Una Revisión Empírica de Heurísticas de Verificación de Casos de Uso basadas en Métricas*

Beatriz Bernárdez, Amador Durán y Miguel Toro

Departamento de Lenguajes y Sistemas Informáticos
E.T.S. Ingeniería Informática, Universidad de Sevilla
Avda. Reina Mercedes s/n 41012 Sevilla, España
e-mail: {beat,amador,mtoro}@lsi.us.es

Resumen En este artículo se presenta una revisión de varias de las heurísticas de verificación de requisitos basadas en métricas desarrolladas por el Grupo de Ingeniería del Software de la Universidad de Sevilla. Esta revisión se basa en los datos empíricos obtenidos de prácticas de alumnos de segundo ciclo de la Ingeniería Informática de la Universidad de Sevilla. El análisis de los datos empíricos ha permitido confirmar la validez de las heurísticas previamente propuestas así como afinar algunos de sus parámetros, mejorando de esta forma su capacidad de predecir la presencia de defectos en los requisitos. Uno de los aspectos más interesantes obtenidos mediante el análisis empírico ha sido la detección de posibles mejoras en el modelo de casos de uso en el que estaban basadas las heurísticas iniciales, proporcionando una importante realimentación a la investigación inicial.

Palabras clave: Ingeniería de Requisitos, Verificación de Requisitos, Ingeniería del Software Empírica

1. Introducción

La importancia de iniciar el aseguramiento de la calidad del software (SQA, *Software Quality Assurance*) desde las primeras fases del desarrollo de software es algo sobradamente conocido y comprobado dentro de la comunidad de ingeniería del software. De forma tradicional, la verificación y validación de requisitos han sido las únicas actividades consideradas parte del SQA en la fase de ingeniería de requisitos (IR). Una visión más amplia de la calidad en la IR, como la presentada en [9], incluye al análisis de requisitos dentro del grupo de actividades de SQA, ya que su objetivo es también mejorar la calidad de los requisitos mediante la detección de conflictos. En el diagrama de actividades UML [3] de la figura 1 se muestra cómo se integran estas tres actividades en el proceso general de IR, resaltando en qué actividades intervienen clientes y usuarios (C&U) y personal de SQA.

En esta visión más amplia de la SQA en la IR, se puede considerar al análisis de requisitos como la actividad en la que se analizan los requisitos previamente elicitados y documentados para detectar posibles conflictos, normalmente mediante la construcción

* Este trabajo está financiado parcialmente por el proyecto CICYT TIC 2000–1106–C02–01 (GEOZOCO) y por el proyecto PCB–02–001 (TAMANSI).

