Representación visual de la ejecución de una arquitectura de software basada en componentes con especificación formal en cálculo ρ_{arq}

Maestría en Ciencias de la Información y las comunicaciones
Universidad Distrital Francisco José de Caldas

UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS

Agenda

- Tema objeto de estudio
- Planteamiento del problema
- Diapositiva resumen
- Objetivos
- Justificación
- Antecedentes
- Hipótesis
- Alcances y Limitaciones
- Metodología
- Resultados
- Conclusiones
- Aportaciones
- Trabajo Futuro

Tema objeto de estudio

- **Representación visual de arquitecturas de software descritas con Cálculo ρ_{arq}.**

Los lenguajes de descripción arquitectural basados en álgebras de proceso requieren de un gran conocimiento técnico y de un esfuerzo alto para realizar labores de análisis.

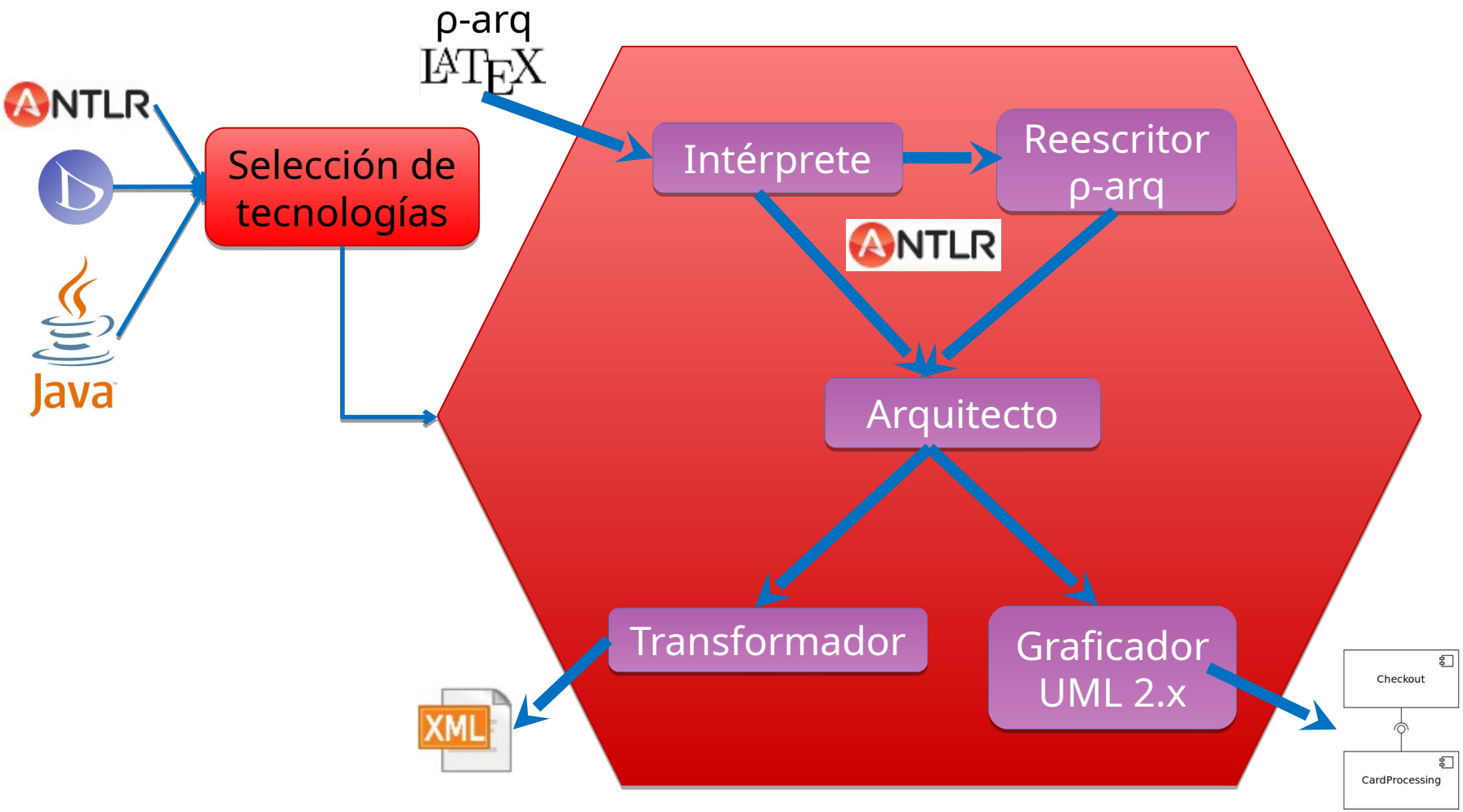
El tema que atañe a este proyecto es la simplificación de dichas tareas de análisis realizando la automatización de la semántica operacional del cálculo $\boldsymbol{\rho}_{\mathbf{arq}}$ en un sistema basado en software que muestre gráficamente y empleando la notación gráfica de UML 2.x el flujo de ejecución de arquitecturas descritas usando dicha álgebra de proceso.

PLANTEAMIENTO DEL PROBLEMA

En la actualidad existen muchos lenguajes que permiten la labor de descripción de arquitecturas de software, sin embargo solo aquellos que manejan un nivel de formalidad facilitan el análisis de propiedades y en particular el modelado de la ejecución de las arquitecturas descritas.

El cálculo ρ_{arq} fue concebido para aplicar las álgebras de proceso al nivel de especificación de arquitecturas de referencia prescriptiva y descriptiva[1], pero hasta el momento dicha labor debe realizarse de forma manual por lo cual el esfuerzo requerido para manejar especificaciones arquitecturales es grande e impide su adopción en el proceso de diseño de arquitecturas.

