# PROGRAMAS PERSISTENTES E HIPERPROGRAMACIÓN

**Jalil Angulo Raquel Ivonné[1], Cortez Michel Fernando Erick[2]**
[1,2]Universidad Autónoma Juan Misael Saracho
Tarija - Bolivia

**Correo electrónico:** jalil.raquel@gmail.com, fecortez@uajms.edu.bo

## RESUMEN

Existen programas que se representan tradicionalmente como secuencias lineales de texto, estos programas se construyen y se almacenan en dispositivos de almacenamiento a largo plazo, tales como un sistema de ficheros, separados del entorno de ejecución que desaparece al final de cada "corrida" del programa. Por el contrario, en sistemas persistentes, los programas se pueden construir y almacenar en el mismo entorno en el que se ejecuten. Esto significa que los objetos a los cuales tiene acceso un programa pueden ya estar disponibles en el momento en que se desarrolle el programa. En este caso los enlaces (links) directos a los objetos se pueden incluir en el programa en vez de descripciones textuales. Un programa que contiene tanto el texto y los enlaces (links) a los objetos, es un hiperprograma.

Este artículo es el resultado de un estudio sobre los hiperprogramas y los lenguajes que los usan en el contexto actual de la tecnología de objetos como son: Napier 88 (ID:2267/napp002), HyperPascal (ID:5108/hyp005), PJama (ID:6912/), Aplicaciones de Hiperprogramación en la Web: HyperWeb, sus ventajas, desventajas y relaciones.

## PALABRAS CLAVE

Hiperprogramación, hiperprograma, persistencia, persistencia ortogonal.

## 1. HIPERPROGRAMACIÓN

En sistemas persistentes, los programas se pueden construir y almacenar en el mismo entorno de ejecución. Esto significa que los objetos a los cuales accede un programa pueden ya estar disponibles en el momento que el programa se desarrolle. En este caso los enlaces (links) directos a los objetos se pueden incluir en el programa en vez de descripciones textuales. Un programa que contiene el texto y enlaces (links) a los objetos es un hiperprograma [1].

Un ejemplo de un hiperprograma [2] se demuestra en la figura 1. Los enlaces incorporados en él son representados por el símbolo no-textual para permitir que se distingan del texto circundante. El primer enlace apunta a un valor del procedimiento de primera clase writeString el cual escribe un aviso al usuario. El programa entonces llama otro procedimiento readString para leer un nombre, y después encuentra una dirección que corresponda a ese nombre. Esta operación se logra llamando a un procedimiento lookup que busca la dirección dentro de una estructura de datos tipo tabla que se encuentra enlazada dentro del hiperprograma. Posteriormente, se escribe la dirección buscada. Observe que los objetos del código (readString, writeString y lookup): están denotados usando exactamente el mismo mecanismo que los objetos de datos (la tabla). Observe también que los nombres del objeto usados en esta descripción

se han asociado a los objetos, simplemente por claridad, y no son parte de la semántica del hiperprograma [4],[5],[6].

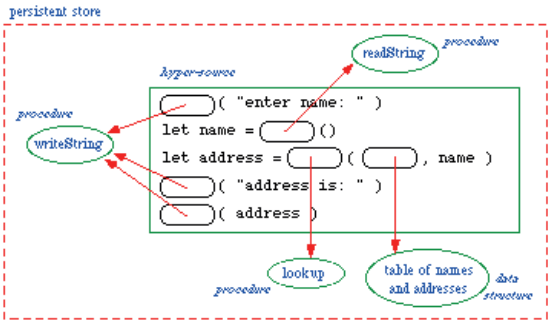


***Figura 1.*** *Ejemplo de un hiperprograma [2]*

## 2. LENGUAJES

El primer lenguaje que soporta hiperprogramación es **Napier88**[7] de tradición ALGOL al igual que su S-ALGOL de los precursores y Picosegundo-ALGOL [8].

En los sistemas de programación persistentes, como es Napier, el programador no precisa crear procedimientos específicos para guardar y recuperar las estructuras de datos que utiliza en sus programas, ya que este proceso se realiza de forma totalmente transparente por el propio sistema. Los sistemas de programación persistentes de la actualidad se basan ampliamente en los tres principios de ortogonalidad, que intentan asegurar que el usuario se abstraiga totalmente del hecho de utilizar objetos persistentes o no persistentes, del proceso de salvaguarda o recuperación de los mismos y de dónde residen esos objetos en cada momento.

