Metaheurísticas como soporte a la selección de requisitos del software

Isabel M. del Águila, José del Sagrado, and Francisco J. Orellana

Dpto. Lenguajes y Computación,
Ctra Sacramento s/n, 04120 Universidad de Almería, Espańa
{imaguila,jsagrado,fjorella}@ual.es

Resumen Las técnicas de optimización y metaheurísticas han sido aplicadas ampliamente en numerosas áreas, entre ellas la Ingeniería del Software. En este trabajo mostramos la incorporación de estas técnicas como soporte a las tareas de selección de un grupo de requisitos de entre aquellos que han sido propuestos por los clientes, validando experimentalmente sus resultados. Los algoritmos metaheurísticos son ejecutados desde una herramienta web que permite la definición colaborativa de los requisitos de un proyecto software y ayudan a los desarrolladores durante la ejecución del mismo.

Keywords: gestión de requisitos, ingeniería de requisitos asistida por computadora, optimización metaheurística

1. Introducción

La aplicación de técnicas de Inteligencia Artificial (IA) en los procesos de desarrollo de software ha obtenido buenos resultados en aquellas actividades que necesitan utilizar conocimiento experto. La etapa relativa a los requisitos de un proyecto software es una buena candidata a la aplicación de estas técnicas, debido a la naturaleza de los requisitos, que tienden a ser imprecisos, incompletos y ambiguos [10]. Además, numerosos estudios estadísticos, como los informes CHAOS publicados desde 1.994, indican que las tareas relacionadas con los requisitos son la principal causa de desastre de la puesta en valor del software [16,1,20]. Así, la utilización de técnicas de IA en la mejora del flujo de trabajo de los requisitos, afectará favorablemente a todo el ciclo de vida del desarrollo de software.

En este trabajo mostramos como aplicar con éxito algunas de estas técnicas a tareas relacionadas con los requisitos, en concreto la priorización y selección de requisitos, utilizando técnicas de optimización combinatoria. Esta solución instancia un patrón arquitectónico que permite la integración sin costuras de procesos de Ingeniería de Requisitos con técnicas de AI sobre la herramienta CARE (Computer Aided Requirement Engineering) InSCo-Requisite [25]. Esta herramienta de carácter académico ha sido desarrollada por el grupo de investigación (DKSE – Ingeniería de Datos, del Conocimiento y del Software) en la

A. Ruíz, L. Iribarne (Eds.): JISBD'2012, pp. 287-297, ISBN:978-84-15487-28-9.
Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.

Universidad de Almería proporcionando al grupo de desarrolladores y usuarios un entorno en el que trabajar de forma colaborativa en la definición de requisitos [21].

El resto del artículo se estructura como sigue. La sección 2 enmarca el trabajo en el ámbito de los procesos de Ingeniería de Requisitos (InRe). El problema de selección de requisitos se formaliza en la sección 3, para mostrar los enfoques de solución en la sección 4 donde también se resumen los resultados experimentales de la aplicación de las metaheurísticas. El resultado de su incorporación en InSCo-Requisite se muestra en la sección 5 para finalmente, en la sección 6, presentar las conclusiones.

