de funciones de tiempo de ejecución

La Figura 7 muestra la interfaz de navegación entre: las funciones usadas en tiempo de ejecución, el código fuente y objeto.

En la ventana de la izquierda, el usuario puede observar las funciones de tiempo de ejecución. En

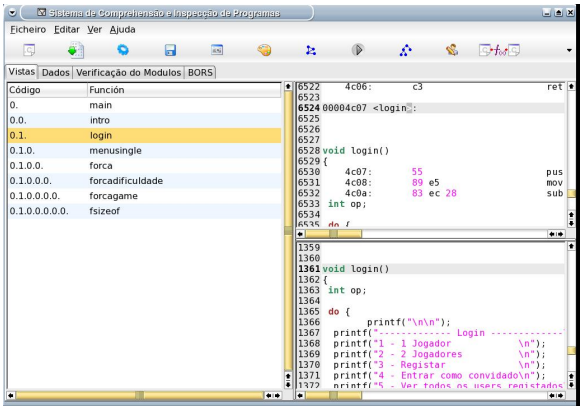

Figura 7: Interfaz de Navegación entre funciones de tiempo ejecucion, código fuente y objeto

las otras dos se pueden ver el código objeto (a la derecha en la parte superior) y el código fuente (a la derecha en la parte inferior). La navegación se produce haciendo click en la función de tiempo de ejecución deseada. El sistema, usando el repositorio de información, detecta el módulo fuente y objeto correspondiente. Luego de esa tarea, desensambla el módulo objeto para obtener el código assembly y bytecode correspondiente. Finalmente, los archivos resultantes son visualizados, resaltando la función correspondiente, en las ventanas de interfaz de PICS.

6.2.6. Visualización de la relación entre el dominio del problema y el dominio del programa

Con PICS es posible construir una vista que relaciona la operación del programa con su resultado. La Figura 8 muestra esta vista. El lector puede observar tres ventanas. En la ventana del fondo

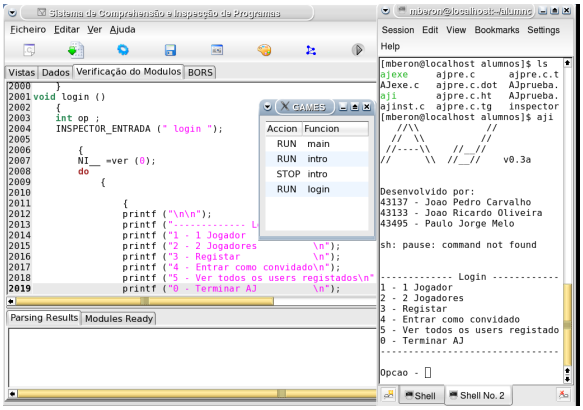

Figura 8: Relación Operacional-Comportamental

se encuentra la interfaz de visualización de código. También es posible ver: la ventana de funciones ejecutadas (recuperadas por el esquema de instrumentación de código) y la que contiene al sistema ejecutando. El lector puede observar la relación directa entre el funcionamiento del sistema (dominio del problema) y las funciones (dominio del programa). De esta manera, el usuario de PICS puede encontrar *significado y sentido* a las operaciones del programa. También esta visualización brinda la información necesaria para comenzar a inspeccionar el código estudiando solamente las funciones utilizadas por el sistema. La inspección se lleva a cabo utilizando las interfaces de PICS descriptas en las secciones anteriores.

7. BORS

La estrategia de relación comportamental-operacional, denifinida por nuestro grupo de investigación, utiliza infromación dinámica y estática del sistema de estudio. Además se basa en la siguiente observación:

"La salida de un sistema esta compuesta de objetos del dominio del problema. Usualmente estos objetos son implementados por tipos de datos abstractos, en el caso de lenguajes imperativos, o por clases, en el caso de lenguajes orientados a objetos. Tanto los TDAs como las Clases tienen objetos de dato que almacenan su estado y un conjunto de operaciones que los manipulan. Entonces es posible describir cada objeto del dominio del problema utilizando los TDAs o clases que los implementan."

Esta estrategia denominada BORS (Behavioral-Operational Relation Strategy) aplica los siguientes pasos para alcanzar su objetivo.

Detectar las funciones relacionadas con cada objeto del dominio del problema: esta tarea es llevada a cabo en forma semi-automática. El usuario selecciona los TDAs que desea explicar utilizando la interfaz de TDAs. Entonces todas las funciones relacionadas con esos tipos son cadidatas a ser explicadas y son almacenadas en una lista.

Construir un árbol de ejecución de funciones usadas en tiempo de ejecución: la salida del esquema de instrumentación de código contiene la información suficiente (inicio y fin de la

ejecución de las funciones) para construir una estructura de datos denominada fe-tree [1] [14](**f**unction **e**xecution **t**ree) que describe como las funciones del sistema de estudio son o fueron ejecutadas.

Explicar las funciones encontradas en el paso 1 usando el árbol construido en el paso 2: combinando la información de los dos pasos anteriores (lista de funciones a explicar y fe-tree) se pueden explicar las funciones mostrando: el contexto donde fueron invocadas (camino desde la raíz hasta la función) y que tarea realizaron para el sistema (sub-árbol cuya raíz es la función que se está explicando). Este proceso se lleva a cabo aplicando un recorrido por niveles sobre el fe-tree. Cada vez que se encuentra un nodo del fe-tree que pertenece a la lista de funciones a explicar se reporta el camino desde ese nodo a la raíz. También es posible mostar el sub-árbol que tiene como raíz el nodo analizado.

