Mejora de una representación genética genérica para modelos

Lorenzo Mandow[1*], José Antonio Montenegro[1**], and Steffen Zschaler[2]

[1] Universidad de Málaga, Andalucía Tech, Departamento de Lenguajes y Ciencias de la Computación, Málaga, España
[2] Department of Informatics, King's College London

Resumen La ingeniería dirigida por modelos (MDE) es una metodología de gran aceptación en la ingeniería del software. La ingeniería del software basada en la búsqueda (SBSE) propugna el uso de métodos metaheurísticos para la resolución de diversos problemas como la optimización y transformación de modelos. Recientemente se ha propuesto Crepe como una representación genética genérica para modelos, con el objetivo de facilitar el acceso a los métodos metaheurísticos a los practicantes de la MDE. En este trabajo propone una mejora de la representación Crepe y se evalúa sobre un conjunto de problemas generados aleatoriamente.

1. Introducción

La ingeniería del software basada en la búsqueda (Search Based Software Engineering - SBSE) es una disciplina que promueve la aplicación de técnicas metaheurísticas a los problemas de la ingeniería del software. Especial atención merecen los problemas relacionados con la optimización y transformación de modelos que aparecen en la ingeniería del software dirigida por modelos [7].

El acercamiento de estas técnicas a los profesionales de la ingeniería del software, y su integración práctica en las herramientas de trabajo habituales, como Eclipse, suponen un importante desafío. En este sentido, se puede destacar el reciente trabajo de Williams [10], que aborda el desarrollo de una representación genética genérica para modelos denominada Crepe, y su integración directa en las herramientas del Eclipse Modeling Framework [1].

El desarrollo de una representación genérica para modelos no es una tarea fácil. Como en tantos otros aspectos de la IA, el uso de una representación genérica, y por tanto muy expresiva, puede tener su contrapartida en una reducción de la eficiencia en la resolución de determinados problemas. Sin embargo, su integración en un conjunto de herramientas automatizadas para la optimización de modelos tiene un potencial de retorno muy elevado.

En este trabajo se analiza la representación de Crepe, y se señala un problema relacionado con la localidad de la representación [9]. Para solventarlo se

[*] La presentación de este trabajo está subvencionada por Plan Propio de Investigación de la Universidad de Málaga - Campus de Excelencia Internacional Andalucía Tech.
[**] Work supported by project MAVI TIN2012-34840 co-funded with FEDER funds.

propone una sencilla mejora basada en la recodificación de los individuos tras las operaciones de generación de sucesores.

Este trabajo se organiza de la siguiente manera. La sección 2 repasa brevemente los conceptos relacionados con la ingeniería dirigida por modelos, la ingeniería del software basada en la búsqueda, y la propuesta de Crepe. La sección 3 describe los aspectos de Crepe relevantes para este trabajo. En la sección 4 se describe una deficiencia potencial en la representación Crepe y se presenta una propuesta de mejora. La propuesta se evalúa en la sección 5 sobre un conjunto de problemas generados aleatoriamente en un dominio de prueba. Finalmente, se presentan algunas conclusiones y líneas de trabajo futuro.

2. Antecedentes

2.1. Ingeniería dirigida por modelos

La *ingeniería dirigida por modelos* (Model-Driven Engineering - MDE) es una metodología para la ingeniería de software que promueve el desarrollo de modelos (representaciones abstractas) para facilitar la estandarización de patrones de diseño, el desarrollo, la reutilización y el mantenimiento de software en general.

La *arquitectura dirigida por modelos* [2] (Model-Driven Architecture - MDA) es un conjunto de estándares promovidos desde 2001 por el consorcio Object Management Group (OMG) y que reflejan su propia visión de la MDE. Entre ellos se encuentran los estándares MOF (Meta-Object Facility), XMI (XML Metadata Interchange), y QVT (MOF Query/Views/Transformation). Actualmente, OMG gestiona también el estándar del lenguaje de modelado unificado (UML), un lenguaje gráfico de modelos para la especificación de sistemas.