Diapositiva resumen



Objetivos

- Objetivo General
  - Construir un sistema basado en software que tenga la capacidad de interpretar las expresiones del cálculo ρ-arq que describen una arquitectura de software basada en componentes con el objeto de visualizar el flujo de ejecución de los módulos de dicha arquitectura de manera gráfica.
- Objetivos específicos
  - Utilizar las herramientas del cálculo ρ_{arq} para la descripción de arquitecturas de software de tal forma que el sistema pueda interpretar la estructura del sistema definido.
  - Emplear la notación gráfica provista por UML 2.x para el despliegue de los elementos definidos para el sistema de software diseñado.
  - Emplear para la persistencia de los modelos en UML 2.x el lenguaje de marcado extensible XML.
  - Determinar las tecnologías para la interpretación y transformación de expresiones que soporten la traducción de configuraciones arquitecturales expresadas en el cálculo ρ_{arq} a notaciones visuales de componentes de software conforme a UML 2.x.
  - Construir una interfaz gráfica que permita visualizar el flujo de ejecución de una arquitectura expresada como componentes de software alambrados con conectores de ensamble.

JUSTIFICACIÓN

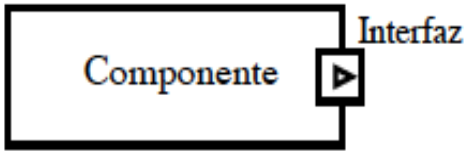
En la actualidad solo algunos LDA tienen herramientas gráficas que facilitan la labor de análisis sobre arquitecturas de software. Es el caso de Rapide con su herramienta POV[3] (Partial Order Viewer) que despliega gráficamente los eventos producidos en la ejecución de una arquitectura. Algunas se apoyan en las características gráficas de otras herramientas como es el caso del proyecto Pi-ADL[4] que emplea las capacidades gráficas de UML y define un perfil que permite la representación de los elementos estructurales propuestos en el lenguaje. Sin embargo, la necesidad de herramientas que faciliten, a los investigadores y en general a los arquitectos de software, la labor de análisis es la que soporta la naturaleza de este proyecto.

Antecedentes

- Rapide
  - Es un lenguaje empleado para definir y ejecutar modelos de arquitecturas de software. Utiliza para su definición el concepto de evento y las relaciones de temporización y causalidad entre dichos eventos. Esto lo habilita para el análisis de modelos de sistemas distribuidos y concurrentes ya que provee la historia causal de un conjunto de eventos del sistema llamados POSETS (Conjunto de eventos parcialmente ordenados).
- Darwin


```
Component  Filtro{
    provide   Salida<stream char>;
    require   Entrada<stream char>;
}
```

- Koala



Antecedentes

- Cálculo ρ_{arq}
  - El cálculo ρ_{arq} es una extensión del cálculo ρ [5] para su aplicación en la especificación de arquitecturas de software. La similaridad sintáctica y de la semántica operacional entre el cálculo ρ_{arq} y el cálculo π posibilita utilizar este con ligeras adaptaciones sintácticas a los propósitos de especificación de arquitecturas de software.

SÍMBOLOS	
x, y, z	variables
a, b, c	nombres
$u, v, w ::= x \mid a$	referencias

EXPRESIONES		
$E, F, G ::=$	$\top$	**Null**
$\mid$	$E \wedge F$	**Composición**
$\mid$	$if(C_1 \cdots C_n)\ else\ G$	**Combinador de selección condicionada**
$\mid$	$x :: \overline{y}/E$	**Abstracción**
$\mid$	$x\overline{y}/E$	**Aplicación**
$\mid$	τ/E	**Reacción interna**
$\mid$	$\exists wE$	**Declaración**
$\mid$	$x : \overline{y}/E$	**Replicación**
$\mid$	$E^{\top}$	**Ejecución exitosa del componente E**
$\mid$	$E^{\perp}$	**Ejecución no exitosa del componente E**
$\mid$	$OSO(E)\ do\ F\ else\ G$	**On Success Of**
$\phi, \psi ::=$	$\dot{\top}$	**Verdad Lógica**
$\mid$	$\perp$	**Falsedad Lógica**
$\mid$	$x = y$	**Restricción ecuacional**
$\mid$	$\phi \dot{\wedge} \psi$	**Conjunción de restricciones**
$\mid$	$\dot{\exists}\phi$	**Cuantificador existencial**

Antecedentes

- Reglas de reducción del cálculo ρ_{arq}

$(A_{\rho_{arq}})$ $\phi \wedge x : \overline{y}/E \wedge x'\overline{z}/F \longrightarrow \phi \wedge x : \overline{y}/E \wedge [\overline{z}/\overline{y}]E \wedge F$ $si\ \phi \models_\Delta x = x', \mathcal{V}(\overline{z}) \cap \mathcal{BV}(E) = \emptyset$

$(C_{\rho_{arq}})$ $\phi_1 \wedge \phi_2 \longrightarrow \psi$ $si\ \phi_1 \wedge \phi_2 \models\!\!\!\models_\Delta \psi$

$(Comb_{\rho_{arq}})$ $\phi \wedge if\ (C_1) \ldots (C_n)\ else\ F\ fi \longrightarrow \begin{cases} E_k, & si\ \phi \models_\Delta \psi_k \\ F, & si\ \phi \models_\Delta \neg\psi_k \end{cases} ; \forall k = 1, 2, \ldots, n$

$Con\ \ C_k ::= \exists\overline{x}(\psi_k\ Then\ E_k)\ ;\ k = 1, 2, \ldots, n$

$(Ejec_\tau)$

(a) $[OSO(E)\ do\ F\ else\ G] \wedge E^\top \longrightarrow F$, *debido a que hay ejecución exitosa del componente*

(b) $[OSO(E)\ do\ F\ else\ G] \wedge E^\perp \longrightarrow G$, *debido a que no hay ejecución exitosa del componente*

