Clasificaciones: Un mecanismo de herencia múltiple para la construcción de modelos fáciles de comprender y mantener

Martín León Aristiaran, Mario M. Berón
Departamento de Informática
Universidad Nacional de San Luis
San Luis – Argentina
emails: mlarist@yahoo.com mberon@unsl.edu.ar

Pedro Rangel Henriques
Departamento de Informática
Universidade do Minho
Braga – Portugal
pedrorangelhenriques@gmail.com

Maria João Pereira
Departamento de Informática
Instituto Politécnico de Bragança
Bragança – Portugal
mjoao@ipb.pt

Resumen

Los actuales mecanismos de Múltiple Herencia (MH) han resuelto gran parte de sus problemas históricos como el problema del diamante y la herencia de distintas implementaciones de métodos y atributos con el mismo nombre.

Sin embargo, los modelos producidos por estos mecanimos presentan al menos uno de los siguientes problemas: i) No contienen información acerca de como es posible extenderlos asegurando la inexistencia de conflictos, o ii) El modelo creado no puede inferirse a partir de características de los elementos del dominio. La existencia de esos problemas agrega gran complejidad al mantenimiento de esos modelos, lo cual desanima el uso de la MH.

En este contexto, este artículo describe una línea de investigación cuyos principales objetivos son crear mecanismos de MH que: i) Brinden una solución a los dos problemas mencionados anteriormente, lo cual facilita el mantenimiento de los modelos, y ii) Produzcan modelos inteligibles y expresivos.
Palabras clave Clasificaciones, Herencia múltiple

1. Contexto

Este trabajo se encuentra enmarcado en el contexto de dos proyectos de investigación, ellos son: i) Ingeniería de Software: Conceptos Métodos y Herramientas en un Contexto de Ingeniería de Software en Evolución y ii) Quixote: Desarrollo de Modelos del Dominio del Problema para inter-relacionar las Vistas Comportamental y Operacional en Sistemas de Software a fin de Facilitar su Comprensión.

El primero es uno de los principales proyectos desarrollados en la Universidad Nacional de San Luis. Es reconocido por el programa de incentivos y tiene gran éxito nacional e internacional. La afirmación antes realizada se sustenta en la presentación de muchas publicaciones en conferencias de reconocido prestigio y en las relaciones que dicho proyecto mantiene con otras universidades extranjeras. Como fruto de esas relaciones, muchos egresados de universidades argentinas han tenido la posibilidad de realizar sus estudios de maestría y doctorado en el exterior.

El segundo, Quixote, es un proyecto bilateral entre Argentina y Portugal que está avalado por el Ministerio de Ciencia, Tecnología e Innovación Productiva de la Nación Argentina (Mincyt) y la Fundação para a Ciência e Tecnología de Portugal (FCT). En dicho proyecto participan la Universidad Nacional de San Luis y la Universidad de Minho.

Entre las tareas realizadas en el contexto del proyecto Quixote, se encuentran diferentes misiones de

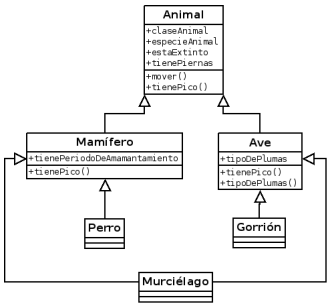

Figura 1: Problemas de ambigüedad de la MH

investigación realizadas desde Argentina hacia Portugal y viceversa. Todas esas misiones han sido solventadas por el Mincyt y la FCT.

2. Análisis de los mecanismos de múltiple herencia y propuesta superadora

El debate sobre la MH lleva décadas en la literatura de orientación a objetos y aún hoy en día es una cuestión en la cual los científicos de la computación mantienen grandes diferencias.

El centro de estos debates no pasa por si la MH es necesaria o no, las limitaciones de la herencia simple son bastante claras, o si es posible solucionar los problemas de ambigüedad de la MH. Es claro que hay varias formas de evitar estas ambigüedades [1, 8, 2, 3, 4]. El debate central gira en torno a si la complejidad que cada una de las implementaciones de MH actuales justifica las ventajas de su adopción [5]. Teniendo en cuenta esto, el desafío es crear un mecanismo de MH que solucione los problemas de ambigüedad, evite la duplicación de código y produzca modelos tan o más fáciles de comprender y mantener que aquellos con herencia simple. Eso constituye el principal objetivo de esta línea de investigación.

