Informe Laboratorio 2: Paradigmas de programación

Rodrigo Mardones

29-06-2020

Indice

Introducción

Los paradigmas de programación son marcos de trabajo para, desde cierta perspectiva abordada según el problema a solucionar, trabajar con herramientas ya establecidas según las normas que lo rigen en el cotexto dado. Es preciso señalar que indiferente de los lenguajes nuevos, modernos y sus herramnientas propias asociadas, siempre se regirán por al menos uno o más paradigmas de programación. Pues son estos quienes nos permiten analizar de buena manera la mejor respuesta a un problema y su contexto especifico.

Objectivo General

Para este el caso de este laboratorio, se plantea la construcción de una pieza de software pequeña que permita emular un controlador de versiones tipo GIT, dentro de los parametros del paradigma de programacion lógica y sus atributos.

Objectivos Específicos

Para abordar el problema se plantean lo siguientes objectivos especificos

- Análisis del problema
- Aspectos de implementación
- construccion de implementacion
- Conclusiones

Este informe solo recopila el estudio y construccion del proyecto de laboratorio descrito, por lo que no se abordarán otros paradigmas que pueden mencionarse como referencia para la construcción del mismo en las páginas contiguas.

Descripción Del Problema

El desarrollo de software es un trabajo que no se escapa del desarrollo en qué equipo, una de los mayores problemas a enfrentar dentro de este punto en la construcción mantenible y ordenada de codigo fuente para su posterior uso en el entorno que se le necesite.

Los problemas más comunes que se desprenden de esta problematica son los siguientes:

- Cambios no supervisados
- errores en cambios no supervisados
- manejo de errores en versiones funcionales del proyecto
- manejo de nuevas funcionalidades en versiones futuras del proyecto
- registro de cambios realizados, quién los hizo , cuándo los hizo y qué hizo
- estaciones de trabajo independientes para no cruzar cambios de distintos colaboradores
- versionado de proyecto para su despligue

Es por lo anterior mencionado que nacen los “controladores de versiones”, software encargado de solucionar todos los estados del problema anterior mencionado, ayudando al al monitoreo, control y desarrollo del proyecto en cuestión más eficiente.

Descripción Del Paradigma

El “paradigma de programación lógico” es un paradigma de programación que como su nombre indica ocupa la lógica booleana para el planteamiento de problemas y su solución. Es decir toda declaración dentro de un programa sumido en este paradigma es valga la redundancia “declarativo” y de atribuido a un valor booleano (verdadero o falso). los que a su vez pueden ser declaraciones que dependas de otras.

Este paradigma requiere de una “base de conocimientos”, es decir, una serie de hechos que asumimos como verdades, y en base a estos tenemos consultas con respecto a los detalles particulares de cada hecho previamente establecido.

Mecanismos básicos

Bajo lo anterior dicho, es también importante destacar los mecanismos básicos que rigen este paradigma en cuestión, los cuales son:

- Unificación.
- Backtraking automático.
- Estructuras de datos basadas en árboles.

bajo lo anterior dicho, la estructura de un programa en este paradigma y en lenguaje empleado para el desarrollo de este laboratorio es el siguiente:

- Dominio : Descripcion breve de los tipos de datos empleados.
- Predicados : lista de todos los predicados(hechos) definidos dentro de un programa.
- Metas: consultas directas sobre la base de conocimientos previamente escrita.
- Clausulas:
- Reglas: Verdades absolutas dentro de la base de conocimientos.

Análisis del problema

Para el caso del paradigma lógico, definimos el problema con los siguientes hechos estrictos que debe tener nuestra representación interna de TDA:

- workspace : Espacio de trabajo donde quedan los archivos sin guardar en nuestro espacio de trabajo.
- Index : Espacio donde se guardan los archivos que se encuentran listos para ser guardados
- LocalRepository: rama de trabajo donde guardaremos nuestros commits de manera local.
- RemoteRepository: trabajo remoto ya guardado fuera de nuestro espacio de trabajo local.

La unidad básica dentro dentro de estos elementos es el commit, un cambios dentro de nuetro proyecto, el cual posee otro tipo de informacion inherente a nuestro proyecto, tal como el nombre de la persona que hizo el commit, el momento en el que lo hizo y los cambios asociados a este.

Otro concepto importante dentro del problema en sí es el uso de "ramas" espacios de trabajo independiente unos de otros y que pueden surgir a partir de ellos mismos. los cuales pueden tener commits distintos entre sí. Para el caso de este laboratorio solo haremos uso de una sola rama la cual será la definida al momento de crearse dentro del programa.

Aspectos de la implementación

Con respecto a los aspectos de la implementación de este laboratorio se definieron los siguientes:

NOMBRE	TIPO	DESCRIPCION
REPO	list	repositorio en cuestion
NOM	string	nombre del repositorio
AU	string	nombre del autor del repositorio
CRAT	string	fecha de creacion del repositorio
INDEX	list	lista de cambios(Index)
LOCAL	list	repositorio local(LocalRepository)
REM	list	repositorio remoto(remoteRepository)

Dentro del programa se definieron los siguientes hechos:

- gitInit(NombreRepo,Autor,RepoOut) : crea Repositorio a partir de base establecida, entregando nombre de repo y autor antes.
- gitAdd(RepoInput, Archivos, RepoOut): agrega archivos al Index para su posterior guardado.
- gitCommit(RepoInput, Mensaje, RepoOutput) : crea un commit con un mensaje y los suma a la rama trabajada.
- gitPush(RepoIn,RepoOut): envia los cambios del localRepository al RemoteRepository.
- git2String(RepoInput, RepoAsString) : muestra los elementos ordenados en un string.

Conclusiones

Sin duda el paradigma de programación lógica nos aporta una nueva mirada al momento de resolver problemas de distinto índole, no solo en la forma de construcción de los problemas sino también en el entendimiento de estos con las distintas caracteristicas asociadas al uso de este paradigma, como lo son el backtracking automático, la unificación y las construccion de estructuras basadas en arboles.

El paradigma de programación lógica nos permite especificar una base de conocimientos ya establecida, por lo que la solución de problemas se construye en base a lo que ya sabemos del problema persé, haciendo que este se vuelva mucho más entendible al trabajar con algo preestablecido.

Al compararlo con el paradigma visto en el laboratorio anterior, resulta mucho más sencillo trabajar en este paradigma, pues declarar hechos como casos base y en base a caracteristicas de caracter booleano, se hace mucho más semejante al algebra booleana.

Referencias

Para el desarrollo de este laboratorio se han consultado las siguientes fuentes bibliograficas:

- Castel de Haro, M., & Llorens Largo, F. (2005). Practicas de logica (1.a ed., Vol. 1). Universidad de Alicante.
- SWI-Prolog documentation. (s. f.). Documentacion Oficial de Prolog. https://www.swi-prolog.org/pldoc/index.html
- Clocksin, W. F. (2003). Programming in Prolog: Using The Iso Standard (5th ed.). Springer.