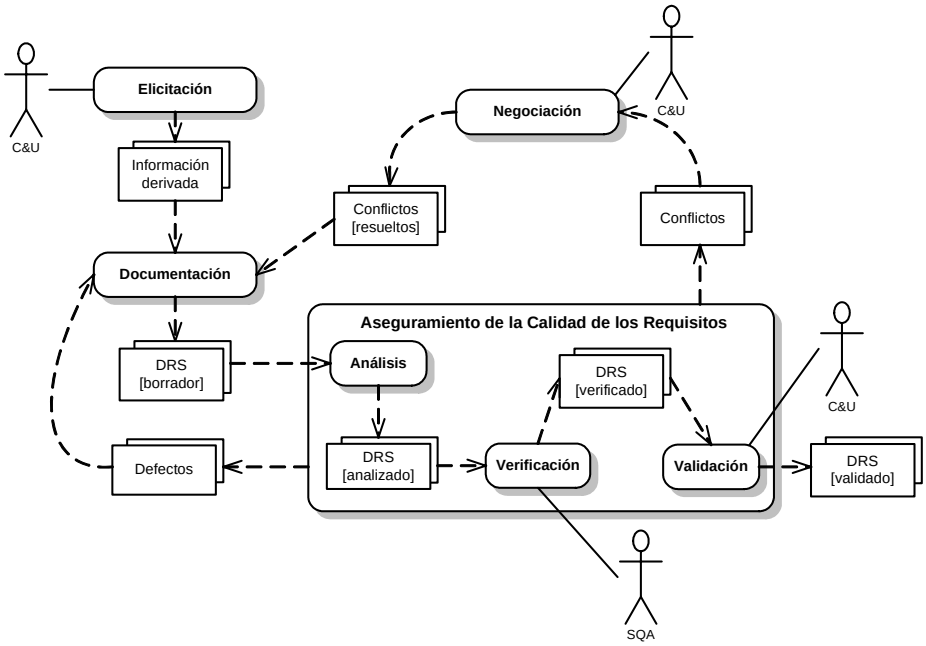

Figura 1. Modelo de procesos de ingeniería de requisitos (simplificado)

de modelos conceptuales que permiten profundizar en el conocimiento del problema y que serán la base del diseño [7]. La verificación puede considerarse como la actividad cuyo objetivo es alcanzar la *calidad interna* [13] exigible a las especificación de requisitos conforme a las normas establecidas por la organización previamente, mientras que la validación tiene como objetivo es asegurar que los requisitos elicitados, documentados, analizados y verificados representan realmente las necesidades de clientes y usuarios [18].

Dentro del contexto de la calidad en la IR, en este artículo se presenta una revisión de varias de las heurísticas de verificación de requisitos basadas en métricas descritas en [8,9] a partir de los datos obtenidos de prácticas de alumnos de la E.T.S. de Ingeniería Informática. La estructura del mismo es la siguiente: en la siguiente sección se describe brevemente la actividad de verificación de requisitos. En la sección 3 se describen las métricas de casos de uso en las que están basadas las heurísticas revisadas. En la sección 4 se revisan 5 de las 10 heurísticas presentadas en [9]. En la sección 5 se comentan algunos trabajos relacionados y, por último, en la sección 6 se exponen las conclusiones y las posibles líneas de trabajo futuro.

2. Verificación de requisitos

Para llevar a cabo la verificación de la calidad de los requisitos se han propuesto diversas técnicas de detección de defectos, principalmente técnicas de lectura. Las di-

ferencias entre estas técnicas residen en el número de participantes, el centrarse o no en un tipo específico de defectos, el tipo y número de reuniones si las hay, etc. Ejemplos de algunas de estas técnicas y de su efectividad pueden consultarse en [1,19,21].

Un concepto importante en la verificación de requisitos es el de *modelo de calidad*. Un modelo de calidad de requisitos consiste en una lista de propiedades deseables que deben tener los requisitos y cuya ausencia se considera un *defecto*. El objetivo básico de la verificación es detectar la mayor cantidad posible de defectos de acuerdo a un modelo de calidad previamente establecido.

El modelo de calidad en el que se basan las heurísticas descritas en [9] está basado en la lista de propiedades de Davis [6], en la que se incluyen propiedades como la no ambigüedad, la comprensibilidad, etc. Para la revisión de las heurísticas, dicho modelo ha sido complementado con la lista de comprobación para casos de uso de Lilly [15], en la que se comprueban propiedades específicas de los casos de uso como el buen uso de las estructuras *«includes»* y *«extends»*, la independencia del diseño, etc.

Las heurísticas revisadas en este artículo tienen como objetivo identificar requisitos que potencialmente puedan contener defectos, basándose para ello en los valores de ciertas métricas que pueden calcularse de forma automática mediante herramientas de gestión de requisitos como REM [8,9]. Si el valor de estas métricas está fuera del rango habitual, un requisito se considera como potencialmente defectuoso y debe ser revisado.