3. Análisis de los problemas de la MH

La Figura 1 ilustra el problema de ambigüedad introducido por la MH. Dicho problema aparece por la herencia de atributos y métodos distintos pero con el mismo nombre. Esa figura expresa que las clases `Ave` y `Mamífero` heredan (atributos y métodos) de `Animal`, que la clase `Gorrión` hereda de `Ave` y que la clase `Perro` hereda de `Mamífero`. El problema surge cuando una clase pretende heredar de dos o más clases que definen un mismo atributo o método. Por ejemplo, si se define que `Murciélago` es una subclase tanto de `Mamífero` como de `Ave`, ¿La clase debería heredar el método `tienePico()` de `Ave` o de `Mamífero`?.

Cada mecanismo de MH resuelve de distintos modos este problema, sin embargo cada uno de estos mecanimos introducen a su vez nuevos inconvenientes.

En las subsecciones siguientes se describirán cada uno de estos mecanismos de MH y sus problemas.

3.1. Patrones para simular herencia simple

Previo al análisis de mecanismos de MH se han analizando diversos patrones [6, 7] que pretenden representar con herencia simple algunas situaciones de diseño en los que la MH es conveniente.

Las soluciones que aportan todos esos patrones sufren de al menos uno de los siguientes inconvenientes: i) Las clases resultan más complejas y con demasiadas responsabilidades, o ii) Los modelos producidos resultan menos expresivos y difíciles de mantener.

3.2. Linearización

Este mecanismo soluciona el problema de la ambigüedad estableciendo un orden a las superclases [1]. De este modo, los atributos y métodos son buscados en las primeras superclases y luego en las últimas. Implementaciones de este enfoque pueden encontrarse en lenguajes como CLOS, Python o Perl 6.

La composición y aplicación de mixins es también lineal [2] y por lo tanto las críticas de esta subsección son también aplicables a ese mecanismo.

Un problema con este este mecanismo es que en algunas situaciones requiere duplicar código para lograr que una clase posea los atributos y métodos deseados.

Por ejemplo, si `Murciélago` hereda primero de `Ave` y luego de `Mamífero`, `Murciélago` here-

dará `mover()` de `Mamífero`, lo cual no es correcto. Por otro lado, si el orden es primero `Mamífero` y luego `Ave`, `Murciélago` heredará `tienePico()` de `Ave`, lo cual tampoco es correcto.

En el primer caso, se deberá sobrecargar `Murciélago` con la misma implementación de `mover()` de `Ave` y en el segundo se deberá sobrecargar `Murciélago` con la implementación `tienePico()` de `Mamífero`. Ambos casos implican duplicación de código.

Otro problema importante de la linearización se encuentra en que el orden de las superclases no guarda relación con ningún concepto del dominio. Esto se debe a que es sólo un agregado arbitrario al modelo que debe ser memorizado por el desarrollador para entener como opera la herencia.

La complejidad dada por los dos motivos señalados hace que los modelos producidos sean difíciles de comprender y de extender.

3.3. Herencia selectiva

Otro modo de evitar el problema de la ambigüedad de la MH es permitir que se puedan heredar algunos atributos y métodos pero no todos [3].

Un lenguaje que adopta esta solución es Eiffel, el cual permite heredar, excluir o renombrar los métodos de las superclases.

Tomando el ejemplo de la Figura 1, con herencia selectiva se podría excluir al método `tienePico()` en la relación de herencia entre `Ave` y `Murciélago` y excluir al método `mover()` en la relación de herencia entre `Mamífero` y `Murciélago`.

Otro mecanismo que permite excluir sólo algunos métodos es el basado en rasgos (traits) [4]. En la composición de rasgos es posible excluir métodos individuales de alguno de los rasgos heredados, lo cual puede ser usado para salvar la ambigüedad dada por la composición de rasgos que definen métodos con los mismos nombres.

El problema de este enfoque radica también en la complejidad de los modelos producidos. Ese resultado se obtiene porque el desarrollador necesita tener encuenta dos aspectos importantes: i) La jerarquía de clases, y ii) Si cada atributo o método ha sido omitido o renombrado en cada una de las relaciones de herencia.

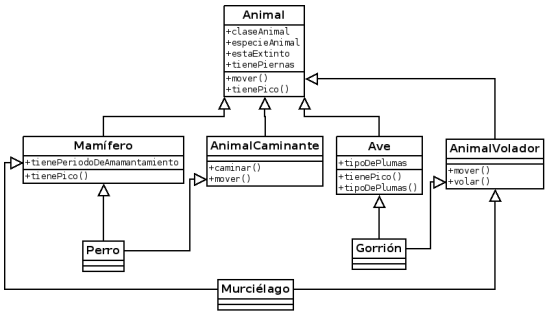

Figura 2: Separación de clases

3.4. Referencia explicita a atributos y métodos

Otro mecanismo -usado en C++- permite la ambigüedad en la herencia pero requiere que se indique la clase que contiene la implementación deseada en cada referencia al método o atributo ambigüo [8]. Es posible entender mejor este mecanismo volviendo a analizar el ejemplo de la Figura 1. Con este mecanismo se podría tener a `Murciélago` como subclase de `Ave` y de `Mamífero`, a pesar de la ambigüedad de los métodos `tienePico()` y `mover()`, pero se tendrá que explicitar cual de las implementaciones es la deseada al hacer referencia a ellos en una instancia de `Murciélago`.