MDA persigue la obtención de un *modelo de la aplicación* que sea independiente de los aspectos técnicos de la *plataforma* sobre la que se ejecutará (p. ej. *.NET Framework*, o *Java Platform EE*). El paso de un modelo de aplicación a otro de plataforma, se debe modelar a su vez mediante un conjunto de correspondencias y transformaciones que permitan automatizar el proceso.

MOF es el estándar de meta-modelado de OMG. Está definido en cuatro niveles. El nivel superior (denominado M3) es el propio lenguaje empleado por MOF para definir metamodelos. Los metamodelos se encuentran en el nivel M2. Uno de los ejemplos más conocidos es el metamodelo de UML, que define dicho lenguaje. Los propios modelos (por ejemplo, expresados en UML) se encuentran en el nivel M1, mientras que el nivel más inferior (M0) se dedica a los datos, por ejemplo en términos de objetos. Para facilitar el intercambio de los metadatos definidos a partir de metamodelos expresados en MOF se ha definido el estándar de intercambio de metadatos XMI. Una descripción más gráfica y detallada de estos conceptos aparece en [4].

El estándar EMOF (Essential MOF) es un subconjunto de MOF que reúne los elementos esenciales para la definición de modelos sencillos. Está estrechamente relacionado con Ecore, que puede considerarse la implementación de EMOF empleada por Eclipse Modeling Framework (EMF), el marco de modelado y generación de código Java basado en el entorno de desarrollo Eclipse.

2.2. Ingeniería del software basada en la búsqueda

La ingeniería del software basada en la búsqueda (SBSE - Search-Based Software Engineering) pretende la resolución de determinados problemas de la ingeniería del software mediante la aplicación de técnicas metaheurísticas. En [7] se puede encontrar una recopilación reciente de técnicas y aplicaciones.

El término «basado en la búsqueda» hace referencia a la definición de un espacio de posibilidades (espacio de estados) y entronca por tanto directamente con el concepto de representación simbólica empleado en Inteligencia Artificial. Los espacios de búsqueda que se plantean habitualmente en los problemas de ingeniería del software son de enorme tamaño. Por ese motivo se recurre a las técnicas metaheurísticas de optimización, que permiten alcanzar soluciones de muy alta calidad con recursos computacionales razonables. En esencia, la SBSE propone la aplicación de *modelos de decisión* basados en la *optimización* a los problemas de la ingeniería del software. Algunos ejemplos son la distribución de máquinas virtuales en configuraciones de centros de datos en la nube [3], o el diseño en la orquestación de servicios para aplicaciones móviles [5].

2.3. Crepe

Williams [10] investiga la hipótesis de que «una representación genérica, que pueda someterse a búsqueda, permitiría que la riqueza de investigación en SBSE fuese aplicable a un amplio rango de problemas del dominio MDE». Uno de los elementos centrales de dicho trabajo es una representación genética genérica para modelos que sigan el estándar MOF (como por ejemplo modelos UML), así como una implementación de esta representación para Ecore denominada *Crepe*. También presenta MBMS (Model-Based Metaheuristic Search), un marco de trabajo sobre EMF para la optimización de modelos basado en Crepe y en un conjunto de operadores genéticos. Este trabajo se centra únicamente en la representación, cuyos aspectos fundamentales se describen a continuación.

El trabajo sobre Crepe ha sido ampliado recientemente con la posibilidad de optimización multiobjetivo [6].

3. La representación de Crepe

Se describen a continuación brevemente los elementos centrales de la representación Crepe. Una descripción más exhaustiva puede encontrarse en [10].

3.1. Definición del espacio de búsqueda

En principio, el meta-modelo empleado en cada caso debería ser suficiente para definir el espacio de búsqueda, es decir, el espacio de modelos posibles. Por ejemplo, la figura 1 presenta un meta-modelo UML para el modelado de zoos [10]. Cada modelo de zoo puede tener un número de jaulas, y un número de animales en cada jaula. Cada jaula dispone de un espacio limitado, y cada

animal precisa de un determinado espacio dentro de su jaula. Adicionalmente, algunos animales pueden comerse a otros, por lo que no deberían de compartir jaula.

