Sobre requisitos no funcionales en software Ingeniería

Lorenzo Chung[1] y Julio César Sampaio do Prado Leite[2]

[1]Departamento de Ciencias de la Computación, Universidad de Texas en Dallas
[2]Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
www.utdallas.edu/~chung/, www.inf.puc-rio.br/~julio

Abstracto. Esencialmente, la utilidad de un sistema de software está determinada tanto por su funcionalidad como por sus características no funcionales, como usabilidad, flexibilidad, rendimiento, interoperabilidad y seguridad. No obstante, ha habido un énfasis desigual en la funcionalidad del software, aunque la funcionalidad no es útil o utilizable sin las características no funcionales necesarias. En este capítulo, revisamos el estado del arte sobre el tratamiento de requisitos no funcionales (en adelante, NFR), al tiempo que brindamos algunas perspectivas para direcciones futuras.

Palabras clave: Requisitos no funcionales, NFRs, objetivos blandos, satisfacción, ingeniería de requisitos, ingeniería de requisitos orientada a objetivos, alternativas, criterios de selección.

1. Introducción

"Es más difícil lidiar con lo blando que con lo duro". [Anónimo]

Esencialmente, la utilidad de un sistema está determinada tanto por su funcionalidad como por sus características no funcionales, como usabilidad, flexibilidad, rendimiento, interoperabilidad y seguridad. Sin embargo, ha habido un énfasis desigual en la funcionalidad del sistema, a pesar de que la funcionalidad no es útil o utilizable sin las características no funcionales necesarias.

Como casi todo lo demás, el concepto de calidad también es fundamental para la ingeniería de software, y se deben tener en cuenta tanto las características funcionales como las no funcionales en el desarrollo de un sistema de software de calidad. Sin embargo, en parte debido a la corta historia detrás de la ingeniería de software, en parte debido a la demanda de tener sistemas en ejecución rápidamente que satisfagan las necesidades básicas, y también en parte debido a la naturaleza "blanda" de las cosas no funcionales, la mayor parte de la atención en la ingeniería de software En el pasado se ha centrado en notaciones y técnicas para definir y proporcionar las funciones que debe realizar un sistema de software.

Una práctica frecuentemente observable, como resultado de este énfasis desequilibrado en el aspecto funcional de un artefacto de software, es que las características de calidad necesarias se tratan sólo como cuestiones técnicas relacionadas principalmente con el diseño detallado o las pruebas de un software.

AT Borgida et al. (Eds.): Mylopoulos Festschrift, LNCS 5600, págs. 363–379, 2009.

sistema implementado. Este tipo de práctica, por supuesto, es bastante inadecuada. El diseño detallado y las pruebas no tienen mucho sentido sin las fases previas de comprender cuál es el problema del mundo real para el cual se podría proponer un sistema de software como solución y también cuáles podrían ser los detalles de la solución de software, es decir, los requisitos. como. Y los problemas del mundo real están más orientados no funcionalmente que funcionalmente, por ejemplo, baja productividad, procesamiento lento, alto costo, baja calidad y cliente insatisfecho.

Aunque la comunidad de ingeniería de requisitos ha clasificado los requisitos como funcionales o no funcionales, la mayoría de los modelos de requisitos y lenguajes de especificación de requisitos existentes carecían de un tratamiento adecuado de las características de calidad. Tratar las características de calidad como un todo, y no sólo como funcionalidad, ha sido un enfoque clave de los trabajos en el área de la ingeniería de requisitos orientada a objetivos [1] [2], y en particular el Marco NFR [3] que trata las características no funcionalidad a un alto nivel de abstracción tanto para el problema como para la solución.

Este capítulo presenta una revisión de la literatura sobre NFR, con énfasis en las diferentes definiciones, esquemas de representación, así como usos más avanzados de los conceptos. Al final, concluimos el capítulo discutiendo cuestiones abiertas en el tratamiento temprano de los NFR y sus impactos en la construcción de software.

2 ¿Qué son los requisitos no funcionales?

En la literatura se puede encontrar una gran cantidad de definiciones de requisitos no funcionales (NFR).

Hablando coloquialmente, se ha hecho referencia a los NFR como “-ilidades” (p. ej., usabilidad) o “-idades” (p. ej., integridad), es decir, palabras que terminan con la cadena “-ilidad” o “-idad”. Se puede encontrar una gran lista de este tipo de palabras, por ejemplo, en [3]. Hay muchos otros tipos de NFR que no terminan ni con “-ilidad” ni con “-idad”, como el rendimiento, la facilidad de uso y la coherencia.

Un trabajo importante sobre NFR es el Marco NFR [1] [3], que desacopla el concepto de funcionalidad de otros atributos de calidad y preocupaciones por la productividad, el tiempo y el costo, mediante un mayor nivel de abstracción. En lugar de centrarse en expresar los requisitos en términos de funciones, restricciones y atributos detallados, el Marco NFR ideó la distinción de los NFR utilizando los conceptos de meta y meta blanda. Se describirán más detalles sobre el Marco NFR en la Sección 4.

En el área de Arquitectura de Software, una palabra clave frecuentemente encontrada es “atributos de calidad” [4], entendida como un conjunto de inquietudes relacionadas con el concepto de calidad. Para una definición de calidad, aquí se utiliza un estándar IEEE [5] como complemento: “La calidad del software es el grado en que el software posee una combinación deseada de atributos (por ejemplo, confiabilidad, interoperabilidad)”.