Por ejemplo, en C++, si `unMurciélago` es una instancia de `Murciélago` entonces para hacer referencia a los métodos `tienePico()` y `mover()` se tiene que indicar respectivamente: `unMurciélago->Mamífero::tienePico()` y `unMurciélago->Ave::mover()`.

Este enfoque cuenta con dos problemas: i) Limita el polimorfismo y ii) Requiere un nivel de compresión del modelo al nivel de atributos y métodos (en lugar de a nivel de clases).

3.5. Separar clases para que puedan heredarse completamente

Otro modo para evitar la ambigüedad de la MH consiste en aislar los métodos ambigüos en nuevas clases y heredar apropiadamente. Volviendo al ejemplo de la Figura 1, esto implicaría crear nuevas clases para `tienePico()` y `mover()`, de modo que `Perro`, `Gorrión` y `Murciélago` puedan heredar clases “enteras”.

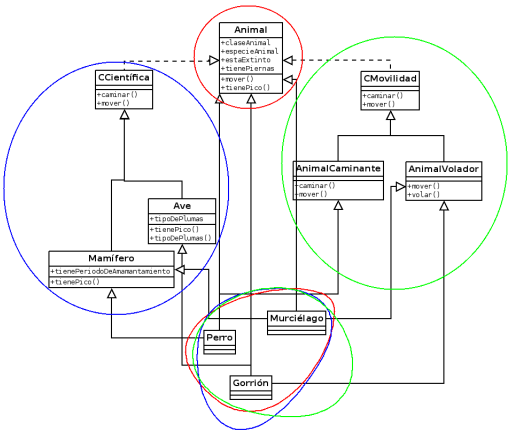

Figura 3: Un modelo basado en clasificaciones

Este mecanismo cuenta con las siguientes ventajas: i) No requiere linearización , ii) No requiere una comprensión del modelo al nivel de atributos y métodos y iii) Evita la duplicación de código.

Sin embargo, cuenta con las siguientes desventajas: i) No le da al desarrollador un modo de distinguir entre que clases pueden ser heredadas conjuntamente sin provocar problemas de ambigüedad, ii) No ofrece criterios para eliminar esa ambigüedad y iii) No siempre es posible separar la clases porque un método podría requerir invocar de un método definido en otra clase.

4. Mecanismo de MH propuesto

4.1. Clasificaciones

La presente investigación ha desarrollado un mecanismo de MH cuyo objetivo es evitar las desventajas de los mecanismos analizados en las subsecciones previas.

El mecanismo propuesto se basa en añadir al modelo un nuevo concepto: el de **clasificación**.

Las clasificaciones suplen de un aspecto inexistente en todos los lenguajes de programación y de modelado orientado a objetos: la distinción entre las clases que representan facetas complementarias y las que representan facetas excluyentes de un conjunto de elementos del dominio. En este sentido, una clasificación puede pensarse como un conjunto de clases que simbolizan alternativas (excluyentes) de una misma faceta.

El concepto de clasificación es importante en la definición de un mecanismo de MH por dos motivos: i) Es fácilmente identificable en características de los elementos del dominio, y ii) Permite distinguir entre que clases pueden ser heredadas conjuntamente y cuales no. El primer motivo permite que los modelos producidos sean fáciles de comprender y de construir. El segundo permite establecer restricciones que eliminen la posibilidad de ambigüedad *sólo* sobre las clases que tienen posibilidades de ser superclases de una misma clase.

Para la eliminación de la ambigüedad se definen dos restricciones sobre las clasificaciones: i) Los métodos y atributos usados en las clases de una clasificación no pueden ser usados en otras clases y ii) Una clase no puede heredar de dos o más clases de una misma clasificación. Las restricciones impuestas están vinculadas con la semántica de los elementos del dominio. Ésta vinculación se debe a que resultaría contradictorio que una clase herede de dos clases que representan conceptos opuestos (distintas implementaciones de una misma faceta) y a que no se espera que clases que que representan distintas facetas requieran definir los mismos atributos y métodos.