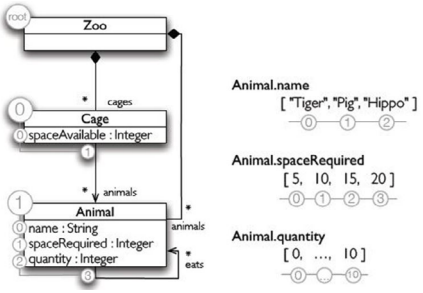


Figura 1: Meta-modelo para el dominio del zoo, e información de finitud para los atributos. Tomado de [10].

Las *características* (features) de cada meta-elemento pueden ser *atributos* o *referencias* a otros meta-elementos, por ejemplo, el espacio disponible en una jaula, o los animales que contiene respectivamente. El uso de tipos de datos básicos en los atributos del modelo (Integer y String), así como la inclusión de relaciones de asociación o referencias «de uno a muchos» en los meta-modelos (como el número de jaulas que puede haber en un zoo), resultan naturalmente en un espacio de modelos infinito.

Crepe mantiene un espacio de búsqueda finito, y ajustado a las necesidades del problema que se pretende resolver, de la siguiente manera: (1) las características de cada clase y los tipos de datos infinitos, se restringen obligatoriamente a un conjunto finito de valores; (2) se acota opcionalmente el ámbito de meta-clases y referencias, estableciendo el número máximo de instancias de cada meta-clase que pueden aparecer en el modelo, y de elementos a los que puede apuntar cada referencia; (3) se permite ignorar elementos (características, meta-elementos) que no sean relevantes para el problema a resolver; (4) se especifica obligatoriamente la *meta-clase raíz* que actúa como contenedor de los objetos del modelo.

Los elementos (1) y (2) se denominan *información para la finitud* (finitisation information), ya que su finalidad es acotar un espacio finito de modelos posibles. En general, todos ellos permiten controlar que el espacio finito sea más o menos extenso. Por ejemplo, una posible información para la finitud de las características asociadas a tipos infinitos en el meta-modelo del zoo aparece en la figura 1 [10]. Para la clase *Animal*, el atributo *name* sólo podrá tomar 3 valores

distintos, *spaceRequired* cuatro valores (se tomarán los mismos para el atributo *spaceAvailable* de la clase *Cage*), y *quantity* once valores.

3.2. Genotipo

Esta sección describe la *representación* del espacio de búsqueda finito descrito anteriormente. La representación genotípica de Crepe se basa en un vector de números enteros, y está parcialmente inspirada en la *programación genética cartesiana* [8]. La finalidad de usar este tipo de representación es ofrecer a los practicantes de la MDE un conjunto de métodos 'listos para usar': escalada, enfriamiento estadísico, o algoritmos genéticos, entre otros.

En la representación de Crepe el genotipo se divide en *segmentos*, cada uno de los cuales representa un *objeto* del modelo. El valor del primer gen de cada segmento identifica la *clase* del objeto. El resto del segmento está compuesto de *pares de característica* (feature pairs), es decir, pares de genes asociados a determinadas *características* del objeto: el primer elemento identifica la característica, y el segundo el valor. La figura 2 muestra la correspondencia entre un segmento y el objeto *Animal* correspondiente.

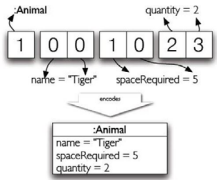


Figura 2: Correspondencia entre el genotipo de un segmento y la clase correspondiente del dominio del zoo. Tomado de [10].

El número máximo y mínimo de segmentos y de pares asociados a características son determinados por el usuario. Si el número de referencias está acotado, puede utilizarse dicha cota para elegir el número máximo de pares. En los pares de característica, el valor máximo de cada característica corresponderá al máximo de valores posibles de entre todas las características.

3.3. Fenotipo

La correspondencia entre un genotipo y su fenotipo se realiza asignando identificadores a los elementos del modelo y a los valores que pueden asignarse a cada una de sus características (aprovechando la información proporcionada para la finitud del espacio de búsqueda). Cada segmento se corresponde con un

objeto del modelo. Cada clase del meta-modelo recibe un identificador, así como cada característica de cada clase. Si una característica corresponde a un atributo de un tipo enumerado (tras aplicación de la finitud), su valor corresponderá con el índice posicional en la enumeración de valores (empezando por el índice cero).