3. Métricas de casos de uso

Las métricas de casos de uso en las que se basan las heurísticas revisadas en este artículo, y que se muestran en la tabla 1, están basadas en el modelo de casos de uso de

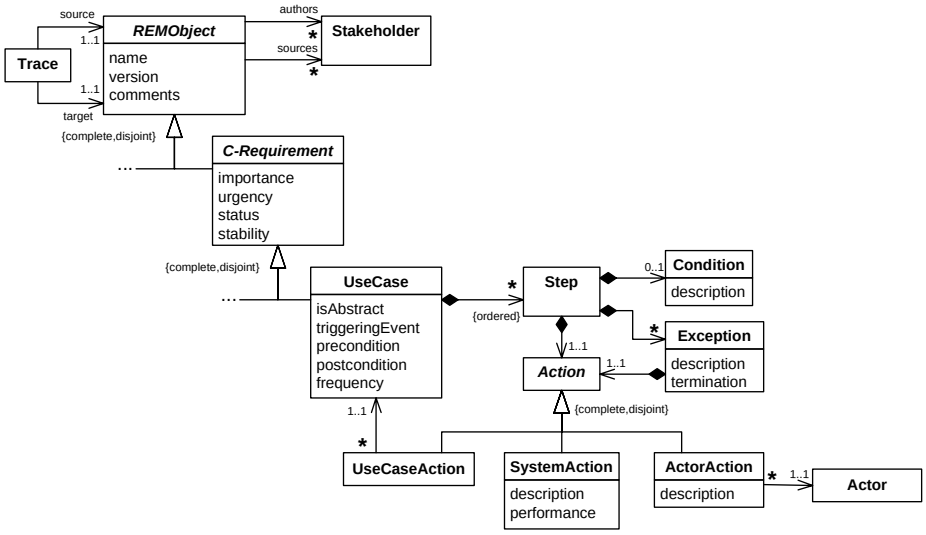

Figura 2. Modelo de casos de uso de REM

Métrica	Descripción
NOS	Número de pasos del caso de uso
NOAS	Número de pasos de actor del caso de uso
NOSS	Número de pasos de acción de sistema del caso de uso
NOUS	Número de pasos que activan otro caso de uso del caso de uso
NOCS	Número de pasos condicionales
NOE	Número de excepciones del caso de uso
NOAS/NOS	Porcentaje de pasos de actor del caso de uso
NOSS/NOS	Porcentaje de pasos de sistema del caso de uso
NOUS/NOS	Porcentaje de pasos de caso de uso del caso de uso
CC	Complejidad ciclomática del caso de uso (NOCS+NOE+1)

Tabla 1. Métricas de casos de uso usadas en las heurísticas revisadas

REM [9] (ver figura 2). En este modelo, un caso de uso se considera como una secuencia de pasos que pueden ser de acción de actor, de acción del sistema o de realización de otro caso de uso (una inclusión si el paso no es condicional o una extensión en caso contrario).

Las heurísticas que se revisan en la siguiente sección están basadas en la experiencia de varios de los autores en la verificación de prácticas de alumnos y en su intuición de que existe un intervalo de valores para ciertas métricas fuera del cual la probabilidad de contener defectos de un caso de uso aumenta. La hipótesis inicial que se pretende validar empíricamente es que dada una métrica de caso de uso m, existe un intervalo $[m_1, m_2]$ tal que dado un caso de uso c, se cumple que:

$$m(c) \notin [m_1, m_2] \Rightarrow P_{def}[c] \gg 1 - P_{def}[c]$$

donde $P_{def}[c]$ es la probabilidad de que un caso de uso c contenga defectos y $1-P_{def}[c]$ es la probabilidad de que un caso de uso c no contenga defectos. En esta hipótesis inicial no se hace ninguna afirmación sobre la probabilidad de que un caso de uso presente o no defectos dentro del intervalo considerado habitual para una métrica, ya que podría haber causas no reflejadas por el valor de la métrica que provocaran la presencia de defectos dentro del intervalo. Sólo se afirma que en el caso de que el valor de la métrica esté fuera del intervalo, la probabilidad de que el caso de uso presente defectos es significativamente más alta que la probabilidad de que no lo haga.