Otra característica también vinculada a la semántica a los elementos del dominio es el hecho de que cada clasificación es válida sólo para un conjunto de elementos del dominio, no para todos. Por ejemplo, una clasificación puede estar definida para los animales, pero no para los gráficos o los números.

4.2. Construcción de un Modelo Basado en Clasificaciones (MBC)

Un ejemplo de MBC - equivalente al modelo de clases de la Figura 2 - es mostrado en la Figura 3. Ese modelo contiene dos clasificaciones: i) Una clasificación científica (`CCientífica` y sus subclases) que sólo posee características sobre la apariencia de los animales y ii) Una clasificación de movilidad (`CMovilidad` y sus subclases) referida a como el animal se mueve.

Una clasificación se define estableciendo una nueva relación entre clases -llamada relación de clasi-

ficación[1] - entre dos clases: i) La clase clasificada y ii) La clase raíz de la clasificación. Las clases de la clasificación son dicha clase raíz y todas sus subclases. La clase clasificada no forma parte de la clasificación, esta clase representa el conjunto de clases para las cuales la clasificación es aplicable.

Existen otros aspectos que son necesarios mencionar sobre las clases de una clasificación: i) Sólo pueden especializar a clases pertenecientes a clasificaciones definidas en la clase clasificada, ii) Dentro de los métodos de una clase puede hacerse referencia a atributos y métodos de la clase clasificada como si fuesen propios o de alguna de las clasificaciones de su clase clasificada, y iii) Sólo se esperan instancias de las clases que posean superclases de todas las clasificaciones, llamadas clases concretas (en contraposición a las clases abstractas). En cuanto a los aspectos i) y iii) mencionados en el párrafo anterior, las clasificaciones pueden tener cierta similitud con el mecanismo de MH del lenguaje CZ [5]. Sin embargo, el mecanismo basado en clasificaciones posee algunas características que no se encuentran en CZ, como: 1) Cada clasificación posee un espacio de nombres propios para definir atributos y métodos; 2) La clase de una clasificación no sólo puede definir como propios los métodos y atributos de su clase clasificada -como en CZ- sino también de todos los métodos y atributos de las clasificaciones definidas en su clase clasificada, y 3) Para ser considerada concreta no sólo debe extender su clase clasificada sino tambien clases de todas las clasificaciones definidas en esta y en las superclases de esta. En la Figura 3 se han enmarcado en líneas de distintos colores cada una de las clasificaciones del modelo. Nótese que las clases concretas pertenecen a todas las clasificafiones definidas en sus clases clasificadas.

4.3. Conclusiones

Hasta el momento, se ha podido detectar una estrategia innovadora que elimina los problemas presentados por los mecanismos de herencia múltiple actuales. Ésta estrategia consiste en permitir la representación, en el modelo, de la distinción entre tipos de subclases que representan facetas complementarias y excluyentes de una clase. El mecanismo de MH basado en Clasificaciones es conveniente porque: i) Esta basada en elementos del dominio claramente identificables, ii) Permite definir las restricciones que evitan la ambigüedad sólo sobre las clases que representan facetas complementarias y iii) El modelo generado posee información acerca de como puede extenderse evitando la ambigüedad.

[1] La relación de clasificación no es una relación de herencia

5. Formación de Recursos Humanos

Las tareas realizadas en la presente línea de investigación están siendo llevadas a cabo como parte de una tesis de Maestría en Ingeniería de Software en la Unviersidad Nacional de San Luis. Se espera que una vez finalizada dicha tesis se deriven de la temática abordada diferentes tesis de licenciatura, maestría y si es posible doctorado.

Referencias

[1] Barrett K, Cassels B, Haahr P, Moon DA, Playford K & Withington PT. **A Monotonic Superclass Linearization for Dylan.**. In *OOPSLA'96 Conference Proceedings*. ACM Press. 1996

[2] Bracha G & Cook W. **Mixin-based inheritance.**. In *Proc. OOPSLA'90*. ACM Press. 1990.

[3] Meyer B. **Object-Oriented Software Construction**. Prentice Hall. 1997.

[4] Schärli N, Ducasse S, Nierstrasz O & Black A. **Traits: Composable Units of Behavior**. 2002

[5] Malayeri, Donna;Aldrich, Jonathan. **CZ: Multiple Inheritance Without Diamonds**. 2009

[6] Mössenböck, Hanspeter. **Twin - A Design Pattern for Modeling Multiple Inheritance**. 1999

[7] Bäumer, Dirk and Riehle, Dirk and Siberski, W. and Wulf, M. and Wulf, Martina. **The Role Object Pattern**. In Washington University Dept. of Computer Science. 1997

[8] Stroustrup, Bjarne. **Multiple Inheritance for C++**. 1999