En el caso de que varios pares de genes en un mismo segmento correspondan al mismo atributo, se opta arbitrariamente por tomar como valor del atributo el del último par. Igualmente, si el número de pares relativos a una misma referencia excede el máximo establecido, los últimos se ignoran.

Especial atención merecen para el presente trabajo las referencias entre objetos, que se discuten con más detalle en la sección 4. Las características de un objeto correspondientes a referencias reciben su valor una vez que se han creado todos los objetos, y estos han recibido a su vez un identificador. Para asignar una referencia de un objeto, se recogen primero todos los identificadores de objetos cuya clase es la misma que el tipo de la referencia. Por ejemplo, si se tratase de una referencia de una jaula a uno de sus animales, se recogerían todos los identificadores de objetos *Animal*. A continuación se elige el objeto tomando el valor módulo el número de candidatos. Si la referencia es de composición, el objeto dejará de ser parte de cualquier otro objeto para serlo únicamente del actual.

3.4. Operadores

En Crepe los operadores de mutación y cruce se encuentran limitados. El *operador de cruce* se limita de manera que el genotipo no pueda partirse dentro de un segmento. Esto permite que los descendientes de un cruce hereden objetos enteros de sus progenitores. Aún así, tal y como se comenta en más detalle en la sección 4, las referencias podrán verse muy afectadas en los cruces.

Por su parte, el *operador de mutación* permite mutar los distintos tipos de genes (selector de clase, selector de característica, y valor de característica) con distinta probabilidad. Por último, se incorporan dos *operadores adicionales* para crear y destruir segmentos en posiciones aleatorias de un individuo, permitiendo así cambiar el tamaño de los modelos generados.

4. Propuesta de mejora

El trabajo de Williams [10] evalúa el impacto del tamaño del genotipo en la redundancia y localidad de la representación [9]. Concretamente, en función del número de segmentos y pares de característica permitidos en la representación. En esta sección se describe una problemática adicional relacionada también con la localidad de la representación.

Considérese el modelo representado en la figura 3(a). En el aparece una única jaula que contiene dos animales, y cumple las restricciones de espacio y supervivencia. Aún así, todavía existe una tercera clase de animales que no están en ninguna jaula. Supongamos que se aplica mutación al gen de clase del primer segmento. El resultado se muestra en la figura 3(b). Los cambios en el genotipo se resaltan en negrita. Al desaparecer un objeto de clase *Animal*, el efecto no es

local, y la semántica de la representación se ve profundamente afectada. Concretamente, y como efecto secundario, la jaula contendrá ahora animales diferentes, violando tanto las restricciones de espacio como las de supervivencia. Además, la relación *eats* del objeto *O2* sufre también un cambio importante, y resulta que el mismo animal se encuentra en dos jaulas diferentes.

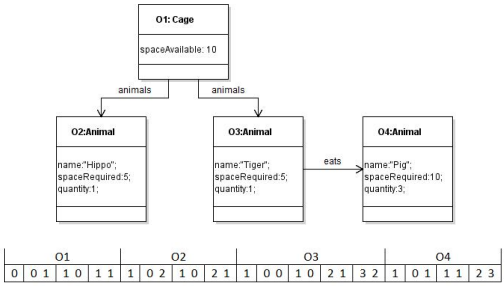
Un efecto similar se podría producir en caso de aplicar el operador de cruce, o mediante las operaciones de inclusión o borrado de segmentos. En general, cualquier operación que pueda hacer cambiar el número de objetos de una determinada clase en el modelo presentará esta problemática.

