Revista de Procesos y Métricas de las Tecnologías de la Información (RPM) ISSN: 1698-2029
VOL. 1, Nº 2, Agosto 2004, 13-24
Asociación Española de Sistemas de Informáticos (AEMES)

UNA PROPUESTA PARA LA VERIFICACIÓN DE REQUISITOS BASADA EN MÉTRICAS*

B. Bernárdez , A. Durán, M. Toro
Universidad de Sevilla, Depto. Lenguajes y Sistemas Informáticos
Avda. Reina Mercedes s/n 41012, Sevilla, Spain
E-Mail : {beat,amador,mtoro}@lsi.us.es

M. Genero
Grupo Alarcos, Departamento de Informática Universidad de Castilla-La Mancha
Paseo de la Universidad, 4 13071, Ciudad Real, Spain
E-Mail: Marcela.Genero@uclm.es

Abstract: In this paper, a set of heuristics, based on metrics, to requirements verification are described. Also, the findings of the empirical studies carried out to validate the heuristics are summarised. The heuristics profit from our experience on verifying requirements specification developed by students of the Computer Science Engineering School. The intuition behind the heuristics is that there exists certain cause—effect relationship between the value of some use case metrics and the presence of concrete defects types in use cases. A first empirical review presented in our previous work has allowed us to identify the list of the main defect types that are detected by the heuristics. Based on this list, we have defined a complete taxonomy of defects to use cases. Using that taxonomy, a set of requirement specifications have been verified and a second empirical review have been carried out to corroborate the validity of heuristics, obtaining some successful results.

Resumen: En este artículo se describen un conjunto de heurísticas para verificación de requisitos basadas en métricas. También se presentan los principales resultados de las revisiones empíricas que se han llevado a cabo para validarlas. Las heurísticas son fruto de nuestra experiencia en la verificación de especificaciones de requisitos desarrolladas por alumnos de Ingeniería Informática de la Universidad de Sevilla. La intuición en la que se basan las heurísticas es que existe cierta relación causa—efecto entre el valor de ciertas métricas para casos de uso y la presencia de defectos en éstos. Una primera revisión empírica, publicada en un trabajo previo, nos permitió identificar los principales tipos de defectos que detectan las heurísticas. Basándonos en dicho conjunto de defectos se ha elaborado una taxonomía de defectos para casos de uso, que se ha usado como instrumento para verificar un conjunto de especificaciones de requisitos objeto de la segunda revisión empírica realizada. El objetivo de dicho estudio ha sido corroborar la validez de las heurísticas y se han obtenido resultados interesantes que confirman la validez de las mismas.

Palabras clave Verificación de requisitos, casos de uso, métricas de software, heurísticas, modelo de calidad

* Este trabajo está financiado parcialmente por los proyectos CICYT WebMade (TIC 2003-02737-C02-01), CICYT CALIPO (TIC 2003-07804-C05-03) y MESSENGER (PCC-03-003-1).

1. INTRODUCCIÓN

Uno de los principales objetivos del documento de requisitos es servir de base a clientes, usuarios y desarrolladores para alcanzar un acuerdo sobre las necesidades que debe satisfacer el sistema a desarrollar. Por este motivo, la calidad de la especificación de requisitos es un elemento crucial para la buena marcha del proyecto de desarrollo de software y para la calidad del producto final.

Durante el proceso de ingeniería de requisitos (IR), es común utilizar los casos de uso para especificar los requisitos funcionales [Bernárdez *et al.* 2004]. La gran aceptación de esta técnica en el ámbito de los sistemas de información, se debe según [Jacobson 2002] a su facilidad para jugar diferentes papeles durante el proceso de ingeniería del software. No obstante, esta amplia y rápida implantación ha provocado en algunas ocasiones, una manera inadecuada de usar los casos de uso debido fundamentalmente a las siguientes razones:

- El uso del lenguaje natural para escribir la especificación textual de los casos de uso. Por su propia naturaleza, el lenguaje natural es ambiguo [Fantechi *et al.* 2002]. No obstante, coincidiendo con autores como Jacobson [Jacobson 2002], pensamos que el lenguaje natural es la posibilidad más razonable para que la especificación de requisitos sea una vía real de comunicación con los clientes y usuarios.
- Las distintas interpretaciones que los autores de casos de uso hacen de las relaciones de inclusión y extensión, que redunda en el uso incorrecto de dichas relaciones provocando que las especificaciones sean difíciles de leer y de comprender.
- La tendencia de los autores de casos de uso a especificarlos bien pensando en la aplicación a desarrollar, en concreto, en su interfaz o en su implementación, bien abusando de estructuras de control similares a las de un programa de código. Esto es debido a que en algunas ocasiones el autor del caso de uso está *demasiado acostumbrado a programar.*

Según [Anda y Sjøberg 2002], las características especiales de la especificación de requisitos al usar los casos de uso han provocado la necesidad de idear métodos específicos para evaluar la calidad de éstos, que si bien son requisitos requieren la comprobación de aspectos concretos por su estructura particular.