Varios otros autores también han tratado este tipo de preocupaciones. Por ejemplo, la calidad básica (funcionalidad, confiabilidad, facilidad de uso, economía y seguridad) se distingue de la calidad adicional (flexibilidad, reparabilidad, adaptabilidad, comprensibilidad, documentación y mejora) en [6].

En el área de ingeniería y gestión, la conocida estrategia QFD (Quality Function Deployment) [7] distingue la calidad positiva de la calidad negativa: “QFD es bastante diferente en que busca los requisitos del cliente tanto "hablados" como "tácitos" y maximiza los requisitos. Cualidad "positiva" (como facilidad de uso, diversión, lujo) que crea valor. Los sistemas de calidad tradicionales tienen como objetivo minimizar la calidad negativa (como defectos, mal servicio)”. Una de las técnicas utilizadas por las estrategias QFD es la Casa de la Calidad [8], en la que el proceso comienza "...con el cliente, cuyos requisitos se denominan atributos del cliente (CA´s) - frases que los clientes usan para describir productos y productos. características...". Por cierto, ninguno de los ejemplos de CA ś en [8] está relacionado con la funcionalidad o solo con la funcionalidad.

En el área de requisitos de software, el término requisitos no funcionales [9] se ha utilizado para referirse a preocupaciones no relacionadas con la funcionalidad del software. Sin embargo, diferentes autores caracterizan esta diferencia en definiciones informales y desiguales. Por ejemplo, en [10] se resume una serie de definiciones de este tipo:

a) “Describir los aspectos no conductuales de un sistema, capturando las propiedades y restricciones bajo las cuales un sistema debe operar. “

b) “Los atributos generales requeridos del sistema, incluyendo portabilidad, confiabilidad, eficiencia, ingeniería humana, capacidad de prueba, comprensibilidad y modificabilidad”.

c) “Requisitos que no se refieren específicamente a la funcionalidad de un sistema. Imponen restricciones al producto que se está desarrollando y al proceso de desarrollo, y especifican restricciones externas que el producto debe cumplir”.

d) “...requisitos globales sobre su desarrollo u costo operativo, desempeño, confiabilidad, mantenibilidad, portabilidad, robustez, y similares. ... No existe una definición formal ni una lista completa de requisitos no funcionales”.

e) “Las propiedades de comportamiento que deben tener las funciones especificadas, tales como rendimiento, usabilidad”.

f) “Una propiedad, o cualidad, que debe tener el producto, como una apariencia, o una propiedad de velocidad o precisión”.

g) “Una descripción de una propiedad o característica que un sistema de software debe exhibir o una restricción que debe respetar, distinta de un comportamiento observable del sistema”.

Después de argumentar que las definiciones no son claras y carecen de consenso, el autor dice: “Para las personas que no quieran deshacerse del término 'requisito no funcional', podemos definir este término adicionalmente como: DEFINICIÓN. Un requisito no funcional es un atributo o una restricción de un sistema”. [10].

Hay otras definiciones adicionales en la literatura que vale la pena agregar a la lista.

h) "... tipos de inquietudes: inquietudes funcionales asociadas con los servicios que se brindarán, y inquietudes no funcionales asociadas con la calidad del servicio, como seguridad, precisión, desempeño, etc.". [2].

i) "El término "requisito no funcional" se utiliza para delinear requisitos que se centran en "qué tan bien" el software hace algo, en contraposición a los requisitos funcionales, que se centran en "qué" hace el software". [11].

> j) “Dicho de otra manera, los NFR constituyen las justificaciones de las decisiones de diseño y limitan la forma en que se puede realizar la funcionalidad requerida”. [12].

A propósito, dejamos la cita a [12] como última definición de las varias presentadas. Esta definición de requisitos no funcionales es de gran importancia y se comentará más adelante en la Sección 4.

Dado que estamos revisando tantas definiciones, podría ser útil centrarnos en la definición de cuatro palabras que parecen centrales para todas las definiciones: calidad, funcionalidad, funcional y no funcional. Las definiciones fueron seleccionadas de WordNet [13]. Word-Net es una base de datos léxica de inglés, donde sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos (synsets), cada uno de los cuales expresa un concepto distinto. Aquí enumeramos la definición principal de cada una de estas palabras tal como aparecen en Wordnet. Hacemos esto para llamar la atención sobre el término no funcional.

Calidad: Sustantivo -- S: (n) cualidad (un atributo esencial y distintivo de algo o alguien)

Funcional: Adjetivo -- S: (adj) funcional (diseñado o capaz de una función o uso particular)

No funcional: Adjetivo -- S: (adj) no funcional (no tener o realizar una función); S: (adj) mal funcionamiento, no funcional (no realiza o no puede realizar su función habitual)

Funcionalidad: Sustantivo -- S: (n) funcionalidad (capaz de cumplir bien un propósito)

Si examinamos cuidadosamente estas definiciones, notamos que la palabra "funcional" es un adjetivo y "calidad" es a la vez sustantivo y adjetivo, mientras que "funcionalidad" es un sustantivo. También notamos que "funcional" se refiere al uso y "funcionalidad" se refiere al propósito.