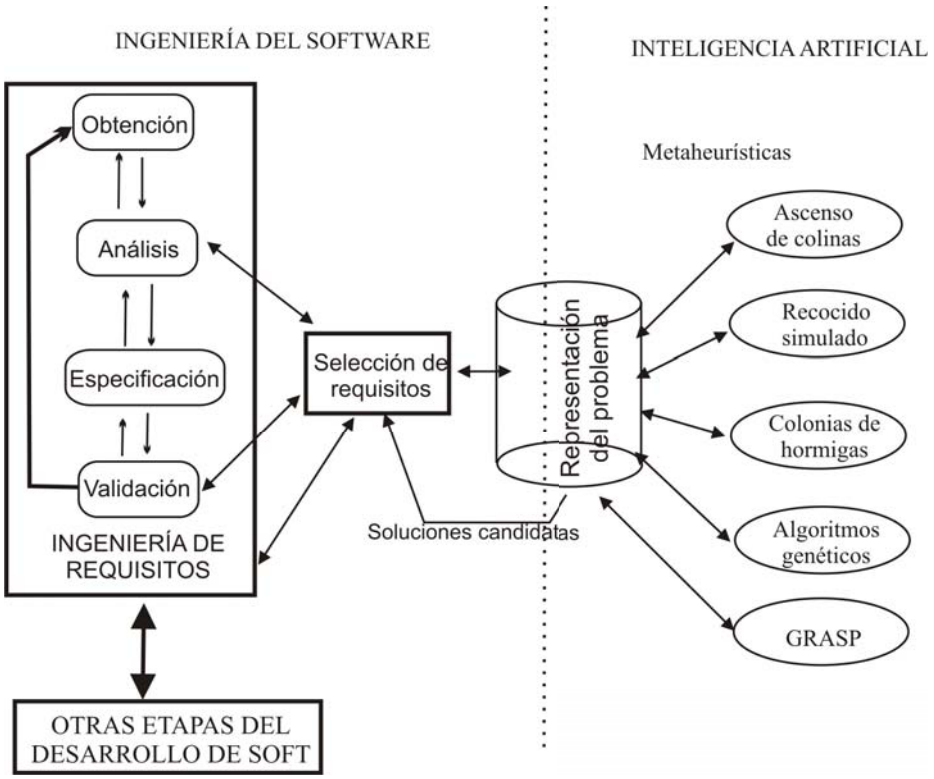


Figura 1. Sinergia entre InRe e IA

2. Visión general de la Ingeniería de Requisitos

La figura 1 muestra la versión más simple del flujo de trabajo de los requisitos [3]. Los requisitos se obtienen de los usuarios o cualquier persona implicada

en la definición del proyecto. Después se estudian y analizan para detectar inconsistencias o incompletitudes, generando una especificación en un determinado lenguaje. Las herramientas CARE ofrecen el entorno para utilizar bases de datos de requisitos, dando soporte a las tareas de especificación de requisitos en formato electrónico y permitiendo la gestión efectiva del proyecto. La validación permite comprobar los requisitos dando integridad al resultado de esta etapa del desarrollo. El ciclo elicitación-análisis-validación se ejecuta durante tantas iteraciones como sea necesario hasta que la especificación de requisitos (SRS-Software Requirement Specification) [2], se haya completado.

La selección de requisitos, de entre los que se hayan definido en los procesos de la InRe, es una tarea importante dentro de esta etapa. Puede aparecer durante el análisis, en las labores de negociación con los clientes o al final del ciclo de requisitos para fijar el alcance del producto a construir. Pero si se utilizan métodos de desarrollo iterativos, en especial los métodos ágiles [7,26], la selección de requisitos es una tarea básica que se retoma a inicio de cada iteración; en este punto, sin opción a cambio durante la ejecución de la iteración, se fijan los requisitos funcionales que se abordan en ese ciclo ágil. Por lo tanto la selección de requisitos adquiere una mayor relevancia dentro de todo el ciclo de desarrollo de software, ya que está presente al principio de cada iteración. Este problema es un proceso de toma de decisiones que utiliza conocimiento experto; el modelado e incorporación de este conocimiento a una herramienta CARE supone un avance importante en el desarrollo de software en general. El hecho de poder hacer cambios fácilmente en InSCo-Requisite es una oportunidad excepcional de poner en marcha la integración de la Ingeniería de los Requisitos (InRe) con las técnicas de IA, haciendo que ambas disciplinas mantengan su independencia [25].