De forma tradicional, la verificación y validación de requisitos han sido las actividades consideradas parte del SQA (*Software Quality Assuranc*e) en la fase de IR. La verificación de requisitos puede considerarse como la actividad cuyo objetivo es alcanzar la *calidad interna* [ISO/IEC 2001] exigible a la especificación de requisitos conforme a las normas establecidas previamente por la organización, mientras que la validación tiene como objetivo asegurar que los requisitos elicitados, documentados, analizados y verificados representan realmente las necesidades de clientes y usuarios [Pohl 1997] [Durán 2000].

Dentro del contexto de la calidad en IR, este trabajo describe un conjunto de heurísticas cuyo objetivo es facilitar la verificación de casos de uso. Estas heurísticas, inicialmente presentadas en [Durán *et al.* 2001], son fruto de la experiencia en la verificación de especificaciones de requisitos elaboradas por alumnos de Ingeniería Informática de la Universidad de Sevilla. Posteriormente, han sido revisadas y evaluadas mediante dos revisiones empíricas, convirtiéndose en un mecanismo que puede ser útil para mejorar la efectividad (proporción de defectos identificados) y la eficiencia (cantidad de defectos identificados por unidad de tiempo) del proceso de verificación de requisitos.

En concreto, las heurísticas tienen como objetivo identificar casos de uso 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 [Durán 2004]. Si el valor de estas métricas está fuera del rango habitual establecido en las heurísticas correspondientes, un caso de uso se considera como potencialmente defectuoso y debe ser revisado.

El resto de este trabajo se divide de la siguiente forma: en la próxima sección se describen las heurísticas definiendo las métricas y resumiendo los resultados de la primera revisión empírica, que reveló los principales tipos de defectos que predicen las heurísticas. En la sección 3 se presenta una taxonomía de defectos para casos de uso que

son habitualmente detectados por las heurísticas. La sección 4 resume la segunda revisión empírica llevada a cabo para validar las heurísticas. Por último, la sección 5 expone las principales conclusiones y apunta las posibles líneas de trabajos futuros.

2. HEURÍSTICAS DE VERIFICACIÓN DE REQUISITOS

Las métricas que sustentan las heurísticas están basadas en el modelo de casos de uso de REM (Figura 1.). En este modelo un caso de uso es básicamente una secuencia de pasos. La acción que se realiza en un paso puede ser: de actor, si representa una acción realizada por un actor; de sistema, si la acción es realizada por el sistema o *de caso de uso* si el paso consiste en realizar otro caso de uso (una extensión si el paso es condicional y una inclusión en otro caso).

En la tabla 2 se muestran las métricas principales de las heurísticas y sus rangos habituales.

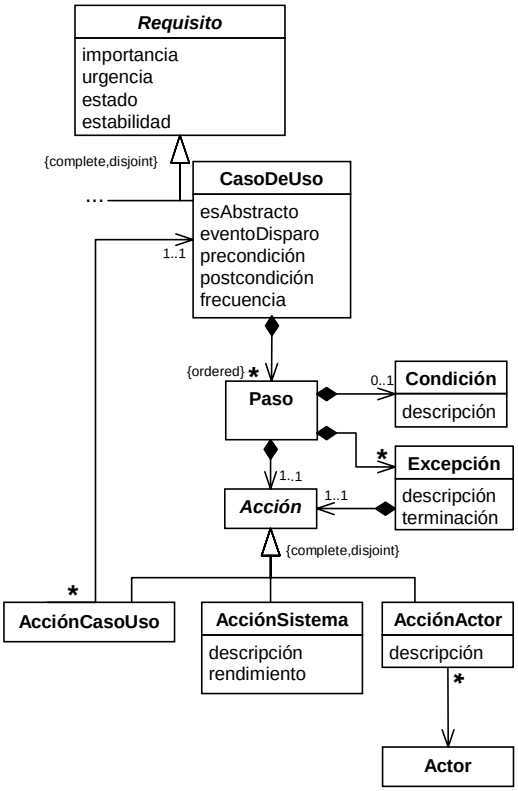

Figura 1. Modelo de casos de uso de REM

Métricas auxiliares	Descripción
NOS	Número de pasos
NOAS	Número de pasos de actor
NOSS	Número de pasos de sistema
NOUS	Número de pasos que activan otro caso de uso
NOE	Número de excepciones
NOCS	Número de pasos condicionales