Utilizamos estas definiciones, ya que si entendemos los términos “requisitos funcionales” y “requisitos no funcionales” fuera de contexto, pueden generar semánticas diferentes. Por supuesto, el término "requisitos no funcionales" no significa requisitos que no son capaces de realizar una función, pero si se interpreta fuera de contexto, puede crear confusión. Si la literatura hubiera sido más cuidadosa al elegir los nombres, esta posible confusión podría haberse evitado.

A pesar de estas observaciones y del hecho de que la funcionalidad también puede verse como calidad, como también se señala en [6], entendemos que la distinción entre estos dos tipos de calidad es extremadamente útil e importante en la ingeniería de software.

Un punto central para la distinción entre funcionalidad y otras cualidades es que, para la construcción de software, el propósito del sistema de software debe estar bien definido en términos de las funciones que realizará el software. Puede parecer extraño, pero esta distinción no es tan evidente en otras áreas de la ingeniería, como pudimos ver en la estrategia QFD. En los negocios y la ingeniería, la función de un artefacto o una actividad tiene que ver principalmente con entidades físicas y suele ser clara y directa. En cambio, en la ingeniería de software cuyos productos son entidades conceptuales, este no es el caso. De hecho, sería extraño detallar, por funciones, el hecho de que un automóvil tiene que poder transportar personas, pero, para construir un sistema de software, un ingeniero de software tiene que entender qué funciones debe realizar. , generalmente con mayor dificultad ya que no son evidentemente visibles, mensurables, táctiles, etc. Además, el software es tan

se están aplicando rápidamente a nuevas áreas de aplicación que no es posible para un ingeniero de software construir siempre sobre la base de las experiencias. Es un hecho bastante conocido que un producto de software puede estar dirigido a un dominio que un ingeniero de software no conoce, un problema al que normalmente otros tipos de ingenieros no tienen que enfrentarse.

La distinción entre funcionalidad y otras cualidades en el campo de la ingeniería de requisitos tiene un beneficio importante: deja claro a los ingenieros de software que los requisitos están destinados a tratar con atributos de calidad y no sólo con uno de ellos. A medida que la industria del software se hizo más madura y los ingenieros de software exploraron diferentes dominios, quedó más claro que no sería suficiente ocuparse únicamente de la descripción de la funcionalidad deseada, sino que también se debían pensar cuidadosamente los atributos de calidad desde el principio.

Entonces, en presencia de tantas definiciones diferentes de NFR, ¿cómo debemos proceder? Queremos que nuestra definición de trabajo sea coherente y se adapte a otras definiciones. Como definición de trabajo, comenzamos con la definición coloquial de NFR, como en el marco NFR [1] [3], es decir, cualquier “-ilidad”, “-idad”, junto con muchas otras cosas que no necesariamente terminan con cualquiera de ellos, como el rendimiento, la facilidad de uso y la coherencia, así como preocupaciones sobre la productividad, el tiempo, el coste y la felicidad personal. En vista de las funciones matemáticas, en forma de,

$$f: yo \rightarrow oh \text{(por ejemplo, suma: int x int} \rightarrow \text{En t)},$$

casi cualquier cosa que aborde las características de$f$, yo, oo relaciones entre$I$yoh serán considerados NFR. Por ejemplo, si la función de suma se puede encontrar fácilmente en una calculadora, si la función se puede construir o modificar fácilmente, de manera rentable y en tiempo, si la función devuelve el resultado rápidamente, quién puede ver la función, el entradas, o la salida, por ejemplo.

3 Algunos esquemas de clasificación

Como se vio en la sección anterior, varios trabajos proporcionan formas de distinguir entre diferentes tipos de preocupaciones sobre la calidad. Una es la distinción entre calidad básica y extra [6]. Otra es la distinción entre preocupaciones (subatributos de un atributo de calidad), factores (o deficiencias, posibles propiedades del sistema, como políticas y mecanismos integrados en el sistema, que tienen un impacto en las preocupaciones) y métodos (medios utilizados). por el sistema para lograr las preocupaciones) [4].

También destaca la norma ISO/IEC 9126 [14] que distingue 4 tipos de niveles de calidad: calidad en uso, calidad externa, calidad interna y calidad de proceso. Basado en estos tipos, [11] proporciona una clasificación orientada a procesos compuesta por:

1) “La identificación de NFR desde diferentes puntos de vista y diferentes niveles de detalle”.
2) “El apoyo para descubrir dependencias y conflictos entre ellos, y discutirlos y priorizarlos en consecuencia”.
3) “La documentación de NFR y la evaluación de esta documentación”.
4) “El apoyo para identificar medios para satisfacer el NFR, evaluar y discutir los medios y tomar decisiones de compensación en consecuencia. Esto incluye la estimación de costos.”, y
5) “El apoyo al cambio y la gestión de proyectos”.

Otra propuesta se hace en [15], utilizando los conceptos del Marco NFR [3], sobre una clasificación de metas y objetivos blandos, impulsada por la “perspectiva no funcional”. Esta clasificación proporciona 4 categorías: objetivos duros funcionales, objetivos duros no funcionales, objetivos blandos funcionales y objetivos blandos no funcionales.