Para validar las heurísticas se han analizado 8 especificaciones de requisitos elaboradas por alumnos de cuarto curso de Ingeniería Informática de la Universidad de Sevilla con un total de 127 casos de uso. Estas especificaciones fueron verificadas por varios de los autores de este trabajo para identificar y registrar defectos según en el modelo de calidad comentado previamente en la sección 2. Los resultados obtenidos pueden verse resumidos en la figura 3.

4. Revisión empírica de las heurísticas de verificación de requisitos

En esta sección se revisan las heurísticas basadas en las métricas NOS, NOAS/NOS, NOSS/NOS, NOUS/NOS y CC previamente descritas en [9].

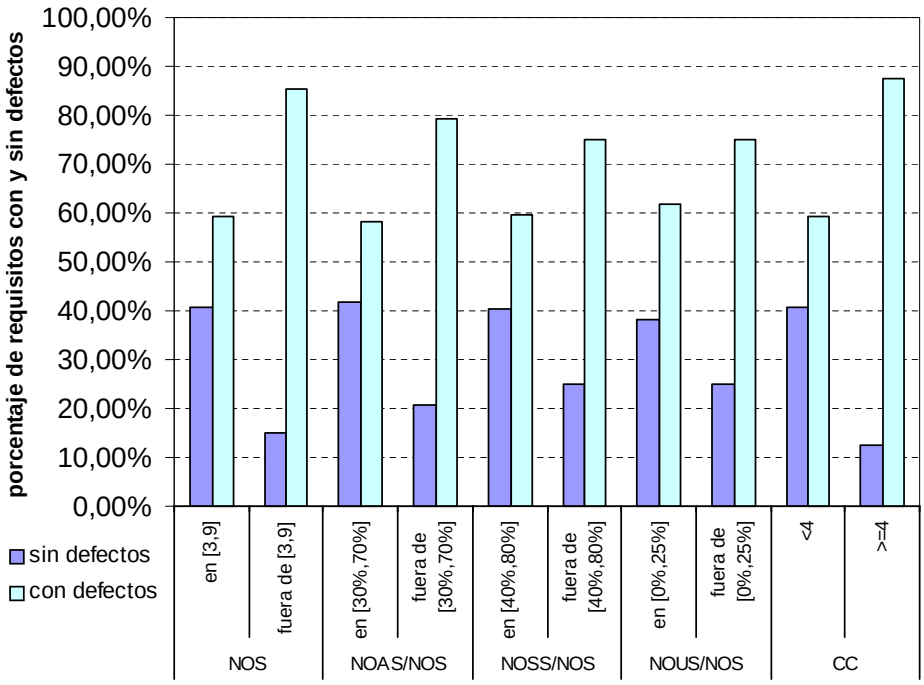

Figura 3. Resultados empíricos

4.1. Revisión de la heurística *Checking the number of steps*

La intuición en la que se basa esta heurística, tal como se describe en [8,9] de acuerdo con otros autores como Cockburn [5], es que los casos de uso demasiado cortos suelen ser incompletos y los demasiado largos son frecuentemente incomprensibles o presentan un nivel de detalle excesivo, por lo que se recomienda que la métrica NOS esté en el intervalo $[3, 9]$.

Como se puede apreciar en la figura 3, los datos empíricos confirman la validez de la heurística, ya que la proporción de casos de uso con defectos fuera del intervalo es del 85 % frente a un 15 % que no los presentan. Mediante un análisis detallado de los datos se ha podido comprobar que los casos de uso con menos de 3 pasos suelen ser incompletos (tal como se intuía originalmente en [9]) o no representan realmente una interacción actor–sistema resultando generalmente triviales. Por otra parte, el hecho de que el caso de uso tenga más de 9 pasos se suele deber a que se haya descrito con demasiado detalle, tal como se comenta en [15], es decir que no sea conciso según el modelo de calidad de Davis [6]. No obstante, también se ha podido apreciar que esta heurística es muy sensible al estilo de redacción, sobre todo si se agrupan varias acciones en un mismo paso o se consideran pasos diferentes, o si se incluyen pasos de interacciones de actores con el entorno sin que el sistema intervenga. Lo más indicado para evitar esta influencia parece ser considerar distintos intervalos en función del estilo de redacción.