Tabla 1. Métricas auxiliares de las heurísticas

Métricas principales	Rango habitual	Descripción
NOS	[4,9]	Número de pasos
NOAS/NOS	[30%,60%]	Proporción de pasos de actor
NOSS/NOS	[40%, 80%]	Proporción de pasos de sistema
NOUS/NOS	[0%,35%]	Proporción de pasos de caso de uso
CC	[1,5]	Complejidad Ciclomática (NOCS+NOE+1)

Tabla 2. Métricas principales y rango habitual

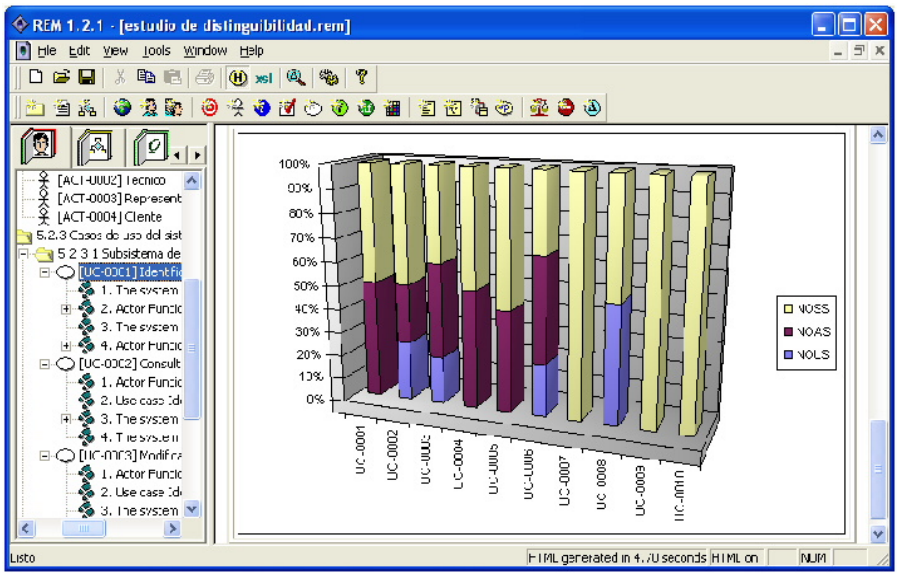

Figura 2. Proporción de NOAS, NOSS y NOUS de un conjunto de casos de uso en REM

El rango habitual de cada métrica ([*inf*, *sup*]) se ha determinado tras estudiar un conjunto de 414 casos de uso elaborados por nuestros alumnos, tal como se describe en [Durán *et al.* 2001].

Tomando como base el rango habitual de las

métricas, la herramienta REM genera automáticamente un "informe de métricas" en el que figuran los valores de las métricas de cada caso de uso del documento de requisitos bajo estudio. Es posible configurar la herramienta REM para que esos valores aparezcan en un formato distinto si están fuera del rango habitual de las métricas. También se pueden generar gráficas como diagramas de barras en los que se visualicen los valores de las métricas (ver figura 2).

2.1. Resumen de las heurísticas

En esta sección se resumen las heurísticas para verificación de casos de uso. Para cada heurística, en primer lugar se justifica su definición. A continuación, se describen los principales tipos de defectos que detecta la heurística. Como ya se ha comentado, la identificación de dichos tipos de defectos es fruto de la primera revisión empírica realizada [Bernárdez *et al.* 2003], tras verificar un conjunto de 127 casos de uso desarrollados por nuestros alumnos.

A) Heurística basada en NOS

La justificación de esta heurística, tal como se describe en [Durán *et al.* 2001] y de acuerdo con otros autores como Cockburn [Cockburn 2001], es que los casos de uso demasiado cortos suelen ser incompletos o no representan realmente una interacción actor–sistema resultando generalmente triviales. Por otro lado, los casos de uso demasiado largos son frecuentemente incomprensibles, presentan un nivel de detalle excesivo [Lilly 1999] o avanzan a un ritmo muy lento [Cockburn 2001], es decir, se dirigen con demasiada lentitud a alcanzar el objetivo del caso de uso siendo más difíciles de leer.