Otro esquema de clasificación se introduce en [16]:

- Requisitos de interfaz: describir cómo debe interactuar el sistema con su entorno, usuarios y otros sistemas. Por ejemplo, interfaces de usuario y sus cualidades (por ejemplo, facilidad de uso).
- Requisitos de desempeño: describir las restricciones de desempeño que involucran
  - límites de tiempo/espacio, como cargas de trabajo, tiempo de respuesta, rendimiento y espacio de almacenamiento disponible. Por ejemplo, "el sistema debe manejar 100 transacciones por segundo".
  - confiabilidad que involucra la disponibilidad de los componentes y la integridad de la información mantenida y suministrada al sistema. Por ejemplo, "el sistema debe tener menos de 1 hora de inactividad cada 3 meses". seguridad, como los flujos de información permitidos.
  - capacidad de supervivencia, como la resistencia del sistema bajo fuego y catástrofes naturales.
- Requisitos operativos: incluyen limitaciones físicas (tamaño, peso), disponibilidad de personal, consideraciones sobre el nivel de habilidad, accesibilidad del sistema para mantenimiento, etc.
- Requisitos del ciclo de vida: se pueden clasificar en dos subcategorías:
  - Calidad del diseño: medida en términos tales como mantenibilidad, mejora y portabilidad.
  - límites al desarrollo, como limitaciones de tiempo de desarrollo, disponibilidad de recursos, estándares metodológicos, etc.
- Requerimientos económicos: costos inmediatos y/o a largo plazo
- Requisitos políticos

La Figura 1 muestra un árbol de calidad del software [17] que tiene como objetivo abordar las preocupaciones sobre los tipos clave de NFR y, lo que es más importante, las posibles correlaciones entre ellos.

FURPS es un acrónimo que representa un modelo para clasificar atributos de calidad del software o requisitos no funcionales, desarrollado en Hewlett-Packard, y luego se añadió +, de ahí FURPS+, para ampliar el acrónimo y enfatizar varios atributos [18]:

- Funcionalidad: conjunto de características, capacidades, generalidad, seguridad
- Usabilidad - Factores humanos, Estética, Consistencia, Documentación
- Fiabilidad: frecuencia/gravedad del fallo, recuperabilidad, previsibilidad, precisión, tiempo medio hasta el fallo
- Rendimiento: velocidad, eficiencia, consumo de recursos, rendimiento, tiempo de respuesta
- Soportabilidad: capacidad de prueba, extensibilidad, adaptabilidad, mantenibilidad, compatibilidad, configurabilidad, capacidad de servicio, instalabilidad, localización, portabilidad.

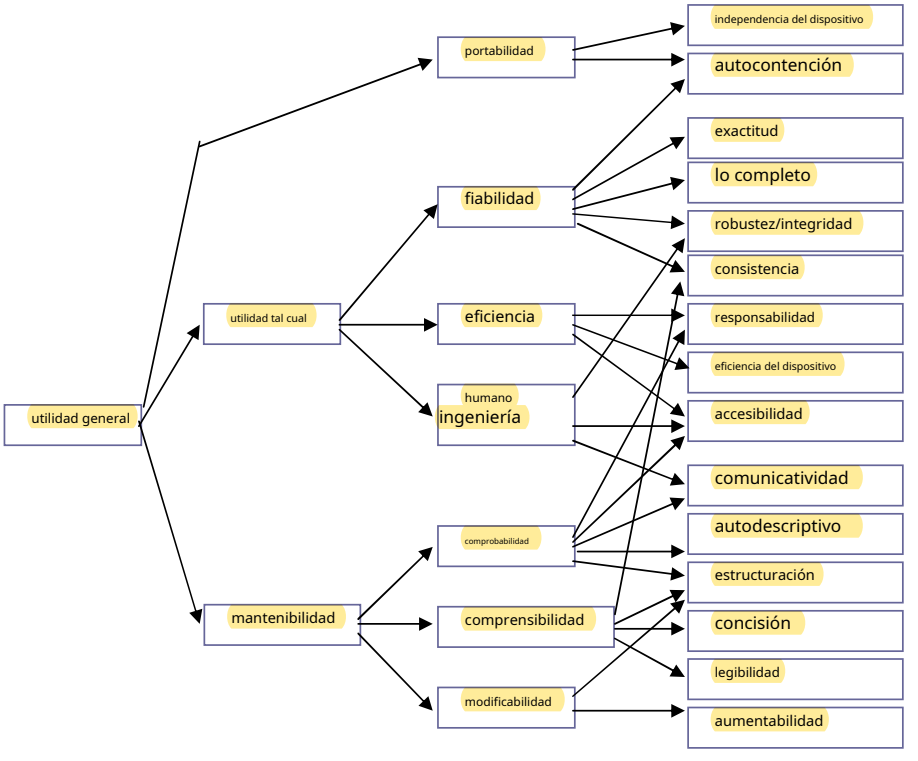

Figura 1.Árbol de calidad del software [17]

Sin embargo, incluso estos esquemas de clasificación tan conocidos son incompatibles entre sí. Por ejemplo, considere el rendimiento. En el esquema de clasificación de Roman, se define en términos de 4 subcategorías, mientras que ni siquiera aparece en el árbol de calidad del software, mientras que se muestra de forma bastante diferente en FURPS+.

Otra observación es que ni Roman ni FURPS+ reconocen interacciones potenciales entre NFRS, mientras que el árbol de calidad del software lo hace hasta cierto punto. Por ejemplo, en el árbol de calidad del software, la portabilidad y la confiabilidad están relacionadas entre sí a través de un subconcepto común de autocontención.