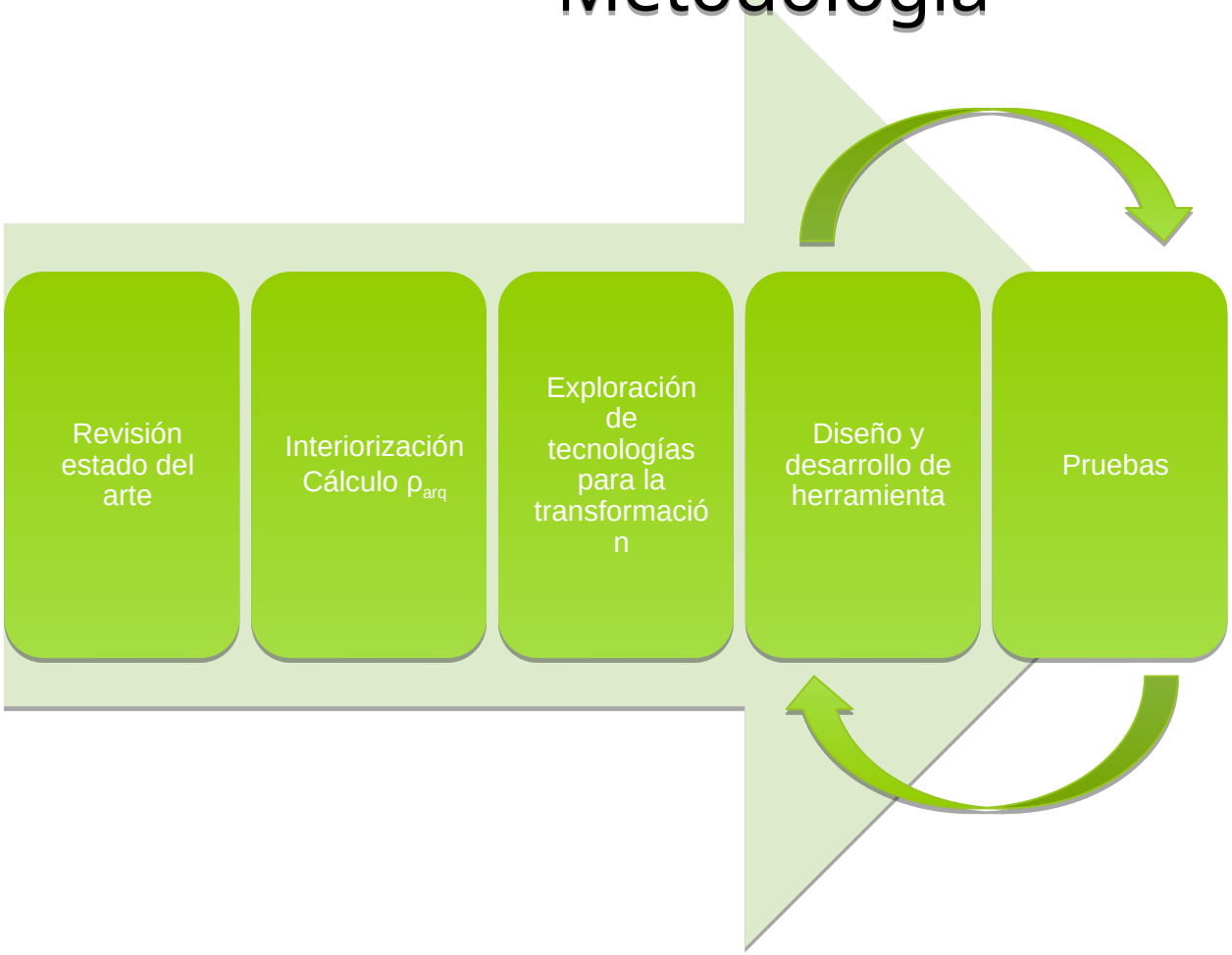
Hipótesis

- A partir de una descripción arquitectural con cálculo ρ_{arq}, de un sistema de software basado en componentes, es posible realizar la traducción de la representación de los elementos a la notación gráfica utilizada por UML 2.x; así mismo, los estados (Configuraciones arquitecturales en un momento dado) por los que fluye la arquitectura, resultado de la semántica operacional sobre las expresiones definidas en el cálculo, podrán ser representados gráficamente usando notación UML 2.x.