4.2. Revisión de la heurística *Checking the rate of different types of steps* (NOAS/NOS)

Esta heurística esta basada en la idea de que un caso de uso sirve básicamente para expresar una interacción actor–sistema. Por ello, el número de pasos de actor y el de pasos de sistema deben estar en torno al 50 %, considerando también la posibilidad de que existan pasos de inclusión o extensión en los que se realice otro caso de uso. El intervalo propuesto originalmente en [9] es $[30\,\%, 70\,\%]$ para NOAS/NOS.

Los datos empíricos (ver figura 3) confirman la heurística ya que el 80 % de casos de uso fuera del intervalo propuesto contienen defectos. El análisis detallado de los datos ha puesto de manifiesto que las situaciones que llevan a esta métrica fuera del intervalo son:

- El hecho de obviar la participación del sistema, por lo que el caso de uso resulta incompleto. Es la situación más habitual.
- El hecho de haber desglosado demasiado las acciones de un actor determinado. En este caso aparecen varios pasos seguidos del mismo actor, lo cual se podría haber evitado uniéndolos en uno solo, separando las acciones por comas en el texto del paso.
- El hecho de incluir interacciones de actores con el entorno del sistema o con otros actores. Este hecho, que no puede considerarse un defecto, sino más bien una información interesante aunque no esencial del caso de uso, pone de manifiesto la necesidad de modificar el modelo de casos de uso de la figura 2, especializando la clase ActorAction en dos nuevas subclases disjuntas, ActorActorAction y ActorSystemAction, de forma que para la heurística sólo se tengan en cuenta los pasos de interacción de actores con el sistema.

4.3. Revisión de la heurística *Checking the rate of different types of steps* (NOSS/NOS)

En el caso de esta heurística, que se basa en el mismo razonamiento que la anterior, los datos también la confirman con un 75 % de casos de uso con defectos fuera del intervalo $[40\,\%, 80\,\%]$, frente a un 25 % sin defectos. En este caso, el análisis detallado de los datos muestra situaciones distintas en función de que el caso de uso sea abstracto (no se realiza por si mismo sino dentro de otros casos de uso) o concreto. Algunos resultados obtenidos son los siguientes:

- En situaciones con NOSS/NOS $> 80\,\%$, el defecto habitual es que se pretende representar un proceso *bacht* que no necesita participación de ningún actor, por lo que no es realmente un caso de uso. En estos casos lo más conveniente es expresar el requisito mediante un requisito funcional redactado en texto plano, sin usar la estructura de caso de uso.
- Algunos casos de uso abstractos con NOSS/NOS por debajo del 40 % no presentan defectos porque, al ser un caso de uso abstracto un *fragmento* de otros casos de uso, puede ocurrir que los pasos de actor estén descritos en los caso de uso que lo activan.

- Si el caso de uso es concreto y se pretende representar una interacción actor–sistema, debe incluirse al menos un paso de actor, el primero, algo que no tiene porqué ser cierto en los casos de uso abstractos. Este resultado permite definir una nueva heurística consistente en detectar aquellos casos de uso concretos cuyo primer paso no sea de actor.

También se han encontrado otras causas que provocan que la métrica quede fuera del intervalo independientemente de si el caso de uso es abstracto o no. Puede ocurrir que abunden los pasos de sistema porque el caso de uso incluya acciones internas del sistema [15], que en teoría no deberían aparecer en el caso de uso, o bien porque se hayan desglosado excesivamente las acciones del sistema, las cuales se podrían unir en un solo paso de sistema separando las acciones por comas.