No obstante, 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.

B) Heurística basada en NOAS/NOS

Esta heurística se basa 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.

Las situaciones que llevan a esta métrica fuera del rango habitual 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 en principio no puede considerarse un defecto, sino más bien una información interesante aunque no esencial del caso de uso, será un defecto cuando se produzca con excesiva frecuencia.

C) Heurística basada en NOSS/NOS

Esta heurística se basa en el mismo razonamiento que la anterior. En situaciones con NOSS/NOS alto el defecto más 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.

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 [Lilly 1999], 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 único paso de sistema separando las acciones por comas.

D) Heurística basada en 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 [Booch *et al.* 1999] deben usarse principalmente como medio para evitar repeticiones de secuencias de pasos en distintos casos de uso

- Los principales defectos que predice la heurística son: 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.
- 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.

E) Heurística basada en CC

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 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 2).

No obstante, hay que reconocer que a veces la métrica CC no representa el número real de caminos alternativos del caso de uso. Esto ocurre principalmente por dos motivos:

- A veces el autor del caso de uso escribe un paso condicional para especificar que el sistema comprueba una situación, por ejemplo, si un paso dice “Si el *password* es correcto el caso de uso continúa”. Ese paso condicional no supone un camino alternativo y en [Cockburn 2001] se recomienda que se sustituya por sentencias como “El sistema verifica que el *password* es correcto” que facilitan la legibilidad del caso de uso mejorando su comprensibilidad.
- El modelo de casos de uso utilizado por la herramienta REM y en el que están basadas las heurísticas (figura 1), 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 [Leite *et al.* 2000], 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 [Henderson-Sellers *et al.* 2002] o en [Lilly 1999], 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.

También se ha podido comprobar que la existencia de excepciones no siempre hace más difícil la lectura del caso de uso. Por esto, quizás sea conveniente redefinir CC dando más peso a los pasos condicionales del caso de uso que a las excepciones.

2.2. Principales defectos detectados por las heurísticas

Como resultado de la primera revisión empírica presentada en [Bernárdez *et al.* 2003] se ha elaborado la tabla 3 que resume los principales defectos detectados por las heurísticas descritas. Como se puede ver en dicha tabla, las causas son distintas si el valor de la métrica está por debajo de *inf* o por encima de *sup*.

Además, el valor de las métricas NOAS/NOS y NOSS/NOS suelen estar relacionados, de forma que si NOAS/NOS es alto, NOSS/NOS es bajo y

viceversa. Por esta razón, las causas que provocan valores altos de una de estas métricas coinciden con las de valores bajos en la otra.

La primera columna de la tabla 3 muestra la métrica bajo estudio, la segunda indica si el valor de la métrica es alto o bajo, y la tercera las principales causas de defectos en esa situación. Tomando como base estas causas se ha elaborado el modelo de calidad que se presenta en la sección 3

3. MODELO DE CALIDAD PARA CASOS DE USO

El concepto de modelo de calidad de requisitos es, según Gnesi [Gnesi 2000], el conjunto de reglas frente a las que evaluar una especificación de requisitos (reglas sintácticas, semánticas, estructura del documento y estructura de las sentencias). Existen dos formas de presentar un modelo de calidad, a saber, como una lista de propiedades deseables (también llamadas características de calidad) de los requisitos [Davis *et al.* 1993], o como una taxonomía de defectos [Basili y Weiss 1981]. Entendiendo que un defecto en los requisitos es la ausencia de una propiedad deseable.

Métrica	**Valor**	**Principales tipos de defectos**
NOS	bajo (<4)	Incompletitud, exceso de modularidad, trivialidad.
NOS	alto (>9)	Existencia de demasiados pasos alternativos, demasiado desglose de los pasos de actor o de sistema.
NOAS/NOS	bajo(<30%)	No incluye todo lo que debe hacer el actor para alcanzar el objetivo del
NOSS/NOS	alto (>80%)	alcanzar el objetivo del caso de uso, demasiado desglose de los pasos del sistema, expresa un proceso *batch*, contiene referencias concretas a elementos de la interfaz, incluye acciones internas del sistema.
NOAS/NOS	alto (>60%)	No incluye todo lo que debe hacer el sistema para alcanzar el objetivo del caso de uso, demasiado desglose de los pasos del actor, incluye demasiadas interacciones del actor principal con elementos del entorno del sistema.
NOSS/NOS	bajo <40%)	
NOUS/NOS	alto (>35%)	Uso abusivo de las estructuras *include* o *extend*, incluye referencias concretas a elementos de la interfaz de usuario, expresa menús de la aplicación.
CC	alto (>5)	Incomprensibilidad.