La figura 9 muestra el procedimiento empleado por esta estrategia. El lector interesado puede encon-

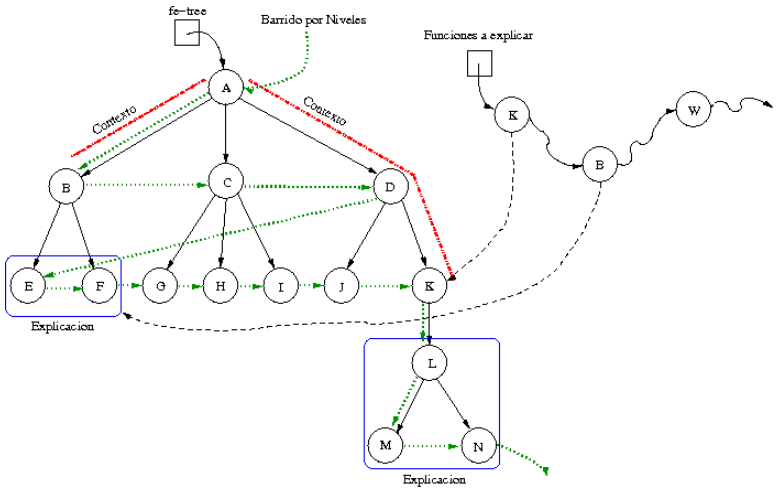

Figura 9: Estructuras de datos, información y procedimeinto utilizado por BORS

trar en [14] una explicación detallada de la estrategia BORS.

8. CONCLUSIÓN

En este artículo se presentó PICS una herramienta para la inspección y comprensión de programas escritos en lenguaje C. Una de las mayores fortalezas de este sistema consiste en la presentación de diferentes vistas útiles del sistema de estudio. Destacándose, en ese contexto, la relación comportamental y operacional obtenida a través de la implementación de la instrumentación de código y de la estrategia BORS. Esta relación puede ser claramente visualizada debido a la posibilidad de ejecución en paralelo del sistema bajo estudio y del sistema de inspeción de código.
La estrategia BORS combina información estática y dinámica con el fin de explicar los objetos del dominio del problema usando elementos del dominio del programa. Para llevar a cabo esta tarea es necesario que el usuario indique que tipos o funciones desea explicar. El resto de las actividades se realizan en forma automática.
PICS es el resultado de muchos esfuerzos de investigación e implementación en el contexto de: *modelos cognitivos*, *visualización de software* y *procesamiento de lenguajes*. PICS ha sido aplicado con éxito a diferentes programas cuyos tamaños varian entre las 0.3 kloc y 5 kloc de código C. El resultado de algunas de esas experiencias pueden ser vistos en [14] [12]. Sin embargo, en los últimos tiempos se realizaron otras pruebas interesantes del sistema. Por razones de extensión, y por desear presentar en

forma completa el sistema de inspección y comprensión, no fue posible describir estos experimentos en este artículo.
Como trabajo futuro se proyecta incorporar técnicas de generación de documentación de alto nivel desarrolladas por el grupo de investigación PCViA (**P**rogram **C**omprehension by **Vi**sual **I**nspection and **A**nimation) de la Universidad de Minho. Además se desea profundizar en técnicas de extracción de la información y visualización de programas. El principal objetivo es producir visualizaciones de alto nivel para lograr otras formas de relacionar el dominio del problema y el dominio del programa.

Referencias

[1] Abdelwahab Hamou-Lhadj.*The Concept of Trace Summarization.*PCODA: Program Comprehension through Dynamic Analysis.(2005), 38-42.

[2] Andy Zaidman, Bram Adams, and Kris Schutter. *Applying Dynamic Analysis in a Legacy Context: An Industrial Experience.* PCODA: Program Comprehension through Dynamic Analysis (2005),6-10.

[3] Franoise Balmas, Harald Werts, Rim Chaabane. *DDGraph: a Tool to Visualize Dynamic Dependences.*Program Comprehension through Dynamic Analysis (2005), 22-27.

[4] `http://wiki.di.uminho.pt/twiki/bin/view/Research/PCVIA`

[5] `http://www.scitools.com/products/understand`

[6] `http://www.grammatech.com`

[7] `http://www.imagix.com`

[8] `http://cscope.sourceforge.net`

[9] `http://www.jgrasp.org`

[10] `http://www.thechiselgroup.org/shrimp`

[11] Maria J. Pereira,*Concepção, Especificação de uma Linguagem Visual*, Ph.D. thesis, Universidade do Minho, Braga, 1996.

[12] Mario M. Berón, Pedro Henriques, Maria J. Varanda, Roberto Uzal, Germán Montejano. *Language Processing Tool for Program Comprehension.* XII Argentine Congress on Computer Science(2006).

[13] Mario M. Berón, Pedro Henriques, Maria J. Varanda, Roberto Uzal. *Herramientas para la compresion de programas.* VIII Workshop de Investigadores en Ciencias de la Computación (2006).

[14] Mario M. Berón, Pedro R. Henriques, Maria J. Varanda Pereira, Roberto Uzal. *Static and Dynamic Strategies to Understand C Programs by Code Annotation.* European Join Conference on Theory and Practice of Software. Braga-Portugal. 2007.

[15] Wang Yuying, Li Qingshan, Chen Ping, Ren Chunde.*Dynamic Fan-in and Fan-out Metrics for Program Comprehension.* PCODA: Program Comprehension through Dynamic Analysis (2005), 38-42.