4.4. Revisión de la heurística *Checking the rate of different types of steps* (NOUS/NOS)

Esta heurística se basa en la idea de que los mecanismos de estructuración de los casos de uso conocidos como *inclusión* y *extensión* [3] deben usarse principalmente como medio para evitar repeticiones de secuencias de pasos en distintos casos de uso. Debido a esto, y a que su uso abusivo hace que las especificaciones de casos de uso sean difíciles de entender, en [9] se planteó $[0\,\%, 25\,\%]$ como intervalo deseable para NOUS/NOS. Tal como se ve en la figura 3, los datos consolidan esta intuición ya que el 75 % de los casos de uso que están fuera del intervalo propuesto presentan defectos.

Las principales causas de estos defectos suelen ser la introducción de aspectos de navegación en los casos de uso, de forma que se crea una especie de caso de uso *menú principal* que es extendido por otros casos de uso en función de la elección del actor principal. Otra causa detectada es la tendencia a organizar los casos de uso como si fueran un programa de ordenador, intentando modularizar en exceso la especificación, con las negativas repercusiones que ello acarrea para los clientes y usuarios no familiarizados con la programación.

4.5. Revisión de la heurística *Checking cyclomatic complexity*

La idea en la que se basa esta heurística es que los casos de uso con abundancia de caminos alternativos suelen ser difíciles de entender y, a veces, hasta ilegibles. Para conocer el número de caminos alternativos se ha definido la complejidad ciclomática (CC) de un caso de uso de la misma forma que se define la CC de McCabe [17] para código fuente: el número de puntos de decisión más 1, es decir el número de pasos condicionales más el número de excepciones más 1 (ver tabla 1). En [9] se propuso como intervalo deseable $[1, 4]$, y los datos empíricos lo confirman con una proporción de casos de uso con defectos del 87,5 %.

Al analizar los casos de uso fuera del intervalo deseable se ha observado que en algunas ocasiones el defecto viene motivado porque se redacta el caso de uso con estilo *pseudocódigo con GOTOs*, con pasos condicionales del tipo *'Si <condición>, el sistema vuelve al paso X'*, con lo que se vuelve prácticamente incomprensible.

En nuestra experiencia con varias empresas de desarrollo de software hemos podido comprobar que a veces, para los casos de uso con una CC alta, los analistas añaden un diagrama de actividades UML [3] equivalente al texto del caso de uso para intentar paliar la dificultad de su comprensión. Desde nuestro punto de vista, esta solución tiene algunos inconvenientes, como la sensación de repetir el trabajo dos veces, la necesidad de comprobar que las dos representaciones del requisito son coherentes entre sí y la confusión añadida que podría entrañar para clientes y usuarios.

En cualquier caso, es necesario reconocer que el modelo de casos de uso utilizado por la herramienta REM (ver figura 2) y en el que están basadas las heurísticas, no está especialmente diseñado para casos de uso con caminos alternativos que impliquen más de un paso, al igual que le ocurre a otros modelos como los escenarios de Leite [14], por lo que en estas situaciones el valor de CC aumenta de forma artificial (es necesario repetir la misma condición varias veces). Este hecho pone de manifiesto la necesidad de incluir en el modelo la posibilidad de poder tener bloques de pasos bajo una misma condición. Otra posibilidad es, tal como se recoge en [12] o en [15], considerar una sección específica para las alternativas en lugar de entremezclar pasos alternativos dentro de la secuencia normal, aunque en nuestra opinión esto fragmenta el caso de uso y lo hace más difícil de entender.

5. Trabajo relacionado

La mayoría de los trabajos publicados hasta ahora en métricas de casos de uso proponen usar métricas como el número de casos de uso, número de actores o número de pasos como métricas de tamaño y complejidad de la especificación de requisitos [11,12,16,20]. Su interés radica en que son útiles para la estimación de ciertos atributos del sistema a desarrollar como el tamaño o el esfuerzo.