Tabla 3. Principales tipos de defectos detectados por las heurísticas

En general, la mayoría de las técnicas de lectura que se han propuesto para verificar requisitos vienen acompañadas de un modelo de calidad de requisitos. Tal como se explica en [Shull *et al.* 2003], técnicas diferentes sirven para identificar diferentes tipos de defectos o, dicho de otra forma, una técnica de verificación de requisitos es apropiada para evaluar determinados aspectos de la calidad de los requisitos y no otros.

Según [Basili y Weiss 1981], para facilitar el uso de un modelo de calidad de requisitos, éste se puede especificar en forma de lista de comprobación (checklist), es decir, una lista de preguntas que se comprueban en cada requisito y cuya respuesta negativa implica la presencia de un defecto en él.

Siguiendo esta recomendación, se ha elaborado una pregunta para comprobar cada tipo de defecto detectado por las heurísticas y se han agrupado bajo una categoría relativa a una característica de calidad

de los requisitos según las propuestas en [Davis *et al.* 1993], [Lilly 1999] y [Cockburn 2001]. La consideración de estas tres propuestas se debe a diferentes motivos:

- La propuesta de Davis [Davis *et al.* 1993] se ha usado por recoger varias propuestas anteriores de propiedades deseables de los requisitos, contemplando cuestiones como la completitud, la ambigüedad o la coherencia entre requisitos.
- Las otras dos propuestas ([Cockburn 2001] y [Lilly 1999]) se han escogido por ser específicas para los casos de uso. La *checklist* de Lilly [Lilly 1999] comprueba situaciones como el uso correcto de las relaciones entre casos de uso (mediante las estructuras *include* y *extend*). Por su parte, la propuesta de Cockburn [Cockburn 2001] constituye una de las guías más exhaustiva en el buen uso de la técnica de casos de uso, considerando aspectos como el nivel de abstracción de los casos de uso y la expresión correcta de las alternativas en éstos.

Completitud

Adaptando la definición de [Davis *et al.* 1993] a los casos de uso, un caso de uso es completo si especifica todo lo que deben hacer el actor y el sistema (externamente) para alcanzar el objetivo del caso de uso y si se consideran todas las respuestas del sistema a situaciones anormales. Para comprobar si un caso de uso es completo se propone la siguiente lista de preguntas:

- ¿Hay respuestas a todas las peticiones que el actor del caso de uso hace al sistema y viceversa?
- ¿Se contemplan todos los posibles escenarios para poder alcanzar el objetivo del caso de uso?
- ¿Se especifican todas las secuencias alternativas a la secuencia normal?
- ¿Se contemplan todas las posibles excepciones a la secuencia normal?

Comprensibilidad

Un caso de uso es comprensible si todos los tipos de lectores (cliente, usuario, jefe de proyecto, desarrollador o responsable de pruebas) pueden entenderlo fácilmente con una mínima explicación del autor [Davis *et al.* 1993]. Para ver si un caso de uso es comprensible, se propone la siguiente lista de cuestiones:

- ¿Es posible leer el caso de uso sin volver atrás en repetidas ocasiones?
- ¿Es difícil seguir la secuencia normal del caso de uso por la presencia de las relaciones *include* o *extend*?
- ¿Es difícil seguir la secuencia de pasos por la existencia de demasiados pasos alternativos?
- ¿Se han desglosado demasiado los pasos de algún actor o del sistema provocando que el caso de uso avance a un ritmo muy lento?
- ¿Aparecen pasos condicionales para expresar que el sistema comprueba una situación que permite al caso de uso continuar su realización?

Concisión

Un caso de uso es conciso si no incluye información superflua o innecesaria. Para saber si un caso de uso es conciso se deben comprobar las siguientes cuestiones:

- ¿Podría el caso de uso ser expresado con menos palabras?
- ¿Existen elementos que se puede obviar o aparecen anotaciones innecesarias y que dificultan la lectura del caso de uso?
- ¿Aparecen demasiadas interacciones entre el actor principal del caso de uso y otros elementos del entorno?

