Implementaciones

Existen varias implementaciones de Smalltalk disponibles:

Squeak: gratuito, open-source y multi-plataforma. Desarrollado activamente.

VisualWorks: implementación proprietaria y multi-plataforma, disponible gratuitamente para uso no comercial.

Gemstone: implementación propietaria que incluye una base de objetos de alto rendimiento

Y otros: GNU Smalltalk, Smalltalk/X, SyX, VA Smalltalk, Dolphin...

Aplicaciones

Desde su creación en los principios de los 80's, Smalltalk ha sido extensamente usado en investigación académica así como en aplicaciones comerciales. Aquí hay algunas aplicaciones Smalltalk actuales que contribuyen al avance de la tecnología del software.

Educación: EToys (Squeak), SqueakBot, BotsInc, Scratch...

Multimedia: Sophie, OpenCroquet, Plopp...

Desarrollo web: Seaside, Aida, Komanche, Swazoo...

Persistencia: bases de objetos (Magma, GemStone), bases de datos relacionales (MySQL, PostgreSQL), mapeo objeto-relacional (Glorp).

Una sesión de dibujos de Plopp

Glosario

Imagen: El ambiente de Smalltalk provee un almacenamiento persistente de objetos, la imagen. Esta contiene código de las aplicaciones (clases y métodos), objetos que mantienen el estado de las aplicaciones y hasta puede incluir las herramientas de desarrollo para inspeccionar y depurar el programa mientras se esta ejecutando.

Máquina Virtual: Una máquina virtual es un programa que es capaz de ejecutar otros programas. Facilita la portabilidad de la aplicación.

Reflexión: Un lenguaje es reflexivo cuando provee mecanismos para inspeccionar y modificar el código de un programa durante la ejecución del mismo.

Tipeo dinámico: Algunos lenguajes fuerzan al desarrollador a especificar el tipo de cada variable (integer, string...); esto es llamado tipeo estático. Tipeo

Smalltalk

un lenguaje de programación
puramente orientado a objetos
y un ambiente dinámico

EUROPEAN SMALLTALK USER GROUP
esug
www.esug.org

Conceptos importantes de Smalltalk

Smalltalk es un lenguaje orientado a objetos y dinámicamente tipado, con una sintáxis simple que puede ser aprendida en quince minutos. Su mayor ventaja es ser muy consistente:

- todo es un objeto: clases, métodos, números, etc.
- una pequeña cantidad de reglas, y ninguna excepción!

Smalltalk corre sobre una máquina virtual. Se desarrolla sobre una imagen donde todos los objetos viven y son modificados.

Sintaxis de Smalltalk

Palabras reservadas	
`nil`	objeto indefinido (valor por defecto de las variables)
`true` y `false`	objetos booleanos
`self`	objeto receptor del mensaje
`super`	objeto receptor del mensaje en el contexto de la super clase
`thisContext`	objeto contexto de ejecución del método actual

Caracteres reservados	
`:=` (o ←)	asignación
`^` (o ↑)	devuelve el resultado de un método
`| var1 var2 var3 |`	declaración de variables temporales
`$a`	caracter `a`
`#(abc 123)`	arreglo con literales: el símbolo `#abc` y el número `123`
`.` (punto)	fin de expresión
`;`	mensajes en cascada
`[ ]`	bloque de código (es un objeto !)
`"comentario"`	
`'string'`	

Envío de mensajes

Un método es invocado al enviar un mensaje a un objeto (el receptor) y el mensaje devuelve un objeto. El mensaje está basado en el lenguaje natural, conformado por un sujeto, un verbo y argumentos. Existen tres tipos de mensajes: unario, binario y de palabra clave.

Mensajes unarios. El mensaje unario no tiene argumentos.

```
array := Array new.
array size.
```

El primer ejemplo crea y devuelve una nueva instancia de la clase Array, al enviarle el mensaje `new`. El segundo ejemplo solicita el tamaño de ese arreglo y devuelve `0`.

Mensajes binarios. El mensaje binario tiene solo un argumento, su nombre es un símbolo y es usado frecuentemente para expresiones aritméticas.

```
3 + 4.
'Hola', ' Mundo'.
```

El mensaje `+` es enviado al objeto `3` con `4` como parámetro. En el segundo caso, el mensaje `,` es enviado al string `'Hola'` con `' Mundo'` como parámetro.

Mensajes de palabra clave. Un mensaje de palabra clave puede tener uno o más argumentos. Los argumentos se sitúan entre cada palabra clave, después de los dos puntos.

```
'Smalltalk' allButFirst: 5.
3 to: 10 by: 2.
```

El primer ejemplo invoca el método `allButFirst:` sobre un string y con el argumento `5`. El método devuelve el string `'talk'`. El segundo ejemplo devuelve una colección con los elementos `3`, `5`, `7` y `9`.

Bloques

Los Bloques son objetos que contienen código que no se ejecuta inmediatamente. Son la base para estructuras de control como decisiones condicionales o lazos. Se pueden utilizar para agregar comportamiento , ej, en los items de un menu.

```
#('Hola' ' Mundo')
       do: [:string | Transcript show: string].
```

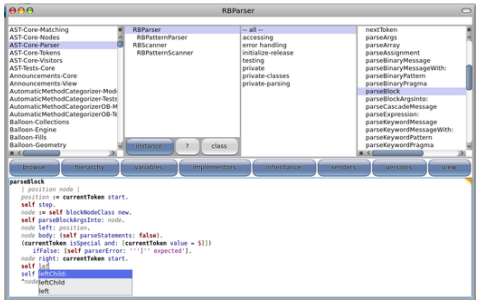
El ejemplo envía el mensaje `do:` a un arreglo de strings con un bloque como parámetro. El bloque se evalúa una vez por cada elemento del arreglo. El parámetro del bloque `string` contiene cada elemento del arreglo, uno a la vez. Como resultado de toda la expresión se muestran en el trascript los strings `'Hola'` y luego `' Mundo'`.

Entorno de desarrollo

La mayoría de las implementaciones de Smalltalk proveen un entorno de desarrollo integrado que permite explorar el código fuente e interactuar con objetos. Muchas herramientas implementadas en Smalltalk están disponibles gracias a su API reflexiva:

- browser de clases y de métodos (+ refactoring);
- inspectores de objetos;
- un depurador;
- administración de releases y control de versiones;
- y mucho, mucho más!

El código puede ser inspeccionado y ejecutado directamente en la imagen, usando combinaciones de teclas y menús.



El browser de clases de Pharo