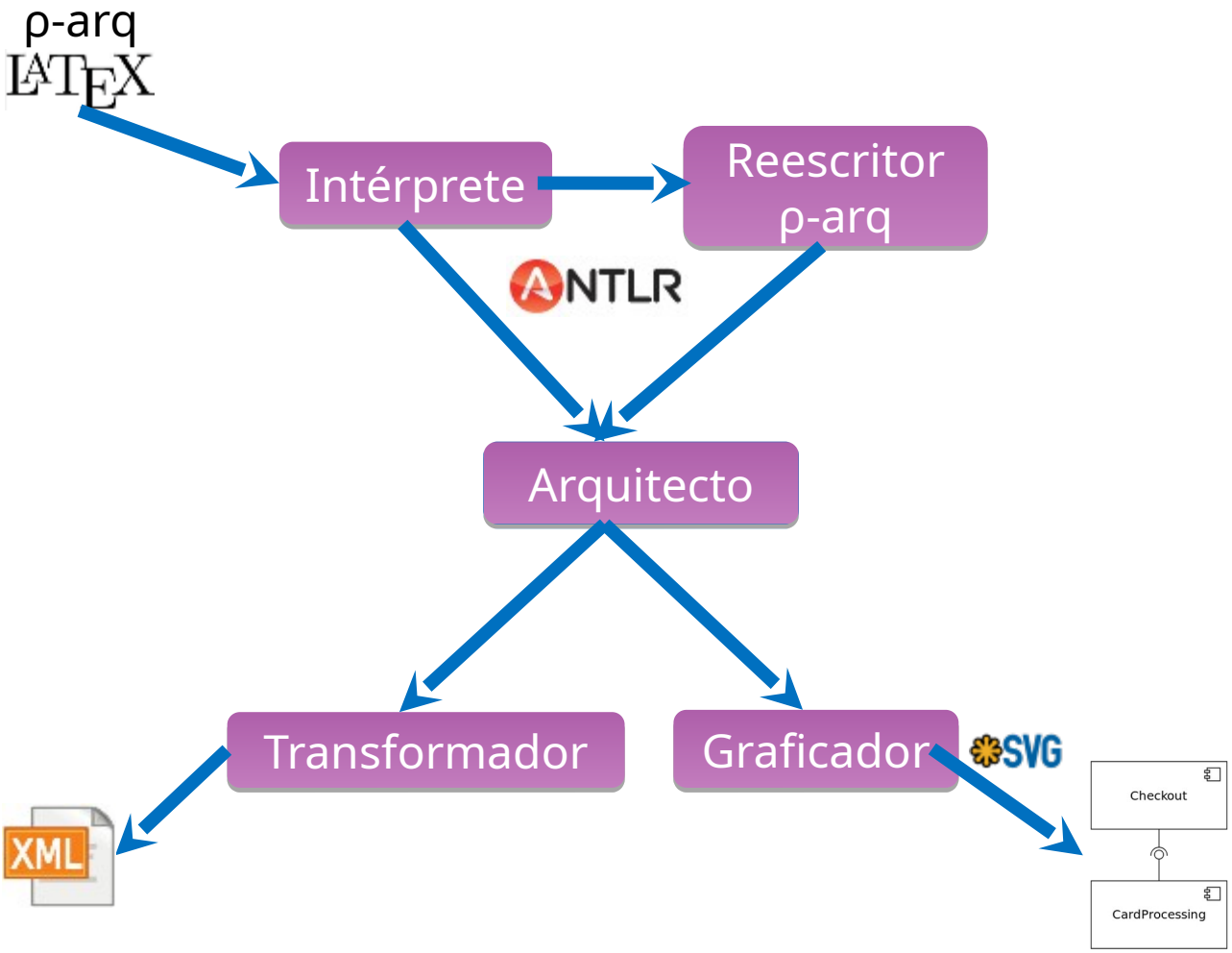
Alcances y Limitaciones

- El proyecto solo contempla la representación de arquitecturas usando UML y el cálculo ρ-arq.
- La composición jerárquica de arquitecturas no fue considerada en este proyecto.
- La herramienta no estará en capacidad de detectar comportamientos anormales del flujo de la arquitectura e informarlos al arquitecto, solamente mostrará el flujo de ejecución descrito en la arquitectura. El análisis del comportamiento corre por parte del Arquitecto.
- La herramienta no realiza validación ni verificación de modelos.

Metodología



Resultados


ρ-arq
LaTeX
Intérprete
Reescritor ρ-arq
ANTLR
Arquitecto
Transformador
Graficador
SVG
XML
Checkout
CardProcessing