Se pueden hacer observaciones similares sobre otros tipos de NFR además del rendimiento, no sólo en relación con los pocos esquemas de clasificación que se muestran aquí sino también con muchos otros que no se muestran aquí, incluido el de [19].

No es sorprendente que los NFR desempeñen un papel crítico en el área del diseño arquitectónico, y se proporciona un esquema de clasificación en el contexto de las evaluaciones ATAM [20], por ejemplo, distinción de las cualidades del tiempo de ejecución (disponibilidad, rendimiento, seguridad) de las cualidades del tiempo de desarrollo (modificabilidad). , integración). ATAM también aborda el riesgo, al tiempo que considera una jerarquía de arquitectura, proceso y organización.

Ahora bien, ¿qué debería hacer entonces un profesional del software, cuando incluso los esquemas de clasificación bien conocidos son inconsistentes entre sí, no sólo terminológicamente sino también categóricamente?

Un profesional del software debe conocer algunos de los esquemas de clasificación más conocidos, como ISO 9126, un estándar internacional para la evaluación de la calidad del software, y considerar uno o más para adoptar con cierta adaptación. Independientemente del esquema de clasificación que un profesional de software decida adoptar, lo más importante a tener en cuenta es que debe saber qué quiere decir con un término NFR, como rendimiento, de modo que el significado de dicho NFR se puede comunicar con el usuario así como con los desarrolladores de sistemas/software para que el producto final se comporte como se espera.

4 Representaciones de requisitos no funcionales

Los requisitos relacionados con los NFR suelen estar separados de los requisitos de funcionalidad. Una forma habitual de representarlos es mediante frases de requisitos, que se enumeran por separado en diferentes apartados de la sección de requisitos técnicos. El estándar IEEE 830 (Práctica recomendada para especificaciones de requisitos de software) es un buen ejemplo, donde la Sección 3.2 se utiliza para especificar requisitos funcionales, mientras que el resto de la Sección 3 se utiliza para describir diferentes tipos de NFR.

Algunos autores proponen una estructura en torno a las oraciones de requisitos como la propuesta por [21] que se compone de: número de identificación, tipo de NFR, caso de uso relacionado con él, descripción, justificación, originador, criterio de ajuste, satisfacción del cliente, insatisfacción del cliente, prioridad. , conflictos, material de apoyo e historia. Todos estos son información textual informal.

En el trabajo clásico con SADT [22], una definición de requisitos debe responder a tres tipos de preguntas: por qué se necesita un sistema (análisis de contexto), qué características del sistema servirán para satisfacer este contexto (requisitos funcionales del sistema) y cómo se implementa el sistema. a construir (requisitos no funcionales del sistema). Aunque no hay una referencia explícita a los requisitos orientados a la funcionalidad frente a los requisitos de calidad, los actigramas de SADT se pueden utilizar para abordar indirectamente algunas de las preocupaciones de NFR. Un actigrama se puede asociar con cuatro tipos de información que interactúan con la actividad: entrada, control, salida y mecanismo. El espíritu de la información de control está muy relacionado con la idea de no funcionalidad, ya que la flecha de control en SADT tiene el propósito de limitar cómo se realiza la actividad. Como tal, SADT proporciona una manera de abordar atributos o limitaciones de calidad.

Los NFR también se representan comúnmente mediante árboles, que expresan el concepto de agrupación o descomposición de NFR [4] y también mediante listas.

Algunos autores han utilizado NFR junto con una notación de representación de requisitos más estructurada, por ejemplo, la combinación de NFR con casos de uso o casos de uso indebido (por ejemplo, ver [23], [24],[25], [26] y [27).]).

Los NFR también se representan como restricciones sobre diferentes partes de un escenario, junto con el tiempo y la ubicación como información contextual en la descripción del escenario [28]. Aquí, la representación de un escenario se compone de: título, objetivo, contexto, recursos, actores, episodios, excepciones y la restricción de atributo, que se aplica al contexto.

recursos y episodios. El contexto de la entidad se divide a su vez en ubicación geográfica, tiempo y condición previa.

Sin embargo, entre muchas propuestas, los enfoques orientados a objetivos, como en [2] [3], fueron los primeros en tratar los NFR con más profundidad.

En KAOS [2], que tal vez fue pionero en promover la ingeniería de requisitos orientada a objetivos, al menos desde una perspectiva de objetivos funcionales, los objetivos son: "... modelados por características intrínsecas como su tipo y atributos, y por sus vínculos con otros objetivos y con otros objetivos". elementos de un modelo de requisitos". KAOS aborda tanto objetivos funcionales como no funcionales, que se formalizan en términos de operadores, como Lograr, Mantener y Evitar, y mediante actividades basadas en lógica temporal aumentada con algunos operadores temporales especiales. Por ejemplo, KAOS ofrece operadores especiales para conceptos como "en algún momento en el futuro" y "siempre en el futuro a menos". Gracias a su naturaleza formal, las representaciones en KAOS son susceptibles de verificación y razonamiento automático. En KAOS, los objetivos se caracterizan como un conjunto de restricciones de alto nivel sobre los estados. Por ejemplo, un objetivo informal: {{Objetivo Mantener [DoorsClosedWhileMoving]}} [2] se puede expresar mediante la siguiente fórmula:

{{∀ tr: Tren, loc, loc': Ubicación En (tr, loc)∧o En (tr, loc')∧ubicación <> ubicación'⇨ tr.Puertas = 'cerrado'∧o (tr.Puertas = 'cerrado')}}.