3. Formalización del problema de selección de requisitos

Extrapolando el vocabulario médico, la selección de un conjunto de requisitos, se puede llamar triaje de requisitos [11], y se define como: *el proceso de determinar que requisitos debe satisfacer un producto software según los recursos y el tiempo disponible.* El software, debido a su naturaleza lógica es un complejo producto industrial, y suele venir definido por un gran número de requisitos que lo caracterizan y que en la mayoría de los casos no pueden completarse dentro de las restricciones definidas por tiempo y recursos, y que, por tanto, deben limitarse de alguna manera [8]. La solución es priorizar estos requisitos candidatos y seleccionarlos según los recursos disponibles. Debido al alto número de posibles soluciones, este problema se ha formulado y resuelto empleando técnicas de búsqueda [5,6,24,13,15,18,28,27]

A continuación proponemos la formulación completa de este problema ya que, tal como se muestra en la figura 1, será el punto de conexión entre las técnicas de búsqueda y las herramientas CARE. InSCo-Requisite trasformará el modelo de los requisitos que guarda en su base de datos en una representación que puedan manejar los algoritmos de búsqueda, y a su vez, estos algoritmos como salida, marcarán los requisitos que formarán parte de su solución.

Sea $R = \{r_1, r_2, \ldots, r_n\}$ el conjunto de requisitos propuestos por un conjunto de clientes $C = \{c_1, c_2, \ldots, c_m\}$. Los clientes tienen asignado un peso w_i que mide su importancia dentro del proyecto. Cada requisito $r_j \in R$ tienen asociado un esfuerzo de desarrollo que representa los recursos necesarios para incluir ese requisito en el producto software final, $E = \{e_1, e_2, \ldots, e_n\}$.

Cada $r_j \in R$ puede ser sugerido por más de un cliente y cada cliente c_i asigna un valor v_{ij} al requisito que representa la importancia que para él tiene la inclusión de ese requisito. Es decir, el valor representa la importancia del requisito r_j para el cliente c_i. Estos valores forman una matriz $m \times n$. La satisfacción global, s_j, o el valor añadido de la inclusión del requisito r_j en el producto software, se puede calcular como la suma ponderada de los valores de importancia para cada cliente, $s_j = \sum_i^m w_i v_{ij}$. Mayores valores de v_{ij}, implican una mayor prioridad de ese requisito para el cliente c_i. Un valor cero representa que ese cliente no toma en consideración ese requisito.

Habitualmente los requisitos están interrelacionados o presentan dependencias, que hacen que tengan que abordase unos antes que otros o que sean excluyentes. Las interacciones entre los requisitos representan restricciones al problema y se agrupan en dos tipos: dependencias funcionales o estructurales que representan relaciones semánticas que no pueden ser ignoradas ni relegadas a un tratamiento posterior en ningún caso, y dependencias por recursos consumidos, que suponen reajustar los valores de esfuerzo o satisfacción por la presencia conjunta o no de dos o más requisitos durante el proceso de selección. Aunando las dos grandes aproximaciones a la definición de las dependencias funcionales [9,4] podemos definirlas como:

- *Implicación.* (r_i *implica* r_j). El requisito r_i no puede ser seleccionado si el requisito r_j no ha sido implementado previamente.
- *Combinación.* (r_i *acoplado-con* r_j). El requisito r_i no puede ser incluido separadamente del r_j.
- *Exclusión.* (r_i *excluye-a* r_j). El requisito r_i no puede incluirse junto al requisito r_j.

De esta forma el objetivo del problema será encontrar el subconjunto de requisitos $\hat{R}$ de entre todos los subconjuntos de R, $\hat{R} \in \wp(R)$, que sea la mejor solución. La calidad de la solución se mide respecto a uno o varios objetivos utilizando una función de evaluación previamente fijada. Generalmente, en el dominio del problema de la selección de requisitos, se persigue construir $\hat{R}$ de forma que satisfaga en la mayor medida las peticiones de los clientes dentro de las limitaciones ofrecidas por los recursos del proyecto y las interacciones entre los requisitos. En términos de la formulación realizada para el problema esto lo conseguimos maximizando la satisfacción y dentro de los límites de esfuerzo. La satisfacción y el esfuerzo del conjunto $\hat{R}$ son respectivamente:

$$\text{sat}(\hat{R}) = \sum_{j \in \hat{R}} (s_j), \quad \text{eff}(\hat{R}) = \sum_{j \in \hat{R}} (e_j) \tag{1}$$

donde j representa al requisito r_j. Siendo B el límite de esfuerzo. Formalmente,

$$\begin{aligned} &\text{maximize } \text{sat}(\hat{R}) \\ &\text{subject to } \text{eff}(\hat{R}) \leq B \end{aligned} \tag{2}$$