Relacionado con la calidad de la especificación de los requisitos, el trabajo de Ben Achour [2] estudia hasta que punto el *buen uso de la técnica* de casos de uso mejora la calidad de la especificación de requisitos. Para conseguirlo, realiza un experimento en el que compara la calidad de varias especificaciones de requisitos, las cuales han sido elaboradas por distintos grupos de autores: los que habían recibido formación específica para utilizar los casos de uso como técnica de especificación y los que no. La formación mencionada es el conocimiento de las *guías de estilo* y las *guías de contenido* para casos de uso. Dichas guías se basan en el modelo de casos de uso del proyecto CREWS así como en las recomendaciones de buen uso propuestas en [4]. Estas guías se usaron en [9] para fijar algunos de los intervalos propuestos.

Nuestro objetivo no es tanto estudiar el efecto de usar correctamente la técnica de casos de uso como automatizar en la medida de lo posible el proceso de verificación de requisitos. Para conseguir esta automatización existen algunas propuestas basadas en procesamiento de lenguaje natural (NLP). Según [10], el propósito del NLP aplicado a las especificaciones de requisitos se resume en aspectos como conocer el vocabulario usado, el estilo de escritura, la ambigüedad (grado de incertidumbre sintáctica o semántica de la sentencia), así como la presencia de frases inconexas, aspectos no especificados o falta de información en la especificación de requisitos.

6. Conclusiones y trabajo futuro

En este trabajo se ha presentado una revisión de las heurísticas basadas en métricas para casos de uso propuestas en [8,9]. Dichas heurísticas se basan en la determinación de intervalos de valores de las métricas en los que los casos de uso tienen una mayor probabilidad de presentar defectos. Como resultado del análisis realizado, se ha comprobado que los datos empíricos confirman las heurísticas propuestas. Además, se han analizado los datos para conocer qué tipo de defectos concretos predicen las heurísticas y cómo poder evitarlos. Los datos empíricos también han puesto de manifiesto la necesidad de modificar algunos aspectos del modelo de casos de uso inicial.

Nuestro trabajo futuro, está orientado a establecer y verificar una hipótesis más sólida que nos permita comparar el comportamiento de los casos de uso que pertenecen al intervalo de valores recomendables de las métricas con los que no pertenecen a él. Hasta ahora sólo se han hecho conjeturas sobre los casos de uso que están fuera del intervalo propuesto por la heurística. Nuestra intuición apunta que para poder probar hipótesis de este tipo es necesario verificar los casos de uso basándonos en tipos concretos de defectos. En otras palabras, se trataría de verificar empíricamente un hipótesis más fuerte que la verificada en este artículo. Dicha hipótesis se centraría en los tipos de defectos que suelen detectar las métricas propuestas, de forma que para un tipo de defecto D y una métrica m_D predictora de dicho tipo de defectos, la hipótesis podría enunciarse como:

$$\begin{array}{c} m_D(c) \notin [m_1, m_2] \Rightarrow P_{def_D}[c] \gg 1 - P_{def_D}[c] \\ \wedge \\ m_D(c) \in [m_1, m_2] \Rightarrow 1 - P_{def_D}[c] \gg P_{def_D}[c] \end{array}$$

donde $P_{def_D}[c]$ es la probabilidad de que un caso de uso c presente un defecto de tipo D. En otras palabras, la hipótesis establece que para un tipo concreto de defectos, el intervalo determina donde la probabilidad es más alta, de forma que fuera del intervalo la probabilidad de que ocurra es mínima. La confirmación de esta hipótesis nos llevaría a poder afirmar que la presencia de ciertos defectos es detectada con una probabilidad relativamente alta por una determinada métrica.

Una vez demostrada esta hipótesis, se pretende realizar un estudio experimental al objeto de evaluar si el conocimiento de las heurísticas y de los valores de las métricas mejora la eficiencia del proceso de verificación de requisitos, que es el objetivo principal de nuestra línea de investigación actual.