La fórmula anterior, donde "o" es un operador temporal que denota el siguiente estado, significa que mientras un tren se mueve de un lugar a otro, sus puertas deben estar cerradas durante el movimiento.

Aunque el lenguaje de representación de KAOS no diferencia entre objetivos funcionales y no funcionales, el gráfico AND/OR de KAOS hace la diferenciación. Los objetivos que pueden asignarse a agentes individuales no necesitan mayor descomposición y pueden “operacionalizarse”, es decir, pueden describirse en términos de restricciones.

Al igual que KAOS, el Marco NFR también promueve la orientación a objetivos, pero con énfasis principal en los NFR. En el Marco NFR, los requisitos no funcionales se tratan como objetivos blandos, es decir, objetivos que deben abordarse no de manera absoluta sino en un sentido suficientemente bueno. Esto es un reconocimiento de las dificultades asociadas tanto con el problema como con la solución correspondiente. En cuanto al planteamiento del problema, a menudo resulta extremadamente difícil, si no imposible, definir un término NFR de forma completamente inequívoca sin utilizar ningún otro término NFR, que a su vez tendrá que definirse. En cuanto a la solución, a menudo también resulta extremadamente difícil explorar una lista completa de posibles soluciones y elegir la mejor u óptima solución, debido a diversas limitaciones de recursos, como el tiempo, la mano de obra y el dinero disponibles para dicha exploración.

Reflejando el sentido de “suficientemente bueno”, el Marco NFR introduce la noción de satisfacer y, con esta noción, se dice que una meta blanda satisface (en lugar de satisfacer) otra meta blanda. El Marco NFR ofrece varios tipos diferentes de contribuciones mediante las cuales un objetivo blando satisface, o niega, otro objetivo blando: HACER, AYUDAR, DAÑAR y ROMPER son los más destacados, así como Y y O (estos también incorporan la noción de “suficientemente bueno” en lugar de que “satisfacción absoluta”). HACER y AYUDAR se utilizan para representar un objetivo blando que satisface positivamente a otro, mientras que ROMPER y DAÑAR para representar un objetivo blando que satisface (o niega) negativamente a otro. Mientras HACER y

DESCANSO reflejan respectivamente nuestro nivel de confianza en que un objetivo blando satisfaga o niegue completamente a otro, AYUDA y DAÑO reflejan respectivamente nuestro nivel de confianza en que un objetivo blando satisfaga o niegue parcialmente otro.

En el Marco NFR, cada objetivo blando o contribución está asociado con una etiqueta, que indica el grado en que se cumple o se niega. En el Marco NFR se ofrece un procedimiento de propagación de etiquetas para determinar el efecto de varias decisiones de diseño, independientemente de si son a nivel de sistema o de software. Además de una etiqueta, cada objetivo o contribución también se puede asociar con un valor de criticidad, como extremadamente crítico, crítico y no crítico.

Cuando se utiliza el Marco NFR, los NFR se publican como objetivos blandos que deben abordarse o lograrse, y se aplica un proceso iterativo y entrelazado para satisfacerlos, a través de descomposiciones, operacionalizaciones y argumentaciones Y/O. A lo largo del proceso, se crea y mantiene una representación visual, SIG (gráfico de interdependencia de objetivos blandos), que realiza un seguimiento de los objetivos blandos y sus interdependencias, junto con el impacto de diversas decisiones a través de etiquetas. En este sentido, un SIG muestra cómo se racionalizan diversas decisiones (de diseño).

Para aliviar las dificultades asociadas con la búsqueda de conocimiento para lidiar con NFR, el Marco NFR ofrece métodos para capturar conocimiento sobre formas de satisfacer NFR y reglas de correlación para conocer los efectos secundarios que tales métodos inducen.

Al igual que con los objetivos en KAOS, los objetivos blandos en el Marco NFR están asociados con las acciones de los agentes (humanos, hardware o software) y, en última instancia, se logran mediante ellas. Esto es consistente con el espíritu del modelo de referencia [29], en el que los requisitos (funcionales) se satisfacen a través de la colaboración entre el comportamiento del sistema de software y los fenómenos ambientales causados por agentes en el entorno, aunque aquí también nos preocupamos por los objetivos blandos. que los requisitos (funcionales) están destinados a ayudar a lograr junto con los fenómenos ambientales. Tenga en cuenta que los requisitos son parte de la solución a algún problema en una parte de la realidad, y la noción de objetivos blandos se puede utilizar para representar cualquier cosa que no sea funcional, ya sea sobre el dominio del problema o la solución.

La familia i*: i* [30], Tropos [31] y GRL (Lenguaje de requisitos de objetivos) [32] heredaron el concepto de softgoal del NFR Framework, con el objetivo de tratar objetivos blandos, o atributos no relacionados con la funcionalidad, como un concepto de modelado de primera clase.

Como se mencionó en la Sección 2, la última definición [12] presentada fue algo diferente del resto: una NFR se describe como una justificación de una decisión de diseño y como una restricción en la forma en que se puede realizar una funcionalidad requerida. Esta es exactamente la razón por la que la identificación y representación adecuadas de los NFR desempeñan un papel clave en la ingeniería de software, ya que se dice que la ingeniería de software tiene que ver con la toma de decisiones.