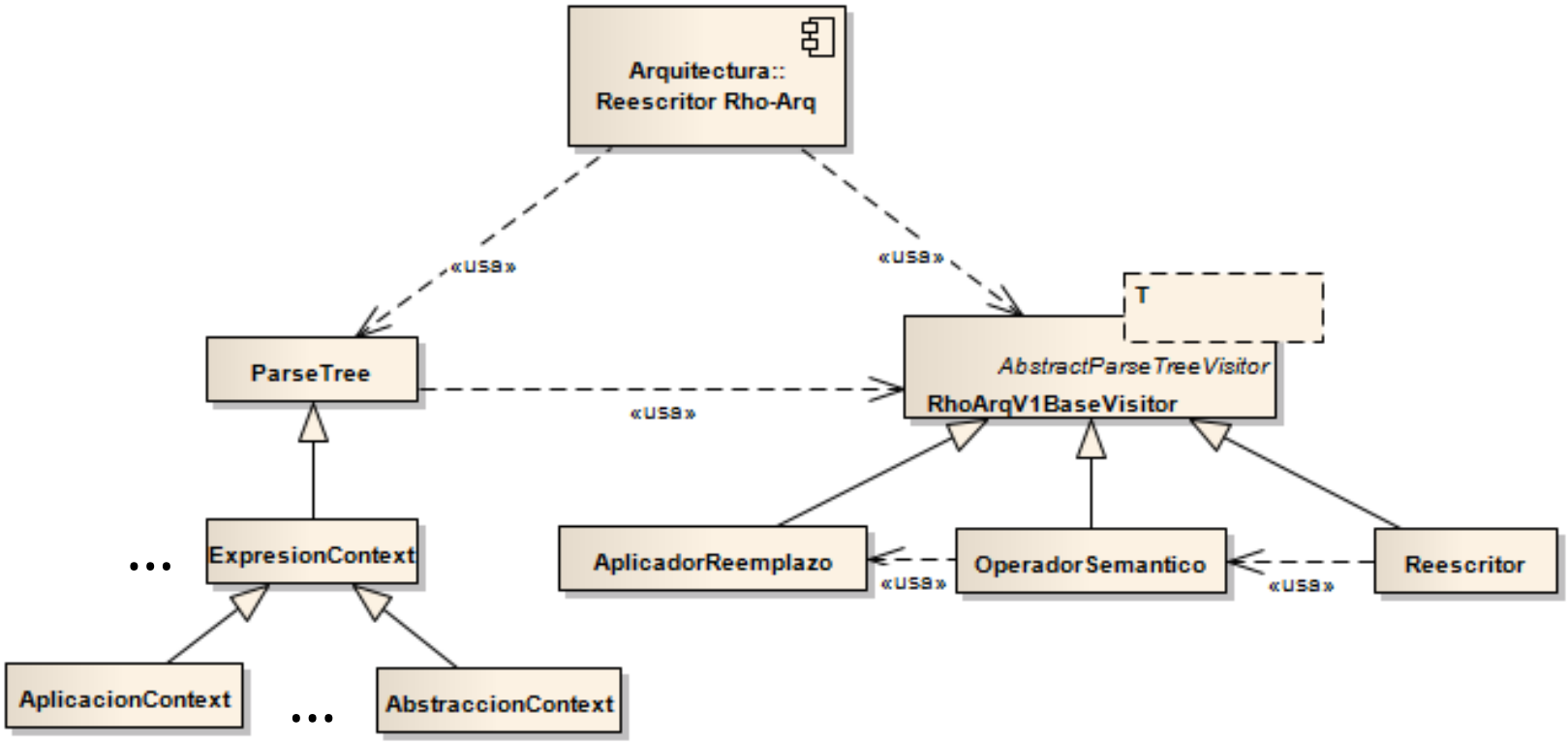
Resultados

- Incorporación de ANTLR

```
grammar RhoArqV1;

//Gramática
//Inicio de documento
documento :  ENCABEZADO (LATEX* ecuacion+ LATEX*)+ '\\end{document}';

//Identificador de ecuaciones latex
ecuacion : '\\begin{equation*}' definicion '\\end{equation*}'
         | '\\begin{equation}' definicion '\\end{equation}';

//Definiciones
definicion : ID '=' expresion;

//Expresion Rho-Arq
expresion: '(' expresion ')'                          #Grupo
               | '[' expresion ']'                    #Grupo2
               | expresion '\\wedge' expresion        #Composicion
               | '\\exists' ID expresion              #Declaracion
               | ID '::' ID '/' expresion             #Abstraccion
               | ID ':' ID '/' expresion              #Replicacion
               | aplicacionreducida '/'  expresion    #Aplicacion
               | '\\tau/' expresion                   #ReaccionInterna
               | interior                             #InteriorDeComponente
               | '[' ID '/' ID ']' expresion          #Reemplazo
               | ejecucionExitosa                     #EjecExitosa
               | ejecucionNoExitosa                   #EjecNoExitosa
               | aplicacionreducida                   #AppReducida
               | observacion                          #ExpObservacion
```

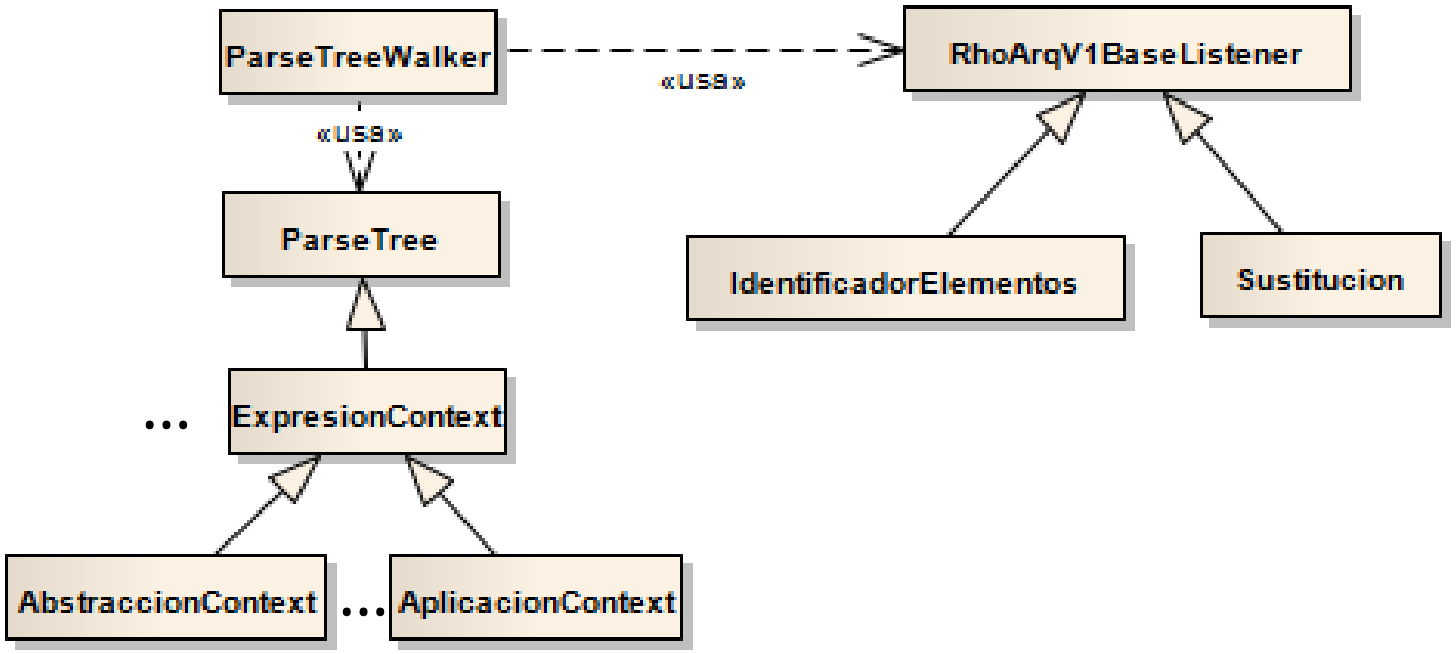
Resultados

- API ANTLR



Resultados

- API ANTLR



Resultados

```
\documentclass[10pt,letterpaper,fleqn]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\author{Alejandro Rico}
\title{Prueba Rho-arq}
\begin{document}
\begin{equation*}
    S=E \wedge F \wedge C_{FE}
\end{equation*}
\begin{equation*}
    REQ_E(r,l,i)=\exists l_E[(r_E::y/y\overline{l_E})\wedge(l_E::i_E/E^{(int)})]
\end{equation*}
\begin{equation*}
    PROV_E(p,s)=p_E:x/xs_E
\end{equation*}
\begin{equation*}
    E=REQ_E(r,l,i)\wedge PROV_E(p,s)
\end{equation*}
\begin{equation*}
    F=(p_F:z/z\overline{s_F})
\end{equation*}
\begin{equation*}
    C_{FE}=r_E\overline{p_F}
\end{equation*}
\begin{equation*}
    C_{EF}=r_F\overline{p_E}
\end{equation*}
\end{document}
```

$$S = E \wedge F \wedge C_{FE}$$

$$REQ_E(r,l,i) = \exists l_E[(r_E :: y/y\overline{l_E}) \wedge (l_E :: i_E/E^{(int)})]$$