No trivialidad

Un caso de uso es no trivial si su secuencia de pasos conduce al actor a conseguir el objetivo que persigue la realización del caso de uso. Para comprobar si un caso de uso es no trivial la siguiente lista de cuestiones debe ser comprobada:

- ¿Expresa el nombre del caso de uso un objetivo de un usuario que el sistema debe implementar?
- ¿Conduce el caso de uso al actor a conseguir alguno de sus objetivos sin representar un conjunto de interacciones triviales?

Uso apropiado de la técnica

La técnica de casos de uso se debe usar cuando el requisito funcional que se representa requiere la interacción del sistema con un actor, bien sea un usuario o bien otro elemento externo al sistema en construcción. Las cuestión asociada a esta

característica de calidad es: ¿Representa el caso de uso un comportamiento que requiere la participación de otro elemento externo al propio sistema?.

Independencia del diseño

Un caso de uso es independiente del diseño si describe sólo comportamiento externo, sin anticipar detalles del diseño o la implementación. Para comprobar si un caso de uso es independiente del diseño se han elaborado las siguientes cuestiones:

- ¿Presenta el caso de uso referencias concretas a elementos de la interfaz de usuario?
- ¿Describe el caso de uso sólo lo que debe hacer el sistema sin especificar cómo debe hacerlo?
- ¿Intenta anticipar el caso de uso cómo van a ser los menús de la aplicación a desarrollar?

4. REVISIÓN EMPÍRICA DE LAS HEURÍSTICAS

Una vez identificados los tipos de defectos que predice cada heurística y para corroborar la relación existente entre el valor de las métricas y la presencia de dichos tipos de defectos en los casos de uso, se ha realizado un segunda revisión empírica, cuyos resultados se muestran en la figura 3.

En dicho estudio, se pretendió contrastar una hipótesis que nos permitiera 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. Nuestra intuición apuntaba que para poder probar hipótesis de este tipo es necesario verificar los casos de uso basándonos en tipos concretos de defectos, aquellos recogidos en el modelo de calidad de la sección 3. En otras palabras, dicha hipótesis se centraría en los tipos de defectos que suelen detectar las heurísticas propuestas. De esta forma, para cada métrica *m*, $D_{\inf}$ y $D_{\sup}$ representan, respectivamente, el conjunto de tipos de defectos que suelen ocurrir cuando el valor de *m* es menor que *inf* o mayor que *sup* (de acuerdo con la tabla 3). Por tanto, $D = D_{\inf} \cup D_{\sup}$ representa el conjunto de tipos de defectos que predice una métrica concreta.

Formalmente, una vez definido el rango habitual, [*inf,sup*] y dado un caso de uso c y una métrica m_D que puede predecir defectos de tipo *D*, la hipótesis contrastada se enuncia así:

$$m_D(c) < \inf \Rightarrow P_{D_{\inf}}[c] >> \overline{P_{D_{\inf}}[c]}$$
$$\wedge$$
$$m_D(c) \in [\inf, \sup] \Rightarrow \overline{P_D[c]} >> P_D[c]$$
$$\wedge$$
$$m_D(c) > \sup \Rightarrow P_{D\sup}[c] >> \overline{P_{D_{\sup}}[c]}$$

donde $P_D[c]$ representa la probabilidad de que un caso de uso *c* presente uno o más defectos del conjunto *D* y $\overline{P_D[c]}$ representa la probabilidad de que el caso de uso *c* no presente ningún defecto de los contenidos en el conjunto *D*.

Para estudiar esta hipótesis, se tomó un conjunto de especificaciones de requisitos elaboradas por alumnos de quinto curso de Ingeniería Informática de la Universidad de Sevilla (6 especificaciones de requisitos con un total de 131 casos de uso). En este caso, varios expertos en calidad de Sadiel, S.A, han revisado dichas especificaciones aplicando la *checklist* descrita en la sección 3. A estos expertos no se les dio a conocer las heurísticas, para evitar que estuvieran condicionados por ellas a la hora de verificar las especificaciones de requisitos bajo estudio.

El análisis empírico de los datos, cuyo resumen puede verse en la figura. 3, muestra un mayor porcentaje de casos de uso con defectos fuera de los rangos habituales de las métricas NOAS/NOS, NOSS/NOS Y NOUS/NOS, confirmando la hipótesis estudiada.

Sin embargo, los resultados obtenidos para la métrica NOS no son muy alentadores, quizás a causa de la influencia de los diferentes estilos de redacción del autor de los casos de uso, como ya se ha comentado al describir la heurística.