Como han señalado varios autores, las NFR necesitan transformarse a través de algún medio, método u operación. Esta es también la razón por la cual una representación orientada a objetivos es tan adecuada para los NFR: inicialmente se expresan en términos generales como requisitos más abstractos, pero luego gradualmente se van refinando en términos y detalles más concretos.

Cuando las NFR eventualmente se operacionalizan, en términos de operaciones, entidades o restricciones de software, se convierten en la justificación de por qué existen tales operacionalizaciones en el sistema de software, es decir, para servir los atributos de calidad especificados como objetivos blandos de NFR. Si los ingenieros de software son lo suficientemente cuidadosos como para mantener el historial del

construcción de software, entonces podrán explicar y justificar por qué existen tales operacionalizaciones. No hace falta decir que este argumento también es válido para la funcionalidad, pero el punto clave aquí es que la elección de una operación específica para materializar un problema de calidad afecta cómo se logra la funcionalidad general. Dicho de otra manera, se toman diferentes tipos de decisiones de diseño a lo largo de un proceso de desarrollo de software, y los NFR actúan como criterio para dichas decisiones de diseño (consulte [33] para una discusión sobre estas decisiones de diseño en el contexto de los casos de uso). Como se argumenta en [12], una NFR no es sólo una justificación a posteriori, sino que limita cómo se realiza la funcionalidad.

Entonces, si el ingeniero de software utiliza atributos de calidad desde el principio durante el proceso de desarrollo de software, habrá una red de explicaciones, limitadas por el uso de la racionalidad, que vinculan los resultados de las decisiones con los atributos de calidad. El trabajo sobre la justificación del diseño [34], basándose en trabajos anteriores sobre la idea de Ibis [35], se centra en la justificación de las decisiones, pero sin tener en cuenta los factores preexistentes que conducen a la decisión. El trabajo sobre la justificación del diseño puede beneficiarse de mejores formas de abordar la naturaleza dinámica y contextual del diseño de software y las limitaciones de trabajar con la multitud de posibles alternativas y sus justificaciones [36].

Se dice que la integración de funcionalidad y otras cualidades es esencial. Aunque nadie cuestionaría esa perogrullada, pocos han propuesto o defendido un proceso que realmente entrelaza estas dos clases de requisitos. Los métodos orientados a objetivos, como el Framework NFR, KAOS y la familia i*, son las pocas excepciones que hacen que la no funcionalidad sea considerada como un concepto de primera clase, estando entrelazada con la funcionalidad, ya que están cosificadas. Sin embargo, no debería sorprender que cada enfoque tenga sus propias formas particulares de entrelazar atributos de funcionalidad y calidad, con varias variaciones distintas y sin necesariamente mantener el historial de desarrollo.

¿Así que lo que? En pocas palabras, el punto que estamos tratando de señalar es doble: no sólo los requisitos no funcionales deben establecerse desde el principio, sino que también pueden ayudar al ingeniero de software a tomar decisiones de diseño, al mismo tiempo que justifican dichas decisiones. Sin embargo, para tener esto en cuenta, es necesario que los atributos de calidad no se consideren simplemente como un conjunto separado de requisitos, sino que se considere la funcionalidad durante todo el proceso de desarrollo.

El Marco NFR y la familia i* tienen una característica intrínseca que los diferencia de otros métodos NFR: la dependencia de un enfoque cualitativo hacia los NFR o objetivos blandos. En el corazón del Marco NFR se encuentra el concepto de que los objetivos blandos son idealizaciones y, como tales, sin que todas sus propiedades definitorias estén necesariamente establecidas. Esta característica es similar a la noción de “racionalidad limitada” propuesta por Herbert Simon [37] cuando explica su comprensión del proceso que utilizan los diseñadores para tomar una decisión dada información incompleta. Esta característica cualitativa se basa en las ideas de contribución y correlación entre objetivos blandos, como se explicó anteriormente. Por medio de contribuciones, un objetivo blando puede descomponerse hasta el nivel de operacionalización y, mediante correlaciones, diferentes objetivos blandos pueden interferir entre sí. Una red de tales contribuciones y correlaciones permite llevar a cabo diferentes tipos de razonamiento cualitativo. La semántica de dicha red está dada por relaciones en tres dimensiones a) descomposiciones sobre un árbol AND/OR, b) contribuciones entre subárboles y c) correlaciones entre diferentes objetivos blandos, todo lo cual conduce a la formación de un gráfico de interdependencia de objetivos blandos (SIG).).

Si bien el marco NFR se centra en las NFR, como se describe al final de la Sección 2, los NFR existen en relación con cosas funcionales. UML es un lenguaje de diseño y análisis orientado a objetos funcionalmente orientado, y uno de los primeros trabajos que detalla cómo el marco NFR podría ayudar a lograr mejores modelos UML se describe en [38], que presenta un proceso para vincular gráficos NFR con modelos UML. . La idea central es realizar cualitativamente objetivos blandos en los modelos UML. La realización de clases UML, por ejemplo, se basó en la introducción de atributos a las clases, métodos o restricciones sobre los atributos.

En i* [30], los vínculos entre softgoals y operaciones (tareas o recursos) son más explícitos, ya que el modelado se lleva a cabo simultáneamente en el contexto del diagrama de Racionalidad Estratégica (SR). En un diagrama SR, las relaciones medios-fines (operador de especialización i*) pueden mostrar cómo las opciones (tareas) se relacionan con diferentes objetivos suaves, al mismo tiempo que muestran los pros y los contras de cada selección.