$$PROV_E(p,s) = p_E : x/xs_E$$

$$E = REQ_E(r,l,i) \wedge PROV_E(p,s)$$

$$F = p_F : z/z\overline{s_F}$$

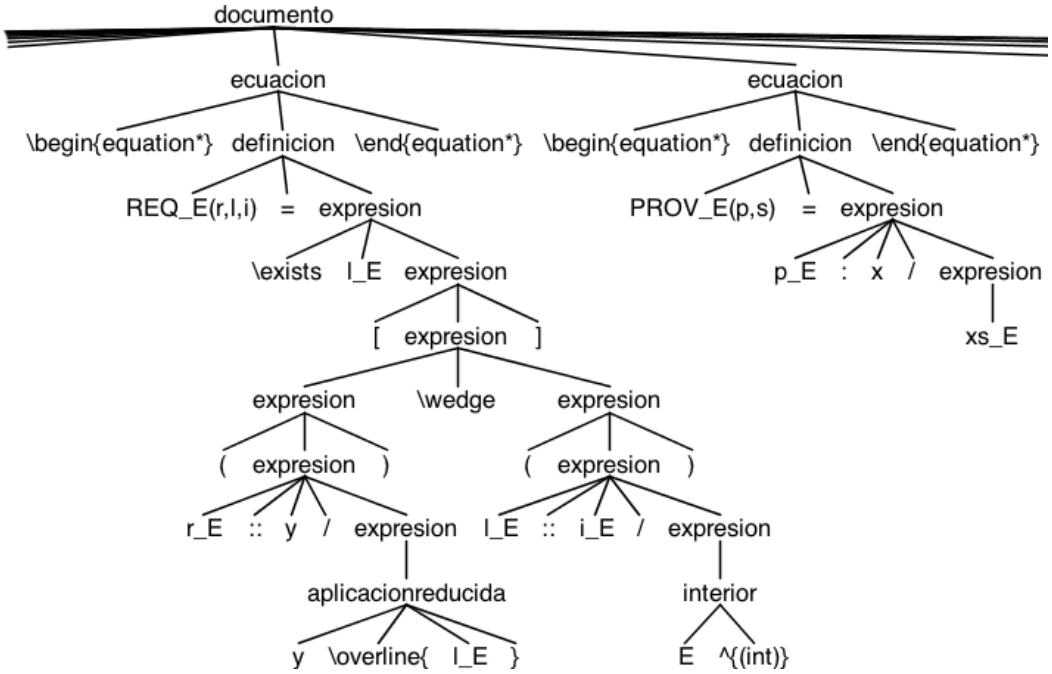
$$C_{FE} = r_E\overline{p_F}$$

Resultados

- Reescritura de expresiones

```
if(expresion instanceof AplicacionContext || expresion instanceof ReplicacionContext){
    ReemplazoContext Reemplazo = null;
    for(int Indice=IndiceExpresion+1;Indice<ExpresionesIzquierda.size();Indice++){
        if(ExpresionesIzquierda.get(Indice) instanceof ReplicacionContext && expresion
        instanceof AplicacionContext){
            ReplicacionContext Replicacion =
        (ReplicacionContext)ExpresionesIzquierda.get(Indice);
            AplicacionContext Aplicacion = (AplicacionContext)expresion;
            Reemplazo = this.reducirAlphaRhoArq(Replicacion, Aplicacion);
        }
        if(ExpresionesIzquierda.get(Indice) instanceof AplicacionContext && expresion
        instanceof ReplicacionContext){
            AplicacionContext Aplicacion = (AplicacionContext)ExpresionesIzquierda.get(Indice);
            ReplicacionContext Replicacion = (ReplicacionContext)expresion;
            Reemplazo = this.reducirAlphaRhoArq(Replicacion, Aplicacion);
        }
        if (Reemplazo != null){
            ArrayList<RhoArqV1Parser.ExpresionContext> Retorno = new
        ArrayList<RhoArqV1Parser.ExpresionContext>();
            Retorno.add(null);
            Retorno.add((ExpresionContext)Reemplazo);
            Retorno.add(expresion);
            Retorno.add(ExpresionesIzquierda.get(Indice));
            return Retorno;
        }
    }
    }
```

Resultados



$$S = \exists l_E[(r_E :: y/y\overline{l_E}) \wedge (l_E :: i_E/E^{(int)})] \wedge p_E : x/xs_E \wedge p_F : z/z\overline{s_F} \wedge r_E\overline{p_F}$$

$$S = \exists l_E[([p_F/y]y\overline{l_E}) \wedge (l_E :: i_E/E^{(int)})] \wedge p_E : x/xs_E \wedge p_F : z/z\overline{s_F}$$

$$S = \exists l_E[(l_E :: i_E/E^{(int)})] \wedge p_E : x/xs_E \wedge (p_F : z/z\overline{s_F}) \wedge [l_E/z]z\overline{s_F}$$

$$S = \exists l_E[([s_F/i_E]E^{(int)})] \wedge p_E : x/xs_E \wedge (p_F : z/z\overline{s_F})$$

$$S = E \wedge F \wedge C_{FE}$$

$$REQ_E(r, l, i) = \exists l_E[(r_E :: y/y\overline{l_E}) \wedge (l_E :: i_E/E^{(int)})]$$

$$PROV_E(p, s) = p_E : x/xs_E$$

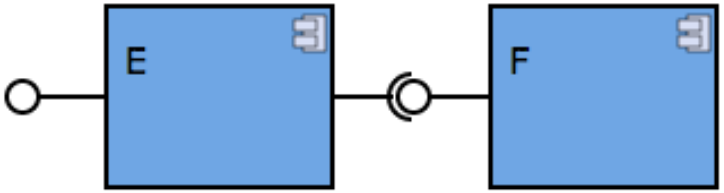
$$E = REQ_E(r, l, i) \wedge PROV_E(p, s)$$