4. Solución basada en metaheurísticas

Las técnicas de optimización y metaheurísticas han sido ampliamente aplicadas en numerosas áreas de la ingeniería y como no podría ser de otra forma la ingeniería del software es una de ellas. Esta sinergia está siendo explotada por una corriente de investigación conocida como Ingeniería del Software Basada en Búsqueda (Search Based Software Engineering) [19]. En concreto en relación con la especificación y los requisitos nos encontramos con un gran número de trabajos que se centran el problema de la selección de los requisitos para la siguiente versión del software (Next Release Problem, NRP [5]). Un resumen de las más significativas se muestra en la tabla 1

Tabla 1. Trabajos que solucionan NRP

	Con interacciones	Sin interacciones
Recocido simulado	Bagnall et al. [5]	Baker et al. [6], del Sagrado et al. [24]
Alg. Voraz (Greedy)	Bagnall et al. [5]	Baker et al. [6]
GRASP	del Sagrado et al. [23]	
GA	Greer & Ruhe [18]	del Sagrado et al. [24]
NSGA-II	Zhang & Harman [27]	Durillo et al. [13], Finkelstein et al. [15], Zhang et al. [28]
MOCell		Durillo et al. [13]
ACS	del Sagrado et al. [23]	del Sagrado et al. [24]

Las técnicas metaheurísticas utilizadas para resolver NRP son muy diversas. Bagnall et al. [5] utilizó en su resolución el ascenso de colinas, una implementación de algoritmos voraces y la técnica del recocido simulado. El trabajo de Baker et al. [6] demostró la aplicabilidad de estas técnicas en casos reales mejorando la efectividad de los expertos. Greer y Ruhe [18], abordaron el problema como un problema de planificación, asignando los requisitos a incrementos, considerando las valoraciones de los clientes y las limitaciones de recursos. Como base para la optimización se utilizó un algoritmo genético. Del Sagrado et al. [24] adaptaron un sistema de colonia de hormigas a NRP, comparando sus soluciones con un algoritmo genético y el recocido simulado. NRP también ha sido tratado como un problema multiobjetivo, aplicando técnicas como NSGA−II, MoCell [28,13,15].

Sólo algunos de estos trabajos han tratado las interacciones entre los requisitos [5,18,27,23]. Bagnall et al. [5] sólo tiene en cuenta las dependencias de impliación y selecciona clientes, con todas sus propuestas, no selecciona requisitos. Greer y Ruhe [18] definen incrementos utilizando las interacciones de implicación y

combinación. Un estudio de cómo afectan las dependencias a los algoritmos genéticos aparece en [27]. El primer trabajo en donde se presentó una representación explicita de los requisitos mediante un grafo de interacciones es [23].

Una metaheuristica ofrece un estrategia para el diseño de heurísticas que den solución a un problema de optimización. Una de las principales ventajas es que permiten obtener soluciones cercanas a las óptimas empleando tiempos de ejecución razonables, gracias principalmente a una exploración más eficiente del espacio de búsqueda. La aplicación de una técnica metaheurística necesita, además de la representación del problema ya descrita, la definición de la estrategia de búsqueda de la solución, específica para cada técnica.

En este trabajo hemos seleccionado tres técnicas para resolver NRP: greedy randomized adaptive search procedure (GRASP), algoritmos genéticos y un sistema de colonia de hormigas (ACS), mostrando su adaptación al problema y una breve comparativa de los resultados que ofrecen sobre dos conjuntos de datos.