Lo deseable en la representación sería que, al realizar una mutación o cruce, se conservasen las buenas propiedades del individuo en las partes no afectadas por la operación. Una posible mejora en este sentido sería añadir un paso de *recodificación* del nuevo individuo, de manera que tras la mutación (o el cruce), todas las relaciones no afectadas directamente conserven su semántica previa. Para ello es suficiente con analizar las referencias antes y después de la operación, y cambiar los valores para que sigan referenciando los mismos objetos, siempre que estos se encuentren aún en el modelo. En el ejemplo de la figura 3, el modelo inicial tiene tres objetos *Animal*, identificados por su orden de ocurrencia en el genotipo del 0 al 2. En la nueva representación, tan sólo hay dos objetos *Animal*, identificados del 0 al 1. El paso de recodificación haría que en la nueva representación, las referencias a los animales 1 y 2 correspondiesen ahora al 0 y 1. El resultado se muestra en la figura 3(c). Nuevamente, los cambios en el genotipo (correspondientes en este caso a la mutación y la posterior recodificación) se muestran en negrita. En general, no es posible determinar cómo deberían reconsiderarse las referencias previamente inexistentes o referidas a objetos desaparecidos, pero al menos sería deseable que sólo se perturbasen las estrictamente necesarias. En la siguiente sección se evalúa la modificación propuesta sobre un caso de prueba.

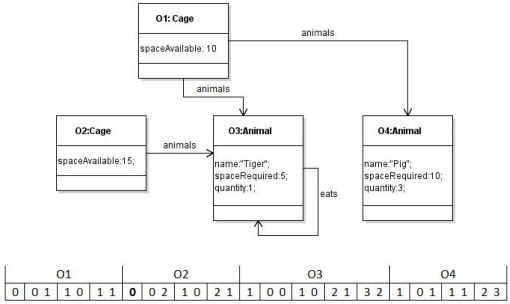
5. Evaluación

En la evaluación de la representación de Crepe se usará el meta-modelo del zoo y la información de finitud ya descritos en la sección 3.1 . El problema será encontrar una asignación consistente de animales a jaulas. Una posible aplicación de este problema aparece en [10] (sec. 4.1.1). Se establece *Zoo* como clase raíz, permitiendo objetos de las tres clases en el espacio de modelos. Se excluyen de la optimización la referencia *eats*, y el espacio requerido por cada animal.

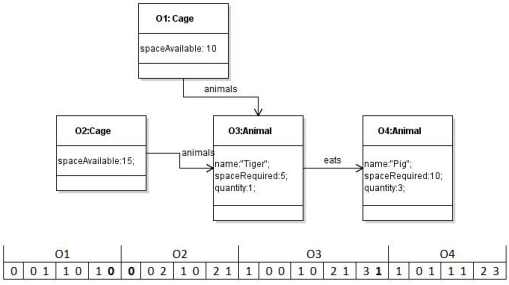
La función de aptitud reflejará las restricciones incumplidas en el modelo: (a) dos animales no deberían estar en la misma jaula si uno puede comerse a otro (en nuestro caso animales *Tiger* y *Pig*); (b) no debería haber objetos *Animal* sin asignar a ninguna jaula; (c) no debería superarse la capacidad de ninguna jaula; (d) no debería haber objetos alguna de cuyas características no esté especificada; (e) el modelo deberá albergar todos los animales descritos en la especificación del problema para cada tipo; (f) por último, ninguna jaula debería estar vacía. La función de aptitud suma 1 por cada incumplimiento de una restricción.



(a)



(b)



(c)

Figura 3: Un modelo (a), el resultado de mutar el bit de clase del objeto *O2* (b), y una alternativa que conserva mejor la semántica original (c).

Se analizará la evolución del valor de aptitud del mejor individuo a través de las distintas generaciones. Siendo ambas representaciones iguales, salvo por el paso de recodificación propuesto en la sección 4, cualquier cambio debería atribuirse precisamente a este paso. La figura 4 muestra la evolución del valor promedio de la aptitud del mejor individuo para Crepe con y sin el paso de recodificación. Los resultados están promediados sobre 50 problemas de manera que la suma de todos los animales en el modelo es 100 y la cantidad exacta de cada uno de ellos se determina aleatoriamente. También se muestra la mejor y peor aptitud alcanzada por el mejor individuo entre los 50 problemas. Se utilizó un algoritmo genético con elitismo del mejor individuo, selección por torneo entre cinco miembros aleatorios de la población, y probabilidad de mutación 0.015 para todos los genes (clase, característica y valor). El tamaño de la población fue de 100 individuos, con 100 segmentos por individuo, y 4 pares por segmento.