Aunque eran de esperar, tampoco los resultados obtenidos para la métrica CC son los deseados, fuera del rango habitual hay numerosos requisitos que no presentan los tipos de defectos que predice la heurística. La razón de esta situación es el posible crecimiento artificial del valor de CC, cuyas causas han sido apuntadas en la sección 2.1. Este resultado indica no sólo la necesidad de introducir algunas mejoras en el modelo de casos de uso inicial en el que se basan las heurísticas [Bernárdez et al. 2003], sino también la posibilidad de redefinir la métrica

CC para que represente el número real de caminos alternativos de casos de uso.

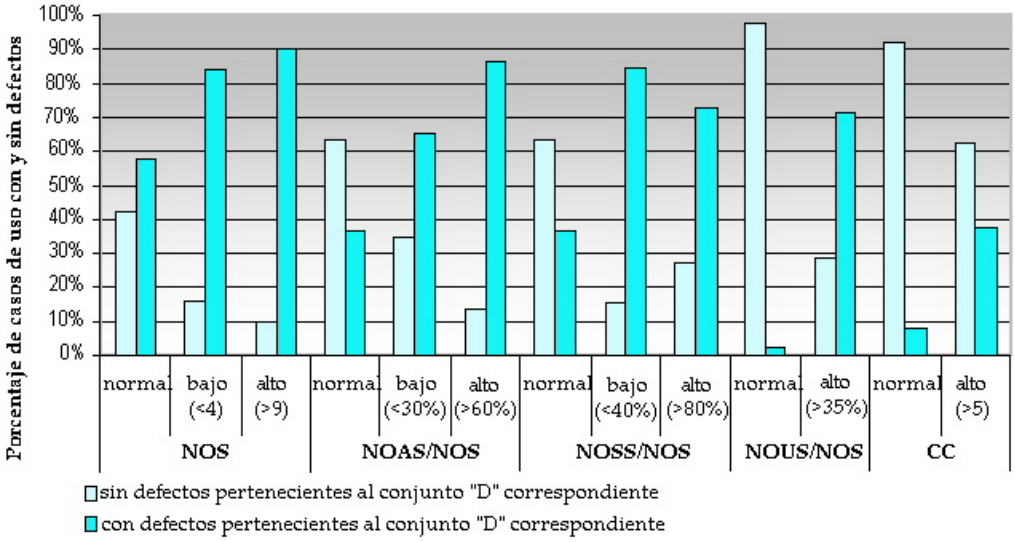

Figura 3. Resultados empíricos

5. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se ha descrito un conjunto de heurísticas para verificación de casos de uso y los resultados del segundo estudio empírico realizado para validarlas. Además se ha presentado un modelo de calidad para casos de uso que sirve de orientación al revisor de los casos de uso sobre qué tipos de defectos detectan las heurísticas propuestas. Con base en los principales tipos de defectos detectados por las heurísticas y tras un análisis exhaustivo de especificaciones de requisitos, se ha elaborado una taxonomía de defectos para casos de uso como complemento a las heurísticas.

Ese análisis ha permitido, no sólo refinar las heurísticas y el modelo de calidad, sino también identificar posibles mejoras en el modelo de casos de uso inicial de REM.

Aunque la idea subyacente de las heurísticas es orientar a un miembro del SQA durante la verificación de requisitos, existe otra posible aplicación de las éstas que es como método de auto—aprendizaje. Es decir, un ingeniero de requisitos que comience a enfrentarse con los casos de uso puede, tras escribir un conjunto de casos de uso, calcular el valor de las métricas de éstos y en función de los valores obtenidos reconocer qué tipos de defectos son los que ha cometido, corrigiéndose a si mismo, mejorando la calidad de los casos de uso especificados e intentando evitar en el futuro cometer de nuevo dichos fallos. Para poner en práctica este método de auto—aprendizaje lo ideal es incorporar a la herramienta REM un menú de ayuda similar a un *asistente* que procese los casos de uso de la especificación e indique qué casos de uso tienen tendencia a presentar defectos según el valor de las métricas. En una futura versión de la herramienta REM, actualmente en desarrollo, esta funcionalidad puede ser incorporada.

Otra de nuestras posibles líneas de trabajo futuro es aplicar las heurísticas en el ámbito industrial para corroborar una vez más su validez y realizar una familia de experimentos que comprueben si el conocimiento de las heurísticas realmente permite mejorar la efectividad y eficiencia del proceso de verificación de requisitos.