GRASP [14], aplica un esquema iterativo, primero construye una solución voraz aleatoria que se mejora con búsqueda local. Se construye una lista de elementos ordenados en base a una función voraz y se realiza una selección aleatoria. La lista se asegura de cumplir las restricciones del problema en cuanto a interacciones y límites de recursos. La función voraz mide la productividad del requisito.

$$g(r_i) = \frac{s_i}{e_i} \tag{3}$$

La búsqueda local intenta mejorar la solución actual buscando en el vecindario. Así, cosiderando las restricciones del problema, intenta sustituir cada requisito por otro que mejore la función de evaluación de la solución, $sat(R')$. El algoritmo acaba cuando se han explorado todos los requisitos de la solución.

Un algoritmo genético emula la evolución natural de una población o conjunto de individuos [17]. Durante una generación cada nueva población se construye aplicando operadores de selección, cruce y mutación sobre los individuos. Los individuos representan posibles soluciones, es decir, un conjunto de requisitos que satisfacen NRP. Los mejores individuos son los que tienen más posibilidades de ser elegidos como padres. Así un individuo $R_k \in \wp(R)$, de la población actual Q, es elegido como padre con una probabilidad:

$$p_{R_i} = \frac{sat(R_i)}{\sum_{R' \in Q} sat(R')} \tag{4}$$

La optimización por colonia de hormigas (Ant Colony Optimization, ACO) es una técnica de búsqueda que emula el comportamiento de las hormigas cuando busca el camino más corto desde su hormiguero hasta la comida [12]. Este proceso natural utiliza una sustancia química llamada feromona que segregan las hormigas conforme avanzan. Al seleccionar un camino una hormiga adopta un comportamiento probabilístico. En cada encrucijada, tiende a seleccionar el camino que tiene más feromona, si la cantidad de esta sustancia no es suficiente

para hacer una elección se hace de forma aleatoria. Conforme pasa el tiempo, los caminos más frecuentados tendrán una mayor concentración de feromona.

Los pasos de la ejecución de este tipo de algoritmos son:

- Inicialización del rastro de feromona. Representado por una matriz que recoge la cantidad de feromona inicial que se actualizará durante la ejecución del algoritmo.
- Selección aleatoria del punto de inicio de cada hormiga.
- En cada etapa la hormiga localiza los nodos visibles, en función de las restricciones del problema, y elige uno combinando la información heurística y la feromona.
- Actualización de la matriz de feromona de la mejor solución. La feromona de un camino (arco) del nodo i al nodo j se denota τ_{ij}.

Cuando se construye la solución de la hormiga k, la regla utilizada para seleccionar desde el nodo i el nodo j se define como:

$$j = \begin{cases} argmax_{u \in N_i^k}[[\tau_{ij}]^\alpha [\eta_{ij}]^\beta], \text{ if } q \leq q_0, \\ u, \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{5}$$

donde q es un valor aleatorio entre $[0, 1]$, y $q_0 \in [0, 1]$, con $(q \leq q_0)$ es un parámetro que controla el balanceo entre intensificación y diversificación en el algoritmo, buscando la mejor cobertura del espacio de búsqueda. La probabilidad con que la hormiga k estando en el nodo i seleccione moverse al j es:

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, \text{ if } j \in N_i^k, \\ 0, \qquad\qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{6}$$

siendo la información heurística una media de productividad $\eta_{ij}^k = \mu(s_j/e_j($ con μ un valor de normalización y N_i^k el conjunto de nodos visibles para la hormiga k según las restricciones del problema. La mejor solución encontrada por las hormigas, $\hat{R}$, actualiza el rastro de feromona de acuerdo con:

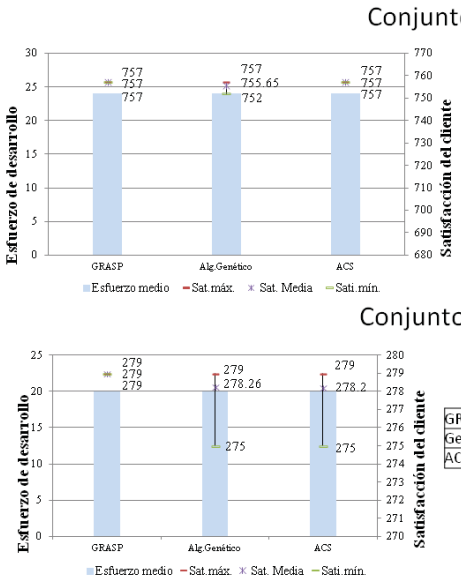
$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \rho \Delta \tau_{ij} \tag{7}$$

siendo

$$\Delta \tau_{ij} = \frac{sat(\hat{R})}{sat(R)} \tag{8}$$

y $\rho \in [0, 1]$ la tasa de evaporación de la feromona,

Estos algoritmos han sido probados empleando dos conjuntos de datos, el primero de 20 requisitos y 5 clientes con $B = 25$, usado por primera vez en [6] y que ha sido ampliamente utilizado para evaluar este tipo soluciones al problema de selección de requisitos [27,13,23], y un segundo conjunto de datos sintético de 100 requisitos, 5 clientes y con $B = 20$. Sobre cada algoritmo se han realizado 100 ejecuciones independientes y sus resultados se muestran en las tablas. Comprobamos que en todos los casos se obtiene el mayor esfuerzo, y que todos los algoritmos alcanza un en algún caso el valor máximo de satisfacción.



	Satisfacción	Esfuerzo	Tiempo (ms)
GRASP	757±0	24	94.14±6.84
Genético	755.65±2.23	24	201.32±33.93
ACS	757±0	24	232.84±36.21

	Satisfacción	Esfuerzo	Tiempo (ms)
GRASP	279±0	20	1702.53±89.6
Genético	278.26±1.01	20	9744.1±229.9
ACS	278.2±1.13	20	20670.24±6235.6

Figura 2. Resultados de las metaherurísticas

Las variaciones del algoritmo genético y la colonia de hormigas, indican no se ha obtenido la convergencia deseada en algunos casos, (en especial en el conjunto de datos de mayor número de requisitos), pero como conclusión general de la experimentación se comprueba que con los tres métodos metaheurísticos se pueden obtener soluciones válidas para los ingenieros del software.

5. Selección de requisitos en InSCo-Requisite

En este y otros trabajos se ha mostrado de forma experimental la validez de las técnicas metaheurísticas en la solución del problema de selección de requisitos. Sin embargo la validez de estas técnicas está ligada a su utilización real insertadas en herramientas necesarias para el desarrollador de software y facilitan el trabajo cotidiano de estos profesionales tal como puso de manifiesto Ian Sommerville en las sesiones invitadas del primer simposio internacional SBSE [22]. El valor real de estos enfoques es combinar la inteligencia computacional con la experiencia de los expertos, en este caso ingenieros del software, para obtener mejores resultados del que obtendrían sólo los expertos. Es decir, GRASP, algoritmos genéticos y ACS, tienen que embeberse dentro de las herramientas software empleadas en el proyecto.

InSCo-Requisite facilita la obtención de los requisitos de los clientes [21] en una serie de proyectos. En cada proyecto participan un conjunto de clientes

dados previamente de alta en el sistema; al asignarlos a un proyecto se les fija un peso w_j, que puede variar entre proyectos, dependiendo de la importancia de ese usuario en cada proyecto concreto.

Esta aplicación web ofrece un conjunto de plantillas que permiten definir *requisitos funcionales, no funcionales y de información*, estos últimos son aquellos relativos a los datos manejados por el software a construir. Estos tres tipos de requisitos son los de más bajo nivel, ya que InSCo-Requisite mantiene una estructura jerárquica que permite manejar diversos niveles de abstracción. Así, se representan de forma separada los *objetivos*, relacionados con el nivel de negocio que se refinan hasta llegar a *requisitos funcionales*, y lo mismo con los *conceptos del dominio* y los *requisitos de información*. Los requisitos funcionales se describen empleando escenarios y es en este nivel donde todos los usuarios pueden optar por asignar un valor que representa la importancia de este requisito desde su punto de vista, v_{ij}.