Estos principios establecen que un sistema posee Persistencia Ortogonal (Atkinson & Morrison, 1995) si cumple las siguientes condiciones[9], [10]:

a) El trato del programa a objetos persistentes y no persistentes es indistinto. (Ortogonalidad al trato).

b) Cualquier objeto debe poder ser persistente, independientemente de su tipo o clase. (Ortogonalidad al tipo).

c) Todos los objetos relacionados con un objeto persistente deben ser persistentes, cumpliendo éstos a su vez con las condiciones a) y b). (Ortogonalidad de Identificación ó transitividad de identificación persistencia).

Los sistemas de programación persistentes que cumplen los criterios de ortogonalidad, como por ejemplo Napier (Morrison et al., 1999) o Pjama (Jordan & Atkinson, 1998), se basan en su mayor parte en una caché que actúa como intermediaria entre el mecanismo gestor de objetos y el almacenamiento persistente. De esta forma, cuando el programa precisa la utilización de un objeto, el gestor de objetos comprueba primero que no resida ya en la caché, antes de solicitar su traslado desde el almacenamiento persistente. El hecho de no encontrar el objeto buscado en la caché se denomina "fallo de objeto", y obliga a recuperar ese objeto del almacenamiento persistente [11], [12].

El sistema Napier88 se diseña como arquitectura acodada [Bro89] [13] que consiste en un recopilado [14,15,16,17], la máquina abstracta persistente (PAM) [18,19] y la arquitectura persistente del almacenaje [13, 20,21]. Todas las capas arquitectónicas de Napier88 son virtuales, en cualquier puesta en práctica, ellas se pueden poner en ejecución por separado o juntas dependiendo de la eficacia.

La versión actual del lenguaje Napier88 es la versión 2.2.1[22]. El lenguaje tiene solamente

algunos cambios a el del lanzamiento 1,0 [23,24] pero el ambiente persistente se ha enriquecido y se ha reorganizado perceptiblemente. Los cambios al lenguaje son: un modelo abstracto dinámico del testigo para los tipos abstractos, y tipos de operadores.

Los cambios principales al ambiente persistente son la disposición de un browser, un compilador para la programación reflexiva, hilos de rosca y los semáforos, una nueva organización del almacén de objetos para proporcionar una navegación libre en el almacén, almacenes distribuidos con la exploración alejada y un sistema de hiperprogramación. El ambiente también proporciona un mecanismo, a través de Internet, para otros sitios para compartir los programas y los datos que se pueden alcanzar por la exploración y la copia, separada de otros repositorios Napier88.

Por otro lado **HyperPascal** es un nuevo lenguaje visual revolucionario que ha demostrado con éxito la viabilidad de la idea de hiperprogramación. Sus capacidades de manipulación de datos se modelan en PASCAL (ISO), pero el uso de las marcas de sintaxis usan capacidades actuales del GUI (Interfaz Gráfica de Usuario). Sin embargo, como prototipo, tiene limitaciones. Por ejemplo, no proporciona ninguna ayuda de funcionamiento, y el ambiente lingüístico es anticuado (el PASCAL fue elegido como caso de prueba para las ideas de hiperprogramación por su simplicidad, no por su modernidad). En este proyecto, usted desarrollaría en un ambiente completamente compilado para HyperPascal, con la ayuda de un run-time (entorno de ejecución), las instalaciones separadas de la compilación, el acceso a las rutinas externas de la biblioteca, el uso de un instrumento de sistema más actualizado con características lingüísticas más actualizadas, tales como orientación a objetos.

Otro lenguaje que permite hiperprogramación es **PJama** que es un prototipo experimental que pone la persistencia ortogonal en ejecución para la plataforma de Java (OPJ), que es un acercamiento a hacer objetos del uso persistente entre las ejecuciones de programa con el esfuerzo mínimo posible requerido.