$$F = (p_F : z/z\overline{s_F})$$

$$C_{FE} = r_E\overline{p_F}$$

$$C_{EF} = r_F\overline{p_E}$$

Resultados



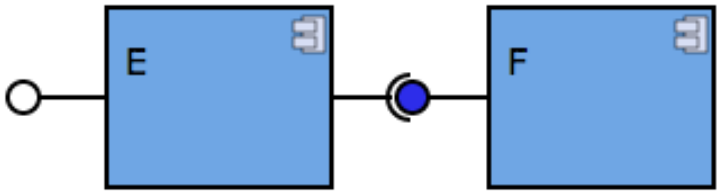
Resultados

$$S = \exists l_E[(r_E :: y/y\overline{l_E}) \wedge (l_E :: i_E/E^{(int)})] \wedge p_E : x/xs_E \wedge p_F : z/z\overline{s_F} \wedge r_E\overline{p_F}$$

$$S = \exists l_E[([p_F/y]y\overline{l_E}) \wedge (l_E :: i_E/E^{(int)})] \wedge p_E : x/xs_E \wedge p_F : z/z\overline{s_F}$$

$$S = \exists l_E[(l_E :: i_E/E^{(int)})] \wedge p_E : x/xs_E \wedge (p_F : z/z\overline{s_F}) \wedge [l_E/z]z\overline{s_F}$$

$$S = \exists l_E[([s_F/i_E]E^{(int)})] \wedge p_E : x/xs_E \wedge (p_F : z/z\overline{s_F})$$



Resultados

$$S = E \wedge F \wedge R \wedge C_{FE} \wedge C_{RE} \wedge C_{RF} \wedge r_F\overline{p1_E}$$

$$REQ_E(r,l,i) = \exists l_E[(r_E :: y/y\overline{l_E}) \wedge (l_E :: i_E/E^{(int)})]$$

$$PROV_E(p,s) = p_E : x/x\overline{s_E}$$

$$E = REQ_E(r,l,i) \wedge PROV_E(p,s) \wedge p1_E : x/x\overline{s1_E}$$

$$F = (p_1F : z/z\overline{s_1F}) \wedge (p_2F : z/z\overline{s_2F}) \wedge \exists l_F[(r_F :: y/y\overline{l_F}) \wedge (l_F :: i_F/F^{(int)})]$$

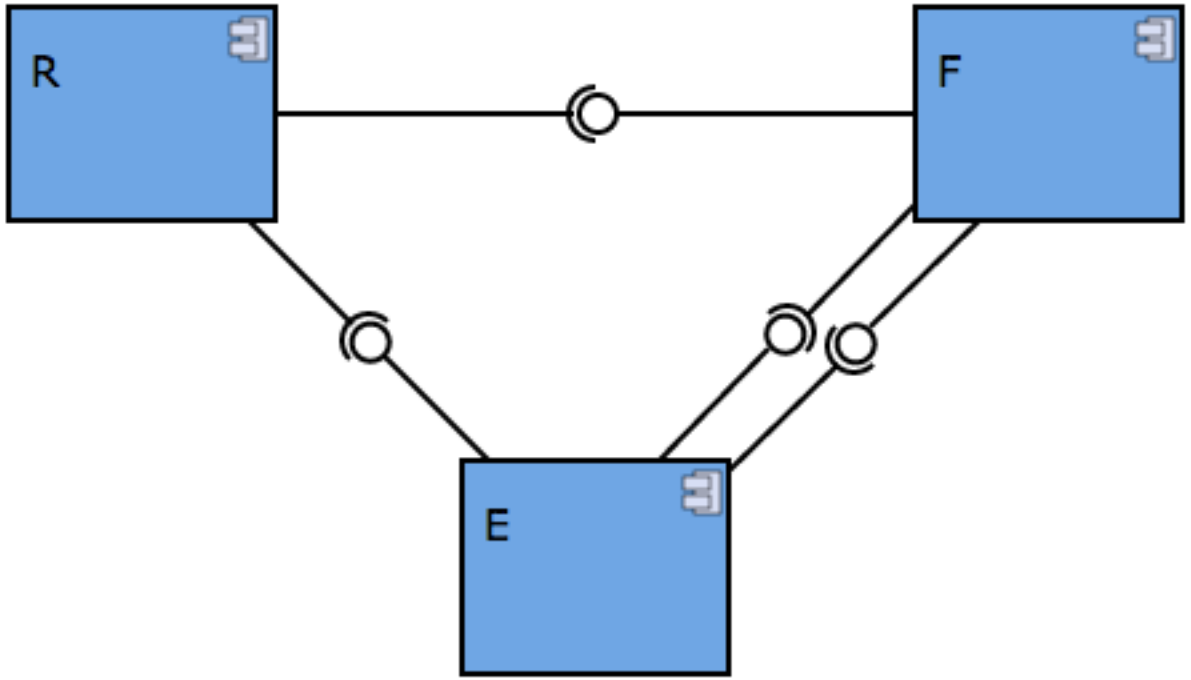
$$C_{FE} = r_E\overline{p_1F}$$

$$R = (\exists l_1R[(r_1R :: y/y\overline{l_1R}) \wedge (l_1R :: i_R/R^{(int)})]) \wedge \exists l_2R[(r_2R :: y/y\overline{l_2R}) \wedge (l_2R :: i_R/R^{(int)})]$$