Agradecimientos

Queremos dar las gracias a Marcela Genero y Mario Piattini, de la Universidad de Castilla–La Mancha, y a Natalia Juristo, de la Universidad Politécnica de Madrid, por su ayuda en la elaboración de este trabajo y por sus acertadas respuestas a nuestras cuestiones.

Referencias

1. V. R. Basili *et al.* The Empirical Insvestigation of Perspective–Based Reading. *Empirical Software Engineering*, 1(2):133–164, 1996.
2. C. Ben Achour, C. Rolland, N. A. M. Maiden, and C. Souveyet. Guiding Use Case Authoring: Results of an Empirical Study. In *Proceedings of the Fourth IEEE International Symposium on Requirements Engineering*, 1999.
3. G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison–Wesley, 1999.
4. A. Cockburn. Structuring Use Cases with Goals. *Journal of Object–Oriented Programming*, Sept. y Nov./Dic. 1997.
5. A. Cockburn. *Writing Effective Use Cases*. Addison–Wesley, 2001.
6. A. Davis *et al.* Identifying and measuring quality in software requirements specification. In *Proceedings 1st Int'l Software Metrics Symposium*, Los Alamitos, California, 1993. IEEE Computer Society Press.
7. A. Durán. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información*. Tesis doctoral, Universidad de Sevilla, 2000.
8. A. Durán, A. Ruiz, B. Bernárdez, and M. Toro. Verifying Software Requirements with XSLT. *ACM Software Engineering Notes*, 29(1):39–44, 2002.
9. A. Durán, A. Ruiz-Cortés, R. Corchuelo, and M. Toro. Supporting Requirements Verificarion using XSLT. In *Proceedings of the IEEE Joint International Requirements Engineering Conference (RE'02)*, Essen, Germany, 2002.
10. F. Fabbrini *et al.* Achieving Quality in Natural Language Requirements. In *Proceedings of the 11 th International Software Quality Week*, San Francisco, 1998.
11. P. Feldt. Requirements metrics based on use cases. Master's thesis, Department of Communication Systems, Lund Institute of Technology, Lund University, Box 118, S-221 00 Lund, Sweden, 2000.
12. B. Henderson-Seller, D. Zowghi, T. Klemola, and S. Parasuram. Sizing Use Cases: How to Create a Standard Metrical Approach. In *Proceedings of the 8th International Conference on Object–Oriented Information Systems*, volume 2425 of *Lecture Notes in Computer Science*. Springer Verlag, 2002.
13. ISO/IEC. ISO 9126.1 Software Engineering–Product quality—Quality model. International Standard 9126.1–2001, International Organization for Standarization, 2001.
14. J. C. S. P. Leite, H. Hadad, J. Doorn, and G. Kaplan. A Scenario Construction Process. *Requirements Engineering Journal*, 5(1), 2000.
15. S. Lilly. Use Case-Based Requirements. Review Checklist. Project report, SRA International, Inc., 1999.
16. M. Marchesi. OOA Metrics for the Unified Modeling Language. In *Proceedings of the 2th EUROMICRO Conference on software Manteinance and Reengineering*, 1998.
17. T. J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, SE–2(4):308–320, Abril 1976.
18. K. Pohl. Requirements Engineering: An Overview. *Encyclopedia of Computer Science and Technology*, 36, 1997.
19. A. A. Porter, L. G. Votta, and V. R. Basili. Comparing Detection Methods for Sofware Requirements Inspections: A Replicated Experiment. *IEEE Transactions on Software Engineering*, 21(6):563–575, Junio 1995.
20. G. Schneider and J. P. Winters. *Applying Use Cases: a Practical Guide*. Addison–Wesley, 1998.
21. G. M. Schneider, J. Martin, and W. T. Tsai. An Experimental Study of Fault Detection In User Requirements Documents. *ACM Transactions on Software Engineering and Methodology*, 1(2):188–204, 1992.