Los resultados muestran que la recodificación mejora la convergencia promedio y proporciona mejores soluciones. Tras 4000 generaciones, la solución mejora sobre el algoritmo original en 48 problemas y la iguala en 2. Las aptitudes promedio fueron 6.1 y 8.8 con y sin recodificación, respectivamente (mejora de un 30.6 %). Pequeñas variaciones de los parámetros mostraron tendencias similares.

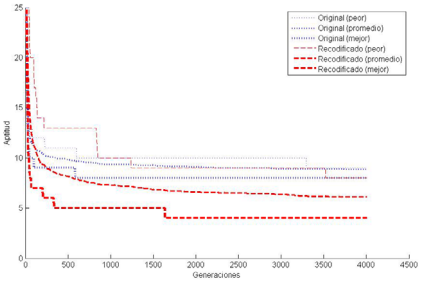


Figura 4: Evolución la aptitud (mejor, peor y promedio) del mejor individuo sobre 50 problemas de modelado de zoos. Datos para la representación Crepe con y sin recodificación.

6. Conclusiones y trabajo futuro

Recientemente se ha propuesto Crepe como una representación genética genérica para modelos basados en MOF. Su objetivo principal es servir de manera

genérica para la resolución de gran variedad de problemas de ingeniería del software mediante técnicas metaheurísticas. Se trata de un objetivo ambicioso pero con un gran potencial de retorno. Este trabajo analiza algunos aspectos de la representación genotípica de Crepe, basada en un vector de números enteros. Si bien Crepe permite la representación de cualquier modelo, se apunta al hecho de que determinadas operaciones pueden tener un gran impacto en la semántica de la representación.

En este sentido, se propone un sencillo paso de recodificación para preservar la semántica del modelo en aquellos aspectos no directamente afectados por las operaciones de cruce y mutación. El rendimiento de Crepe con y sin recodificación se ha analizado sobre un conjunto de problemas aleatorios sobre un dominio de prueba empleando un algoritmo genético. Los resultados muestran que, en este sencillo dominio, la recodificación mejora la convergencia del algoritmo.

En relación a los trabajos futuros, el análisis de la mejora propuesta sobre una mayor variedad de dominios y problemas de optimización es una continuación natural de este trabajo. Concretamente, sería interesante analizar problemas con una mayor riqueza de relaciones entre distintas clases de objetos, donde el impacto de las operaciones en la localidad de la representación Crepe original puede ser muy elevado. El diseño de representaciones alternativas, y su comparación con Crepe, es también una importante línea de trabajo futuro. Por último, sería muy interesante disponer de un banco de pruebas de problemas de optimización de modelos, así como realizar un análisis de los requisitos objeto de optimización más frecuentes.

Referencias

1. http://www.eclipse.org/modeling/emf/
2. http://www.omg.org/mda/
3. Chatziprimou, K., Lano, K., Zschaler, S.: Surrogate-assisted online optimisation of cloud iaas configurations. In: IEEE 6th International Conference on Cloud Computing Technology and Science, CloudCom 2014. pp. 138–145 (2014)
4. Djurić, D., Gašević, D., Devedžić, V.: The tao of modeling spaces. Journal of Object Technology 5(8), 125–147 (2006)
5. Efstathiou, D., McBurney, P., Zschaler, S., Bourcier, J.: Efficient multi-objective optimisation of service compositions in mobile ad hoc networks using lightweight surrogate models. J. UCS 20(8), 1089–1108 (2014)
6. Efstathiou, D., Williams, J.R., Zschaler, S.: Crepe complete: Multi-objective optimization for your models. In: Proc. of the First International Workshop on Combining Modelling with Search- and Example-Based Approaches. pp. 25–34 (2014)
7. Harman, M., Mansouri, S.A., Zhang, Y.: Search-based software engineering: Trends, techniques and applications. ACM Comput. Surv. 45(1), 11:1–11:61
8. Miller, J.F.: Cartesian genetic programming. In: Miller, J.F. (ed.) Cartesian Genetic Programming, pp. 17–34. Springer (2011)
9. Rothlauf, F.: Representations for evolutionary and genetic algorithms. Springer (2006)
10. Williams, J.R.: A novel representation for search-based model-driven engineering. Ph.D. thesis, University of York (2013)