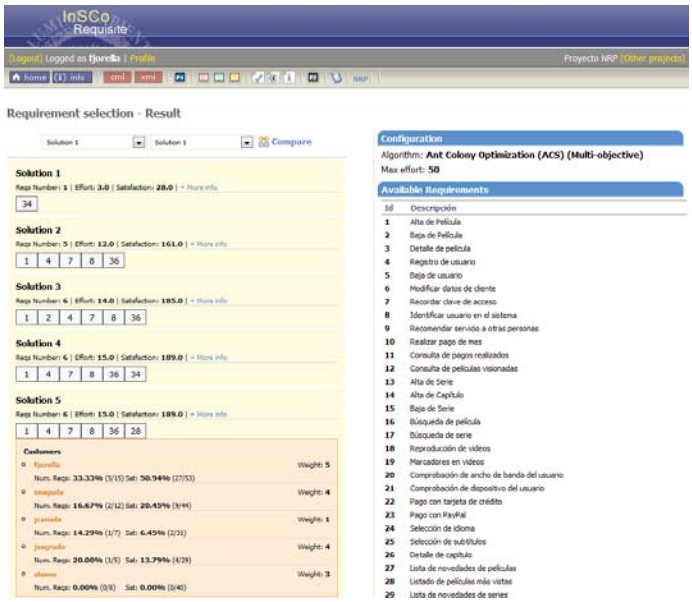


Figura 3. Interfaz de InSCo-Requisite

InSCo-Requisite trasforma la base de datos de requisitos en una representación manejable por los algoritmos de búsqueda. A partir de aquí se lanzan los algoritmos metaheurísticos, bien con los parámetros por defecto o con los que el usuario ajuste (como el número de hormigas, tamaño de la población o peso de la búsqueda local en GRASP).

Los requisitos seleccionados por el algoritmo se muestran al usuario incluyendo información sobre los recursos totales consumidos y la satisfacción. Además tal

como muestra la figura 3 los hiperenlaces permiten moverse entre la lista de requisitos seleccionados para facilitar la decisión final del experto.

6. Conclusiones

Dentro del ámbito de la ingeniería de requisitos se han definido y utilizado tres algoritmos de metaheurísticos de búsqueda, que se han incorporado dentro de una herramienta de definición y modelado de requisitos. Esta funcionalidad de InSCo-Requisite permite a los desarrolladores obtener de entre los requisitos candidatos obtenidos de los clientes la lista de aquellos que cumplen con los criterios definidos en el problema de búsqueda.

Los resultados experimentales sobre dos problemas tipo, permiten afirmar que GRAPS, los algoritmos genéticos y ACS son metaheurísticas aplicables al problema con las adaptaciones y funciones de evaluación definidas. Por otra parte, la puesta en valor de los algoritmos dentro de InSCo-Requisite, pone en práctica una arquitectura que facilita las mejoras en paralelo, tanto en las técnicas de IA como de las técnicas de modelado y gestión de requisitos.

Como trabajo futuro planteamos la mejora de las técnicas metaheurísticas, incorporando una visión multiobjetivo del problema y la mejora en la integración con el resto de etapas del desarrollo de un proyecto software así como con otras herramientas software similares.

Agradecimientos. Este trabajo se ha llevado a cabo en el marco de los proyectos del Ministerio de Educación, Cultura y Deportes TIN2010-20900-C04-02 y proyecto PIO-TEP-6174 de la Consejería de Economía, Inovación y Ciencia de la Junta of Andalucía.