$$C_{RE} = r_1R\overline{p_E}$$

$$C_{RF} = r_2R\overline{p_2F}$$

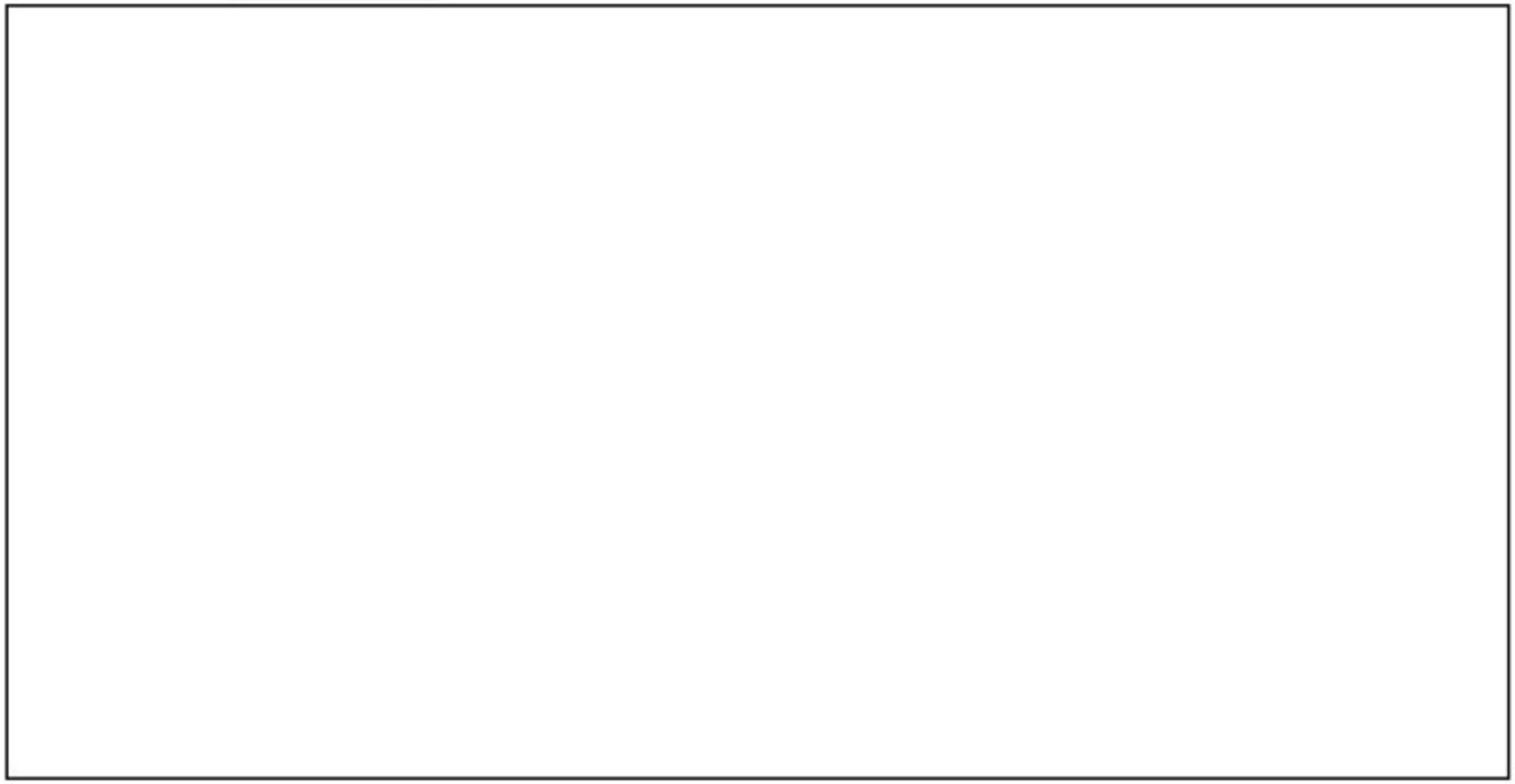
Resultados



PintArq

Seleccionar archivo Ningún archivo seleccionado

Cargar Arquitectura



Conclusiones

- Es posible identificar a partir de la descripción de una arquitectura en el cálculo ρ_{arq} los elementos estructurales de dicha arquitectura para representarlos usando la notación gráfica de UML.
- Para establecer la configuración de los elementos empleados en la descripción es necesario identificar las expresiones de conexión representadas bajo aplicaciones reducidas.
- La semántica operacional del cálculo ρ_{arq} puede automatizarse empleando búsqueda de patrones de expresión que permitan identificar las interacciones descritas por el cálculo.
- Es posible identificar las expresiones que cumplen con las reglas de reducción para generar interacciones y cambios de estado en una arquitectura.
- El despliegue gráfico del modelo de componentes en UML tiene algunas complicaciones ya que en algunos casos es imposible dibujar el grafo sin que haya intersección de las interfaces.

Aportes

- Algoritmos de identificación de patrones de expresión.
- Automatización de la semántica operacional del cálculo $\rho_{arq.}$
- Algoritmos de identificación para la configuración de arquitecturas, incluyendo composición jerárquica.
- Algoritmo de despliegue gráfico basado en ubicación polar.

Trabajo Futuro

- Construcción de una herramienta gráfica de edición de arquitecturas de software y escritura en cálculo ρ_{arq}.
- Despliegue gráfico de la composición jerárquica de arquitecturas.
- Implementación de herramientas de análisis para identificación de abrazos mortales.

Bibliografía

[1]. Henry Alberto Diosa. Especificación de un Modelo de Referencia Arquitectural de Software A Nivel de Configuraci´on, Estructura y Comportamiento. PhD thesis, Universidad del valle- Escuela de ingeniería de sistemas y computación, Febrero 2008.

[2]. Diosa, Henry Alberto, Diaz Frias Juan Francisco, Gaona Cuevas Juan Francisco. Cálculo para el modelado formal de arquitecturas de software basadas en componentes: Cálculo ρ-arq. Revista científica Universidad Distrital, 12:172-184, 2010.

[3]. David C. Luckham. Rapide: A language and toolset for simulation of distributed systems by partial orderings of events. Technical report, Stanford, CA, USA, 1996.

[4]. F. Oquendo, “π-ADL: an Architecture Description Language based on the higher-order typed π-calculus for specifying dynamic and mobile software architectures,” ACM SIGSOFT Softw. Eng. Notes, vol. 29, pp. 1–14, 2004.

[5]. H. Cirstea, “Calcul de réécriture: fondements et applications,” Université Henri Poincaré - Nancy I, 2000.

[6]. R. van Ommering, F. van der Linden, J. Kramer, and J. Magee. The koala component model for consumer electronics software. Computer, 33(3):78 –85, mar 2000.

Muchas gracias!!!