Requisitos para proporcionar hiperprogramación en Java:

- un repositorio persistente con root(s), disponibilidad e integridad referencial.
- reflexión lingüística como técnica de programación.
- un browser del repositorio persistente.
- una especificación de hyper-links escrito en Java.
- un mecanismo para preservar hyper-links sobre la compilación.
- un editor de hiperprogramación.
- Una interfaz gráfica de usuario.

La impresión de hiperprogramas y de la transferencia de hiperprogramas a partir de un sistema a otro es obstaculizada por la presencia de hyper-links. Es, sin embargo, posible traducir cada hyperprogram al HTML, representando los hyper-links como URLs. Esto fue hecho para publicar la fuente del compilador Napier88, que es en sí mismo un hiperprograma, y es lo que hace también Java.

**HyperWeb** es una aplicación web para la construcción de aplicaciones web. Las aplicaciones web se construyen vía un interfaz del alto nivel, o la representación del hipercódigo. Por razones pragmáticas, esta representación del hipercódigo hace las páginas web en una manera similar a la que se mostrará en el web browser del usuario. Cada elemento se aumenta con los componentes adicionales que proporcionan el hipercódigo que

corrige funcionalidad.

Las aplicaciones web se construyen con los objetos que se crean y se almacenan en un servidor persistente que proporciona seguridad de tipo y las características de integridad referencial en estos objetos en tiempo de ejecución, y continuamente mantenerlos durante el ciclo de vida de la aplicación. En contraste, la herramienta de lado del cliente no puede asegurarse de que los objetos no sean suprimidos, sean movidos, o no substituidos por los objetos de diverso tipo una vez que la aplicación sea desplegada o en la evolución subsecuente (o concurrente) de la aplicación.

HyperWeb proporciona la primera prueba de la existencia del uso del concepto de hiperpogramación para la construcción de aplicaciones web. La interfaz de HyperWeb se genera enteramente dentro de los límites del HTTP estándar y es por lo tanto accesible vía cualquier web browser. HyperWeb es una aproximación del lado del servidor en el cual las aplicaciones web se construyen fuera de los objetos que se crean y se mantienen en un servidor persistente. Como tal, las garantías estáticas de la seguridad del programa, se proporcionan y se mantienen durante el ciclo de vida de la aplicación.

Los acercamientos más actuales a la seguridad del desarrollo web están basados en el cliente. Pocos acercamientos existentes han adoptado un modelo del lado del servidor, y por lo tanto las ventajas obtenidas de seguridad del lado del servidor no se alcanzan. Los sistemas relevantes con metas similares al de HyperWeb incluyen el < bigwig > y el sistema de WÓbjects.

## 3. CONCLUSIONES

El concepto de hiperprogramación transfiere de un lenguaje persistente polimórfico común a un lenguaje persistente polimórfico orientado a objetos.

La hiperprogramación ofrece varias ventajas referidas al rendimiento y la evolución natural que apunta a un entorno orientado a objetos.

El término "Persistente" se aplica a los sistemas que proporcionan un mecanismo transparente y automático (en mayor o menor medida) de salvaguarda y recuperación de objetos. Entre estos sistemas se encuentran sistemas operativos orientados a objetos, sistemas gestores de bases de datos orientados a objetos, y sistemas de programación orientados a objetos.

La hiperprogramación proporciona un nuevo estilo para enlazar programas a datos persistentes, permitiendo que los acoplamientos directos a los datos sean incorporados en los programas fuente ejecutados en un repositorio persistente. Se destacan ventajas tales como: en detalle permite una composición más apropiada de programas y permite enlaces entre los programas ejecutables y el código fuente, en contraste con los sistemas tradicionales que confían en la convención. La hiperprogramación permite que un código fuente contenga enlaces a los artículos de datos en el repositorio persistente. Esto mejora la confiabilidad del programa. También reduce la cantidad de código escrita permitiendo enlaces directos a los datos en el lugar de descripciones textuales. Ambas técnicas contribuyen a la comprensión del ambiente persistente con el soporte de la puesta en práctica de las herramientas del navegador del almacén y permiten que las representaciones del código fuente sean asociadas a todos los programas

ejecutables en el repositorio persistente.