Referencias

1. Chaos report: the state of the software industry. Tech. rep., Standish Group International (1994)
2. IEEE recommended practice for software requirements specifications. IEEE Std 830-1998 p. i (1998)
3. Abran, A., Moore, J., Bourque, P., Dupuis, R., Tripp, L.: Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, Los Alamitos (2004)
4. van den Akker, M., Brinkkemper, S., Diepen, G., Versendaal, J.: Software product release planning through optimization and what-if analysis. Information and Software Technology 50(1-2), 101111 (2008)
5. Bagnall, A.J., Rayward-Smith, V.J., Whittley, I.: The next release problem. Information & Software Technology 43(14), 883–890 (2001)
6. Baker, P., Harman, M., Steinhöfel, K., Skaliotis, A.: Search based approaches to component selection and prioritization for the next release problem. In: ICSM. pp. 176–185. IEEE Computer Society (2006)
7. Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Jeff, S.: The agile manifiesto. Tech. rep., Agile Alliance (2001)

8. Berander, P., Svahnberg, M.: Evaluating two ways of calculating priorities in requirements hierarchies - an experiment on hierarchical cumulative voting. Journal of Systems and Software 82, 836–850 (May 2009)
9. Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., och Dag, J.N.: An industrial survey of requirements interdependencies in software product release plannin. In: RE. pp. 84–93. IEEE Computer Society (2001)
10. Cheng, B.H.C., Atlee, J.M.: Research directions in requirements engineering. In: Briand, L.C. Wolf, A. (ed.) FOSE. pp. 285–303 (2007)
11. Davis, A.M.: The art of requirements triage. IEEE Computer 36(3), 42–49 (2003)
12. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 26(1), 29–41 (feb 1996)
13. Durillo, J.J., Zhang, Y., Alba, E., Harman, M., Nebro, A.J.: A study of the bi-objective next release problem. Empirical Software Engineering 16(1), 29–60 (2011)
14. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
15. Finkelstein, A., Harman, M., Mansouri, S.A., Ren, J., Zhang, Y.: A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. Requir. Eng. 14(4), 231–245 (2009)
16. Glass, R.L.: Facts and Fallacies of Software Engineering. Addison Wesley (2002)
17. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine learning. Addison Wesley (1989)
18. Greer, D., Ruhe, G.: Software release planning: an evolutionary and iterative approach. Information & Software Technology 46(4), 243–253 (2004)
19. Harman, M.: The current state and future of search based software engineering. In: 2007 Future of Software Engineering. pp. 342–357. FOSE '07, IEEE Computer Society, Washington, DC, USA (2007)
20. Johnson, J.: CHAOS chronicles v3.0. Tech. rep. (2003), `http://standishgroup.com/chaos/toc.php`
21. Orellana, F.J., Cañadas, J., del Águila, I.M., Túnez, S.: Insco requisite - a web-based rm-tool to support hybrid software development. In: Cordeiro, J., Filipe, J. (eds.) ICEIS 2008 (3-1). pp. 326–329 (2008)
22. Penta, M.D., Poulding, S. (eds.): Search Based Software Engineering, 2009 1st International Symposium on. IEEE (may 2009)
23. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Requirements interaction in the next release problem. In: Krasnogor, N., Lanzi, P.L. (eds.) GECCO (Companion). pp. 241–242. ACM (2011)
24. del Sagrado, J., del Águila, I., Orellana, F.: Ant colony optimization for the next release problem: A comparative study. In: Search Based Software Engineering (SSBSE), 2010 Second International Symposium on. pp. 67 –76 (sept 2010)
25. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Architecture for the use of synergies between knowledge engineering and requirements engineering. Lecture Notes in Computer Science 7023, 213–222 (2011)
26. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall, illustrated edition edn. (Oct 2001)
27. Zhang, Y., Harman, M.: Search based optimization of requirements interaction management. In: Search Based Software Engineering (SSBSE), 2010 Second International Symposium on. pp. 47 –56 (sept 2010)
28. Zhang, Y., Harman, M., Mansouri, S.A.: The multi-objective next release problem. In: Lipson, H. (ed.) GECCO. pp. 1129–1137. ACM (2007)

A. Ruíz, L. Iribarne (Eds.): Actas de las "*XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'2012)*", Jornadas SISTEDES'2012, Almería 17-19 sept. 2012, Universidad de Almería.