REFERENCIAS

[Anda y Sjøberg 2002] B. Anda y D. Sjøberg. Towards an inspection technique for use case models. En *Actas del 14 o Software Engineering Knowledge Engineering (SEKE'02*), Ischia, Italia, 2002.

[Basili y Weiss 1981] V. R. Basili y D. Weiss. Evaluation of a Software Requirements Documents by Analysis of Change Data. En *Actas del 5º International Conference on Software Engineering*, San Diego, California, 1981.

[Bernárdez *et al.* 2003] B. Bernárdez, A. Durán, y M.Toro. Una revisión empírica de heurísticas de verificación de casos de uso basadas en métricas. En *Actas de las JISBD'03*, Alicante, 2003.

[Bernárdez *et al.* 2004] B. Bernárdez, A. Durán, y M. Genero. *Metrics for Software Conceptual Models*, capítulo Metrics for Use Cases: A Survey of Current Proposals. Imperial College, 2004.

[Booch *et al.* 1999] G. Booch, J. Rumbaugh, y I. Jacobson. *The Unified Modeling Language User Guide*. Addison–Wesley, 1999.

[Cockburn 2001] A. Cockburn. *Writing Effective Use Cases*. Addison–Wesley, 2001.

[Davis *et al.*1993] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A.Dinh, G. Kincaid, G. Ledeboer, P. Reynols, P. Sitaran, A. Ta, y M. Theofanos. Identifying and measuring quality in software requirements specification. En *Actas del 1 o Int'l Software Metrics Symposium*, Los Alamitos, California, 1993. IEEE Computer Society Press.

[Durán *et al.* 2001] A. Durán, B. Bernárdez, A. Ruiz-Cortés, y M. Toro. An XML–Based Approach for the Automatic Verification of Software Requirements Specifications. En *Actas del 4 o Workshop on Engenharia de Requisitos*, Buenos Aires, Argentina, 2001.

[Durán 2000] A. Durán. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información*. Tesis doctoral, Universidad de Sevilla, 2000.

[Durán 2004] A. Durán. REM página web de la herramienta REM:. http://rem.lsi.us.es, 2004.

[Fantechi *et al.*2002] A. Fantechi, S. Gnesi, G. Lami, y A. Macari. Application of Linguistic Techniques for Use Case Analysis. En *Actas del IEEE Joint International Requirements Engineering Conference (RE'02)*, Essen, Alemania, 2002.

[Gnesi 2000] S. Gnesi. Analysis of Software Requirements, 2000. Disponible en http://www.iei.pi.cnr.it/ERI/iei/qmslideseri.ppt.

[Henderson-Sellers *et al.*2002] B. Henderson-Seller, D. Zowghi, T. Klemola, y S. Parasuram. Sizing Use Cases: How to Create a Standard Metrical Approach. En *Actas del 8 o International Conference on Object–Oriented Information Systems*, volumen 2425 de *Lecture Notes in Computer Science*. Springer Verlag, 2002.

[ISO/IEC 2001] ISO/IEC. ISO 9126.1 Software Engineering–Product quality— Quality model. International Standard 9126.1–2001, International Organization for Standarization, 2001.

[Jacobson 2002] I. Jacobson. Use Cases–Yesterday, Today and Tomorrow. Informe técnico, Rational Software, 2002.

[Leite *et al.*2000] J. C. S. P. Leite, H. Hadad, J. Doorn, y G. Kaplan. A Scenario Construction Process. *Requirements Engineering Journal*, 5(1), 2000.

[Lilly 1999] S. Lilly. Use Case–Based Requirements: Review Checklist. Informe técnico, SRA International, Inc., 1999.

[McCabe 1976] T. J. McCabe. A Complexity Measure. *IEEE Transactions on Software Engineering*, SE–2(4):308–320, Abril 1976.

[Pohl 1997] K. Pohl. Requirements Engineering: An Overview. *Encyclopedia of Computer Science and Technology*, 36, 1997.

[Shull *et al.* 2003] F. Shull, J.Carver, G.Travassos, J. Maldonado, R. Conradi, y V.Basili. *Lecture Notes on Empirical Software Engineerin*g, capítulo Replicated Studies: Building a Body of Knowledge about Software Reading Techniques, páginas 39–84. A World Scientific Singapore, 2003. Eds. Juristo N. and Moreno A.