## BIBLIOGRAFÍA O REFERENCIAS


1. http://www-systems.cs.st-andrews.ac.uk/wiki/Hyper_Programming/.
2. http://www-systems.cs.st-andrews.ac.uk/wiki/Persistence/.
3. Kirby, G.N.C., Connor, R.C.H., Cutts, Q.I., Dearle,A., Farkas,A.M. & Morrison, R. "Persistent Hyper-Programs". In Proc. 5th International Workshop on Persistent ObjectSystems, San Miniato, Italy (1992).
4. Atkinson MP, Morrison R, Pratten GD. Designing a Persistent Information Space Architecture. In: Proc. 10th IFIP World Congress, Dublin, 1986, pp 115-20.
5. Connor RCH. The Napier Type-Checking Module. Universities of Glasgow and St Andrews Report PPRR-58-88, 1988.
6. PS-algol Reference Manual, 4th edition Universities of Glasgow and St Andrews Technical Report PPRR-12-88 (1988).
7. Polimorfismo, reutilización de la persistencia y del software en un ambiente orientado al objeto fuertemente mecanografiado Morrison, R., marrón, A.L., Connor, R.C.H. y Dearle, A. Tecnología de dotación lógica Journal, diciembre (1987) pp 199-204.
8. Tipos, atascamientos y parámetros en un ambiente persistente Atkinson, M.P. y Morrison, R. En tipos y persistencia de datos, Atkinson, M.P., Buneman, O.P. y Morrison, R. (ed), Springer-Verlag (1988) pp 3-20.
9. El rendimiento en Sistemas de Programación Persistentes. J. Baltasar García Perez-Schofield1, Tim B. Cooper2, Emilio García Roselló3, Manuel Pérez Cota3 1Escuela Superior de Ingeniería Informática. Departamento de Informática. Universidad de Vigo [ Bro89 ] El Objeto Persistente Almacena El Marrón, A.L. Ph.D. Tesis, Universidad de St Andrews (1989).
10. En la construcción de los ambientes de programación persistentes Dearle, A. Ph.D. Tesis, universidad de St Andrews (1988).
11. Tipos y polimorfismo en los sistemas de programación persistentes Connor, R.C.H. Ph.D. Tesis, universidad de St Andrews (1990).
12. Entregar las ventajas de la persistencia a la construcción y a la ejecución Cutts del sistema, Q.I. Ph.D. Tesis, universidad de St Andrews (1992).
13. Reflexión e Hiperactivo-Programacio'n en los sistemas de programación persistentes Kirby, G.N.C. Ph.D. Tesis, universidad de St Andrews (1992).
14. El Marrón De la Máquina, el A.L., El Carrick, El R., El Connor, el R.C.H., El Dearle, El A. Y El Morrison Abstractos Persistentes, R. Universidades de Glasgow y del informe técnico Pprr-59-88 (1988) del St Andrews.
15. La Máquina Abstracta Persistente Connor, R.C.H., Marrón, A.L., Carrick, R., Dearle, A. Y Morrison, R. En sistemas persistentes, Rosenberg, el J. y Koch del objeto, D.M. (ed), Springer-Verlag (1990) pp 353-366.
16. On the Integration of Concurrency, Distribution and Persistence Munro, D.S. Ph.D. Thesis, University of St Andrews (1993).
17. The Napier88 Reference Manual (Release 2.0) Morrison, R., Brown, A.L., Connor, R.C.H.,

Cutts, Q.I., Dearle, A., Kirby, G.N.C. & Munro, D.S. University of St Andrews Technical Report CS/94/8 (1994).

18. El Manual De Referencia Napier88 Morrison, R., Marrón, A.L., Connor, R.C.H. Y Dearle, A. Universidades de Glasgow y del informe técnico Pprr-77-89 (1989) del St Andrews.

19. Napier88 Release 1.0 Morrison, R., Brown, A.L., Connor, R.C.H. & Dearle, A. University of St Andrews (1989).

20. http://www.cs.berkeley.edu/~benr/publications/auscc04/papers/fox-auscc04.pdf.

21. http://www.naccq.ac.nz/conference05/proceedings_05/posters/349.pdf.

22. http://hopl.murdoch.edu.au/showlanguage2.prx?exp=5108.

23. http://www.naccq.ac.nz/conference04/proceedings_03/pdf/231.pdf

24. p.lyons@massey.ac.nz (investigadores de hyperpascal).