El Marco NFR presentado en esta Sección se adapta a cualquier esquema de clasificación que se analizó en la Sección 3, a través de descomposiciones Y/O, y va más allá al ofrecer esos conceptos de operacionalizaciones y argumentaciones, junto con correlaciones positivas/negativas. En términos de estos conceptos, el Marco NFR ayuda a racionalizar las decisiones de diseño, tanto a nivel de sistema como de software. Para la representación de los NFR, reconoce los NFR como objetivos blandos y las relaciones entre ellos como contribuciones parciales/totalmente positivas/negativas.

5 direcciones futuras

Ha habido varios trabajos que exploraron más a fondo los conceptos de objetivos blandos, al tiempo que dieron forma a algún escenario para direcciones futuras. Los clasificamos en seis áreas: variabilidad del software, análisis de requisitos, obtención de requisitos, reutilización de requisitos, trazabilidad de requisitos y desarrollo orientado a aspectos. Cada una de estas áreas ha explorado aspectos particulares de la idea de un SIG (Softgoal Interdependency Graph).

Cuando se trata de líneas de productos de software, la idea de variabilidad es crítica, ya que, en algunas partes de una línea de productos, existirán puntos de variación de arquitectura para permitir la producción de diferentes alternativas necesarias para componer diferentes productos a partir de una única arquitectura común. Los trabajos de [39] [40] [41] exploraron el hecho de que las alternativas son intrínsecas en los modelos SIG, ya que son gráficos Y/O. El uso de un enfoque orientado a objetivos para las líneas de productos ofrece una forma fluida de producir arquitecturas de líneas de productos, ya que las características no sólo se identifican, sino que también se justifican, en términos de objetivos blandos.

Una de las principales ventajas del enfoque cualitativo de un SIG es que facilita el análisis. La idea misma de que puede haber diferentes tipos de relaciones entre objetivos blandos y entre objetivos blandos y operacionalizaciones en un SIG brinda la oportunidad de realizar análisis, propagando el impacto de las decisiones a lo largo de las relaciones de contribución de correlación. Utilizando el concepto de propagación de etiquetas sobre un gráfico SIG, es posible evaluar cómo una operacionalización determinada de un NFR afectará a todo el gráfico. El proceso original fue ideado en el Marco NFR [3] y siguieron variaciones [42] [43]. Utilizando la idea de propagación de etiquetas, se podrían realizar diferentes tipos de análisis desde el principio antes de comprometerse con una arquitectura o un código, como se ve en la exploración de preocupaciones de seguridad [44], en la elección visual de operacionalizaciones [45] y en la formulación de una i * análisis de modelos como problema SAT [46].

Obtener requisitos requiere el uso de diferentes conjuntos de técnicas, pero la mayoría de ellas se centran únicamente en descubrir la funcionalidad. El trabajo sobre la obtención de objetivos debe considerar tanto la funcionalidad como otras cualidades. El trabajo en este sentido incluye una propuesta sobre un esquema de obtención que parte de un léxico extendido [47], técnicas de cuadrícula de repertorio para ayudar a clarificar la denominación durante el proceso de obtención [48] y una teoría del constructo personal para obtener la contribución entre objetivos blandos [49]. .

El marco NFR [3] ha identificado que los catálogos NFR, compuestos por gráficos SIG, eran un aspecto importante en la creación de software utilizando el concepto softgoal. Un trabajo posterior [50] exploró más a fondo las ideas al enfatizar el aspecto de la reutilización, al tiempo que propuso un método para mantener una organización con objetivos blandos destinada a la reutilización. La idea de una trazabilidad centrada en objetivos basada en gráficos de objetivos blandos se explora en [51] [52], en el que la red de objetivos blandos se utiliza como base para explicar los cambios en la evolución del software.

La relación entre los atributos de calidad y el desarrollo orientado a aspectos se ha explorado en [53] y [54]. Más tarde, otros han identificado el importante papel que desempeña el Marco NFR - en particular, los objetivos blandos - al abordar los primeros aspectos [27], [55] [56] [57] [58] [59] (Ver [60] para una encuesta sobre el tema).

Aunque estos resultados recientes ayudaron a comprender mejor el concepto general de requisitos de calidad y abrieron nuevos caminos para futuras investigaciones, también es necesario avanzar más en otras cuestiones, como la integración de los NFR en otros modelos de requisitos, como el modelo de referencia [29] y el modelo de cuatro variables [61] que han tenido influencias significativas en el área de la ingeniería de requisitos. Aunque estos modelos abordan cuestiones de rendimiento o precisión, son esencialmente modelos funcionales y sin la noción de objetivos. Como se mencionó brevemente en la Sección 4, por ejemplo, el modelo de referencia establece que los requisitos (funcionales) se satisfacen mediante la colaboración entre el comportamiento funcional del sistema de software y los fenómenos (funcionales) en el entorno. KAOS [2] va más allá de estos modelos funcionales e introduce tipos generales de objetivos suaves para el sistema en general, al tiempo que aborda cuestiones de rendimiento, precisión y seguridad para el sistema de software.

También estamos de acuerdo con [11] en la necesidad de realizar más investigaciones empíricas sobre el uso de NFR durante la ingeniería de requisitos y en el uso de estudios etnográficos sobre cómo los equipos de software abordan los problemas de calidad como requisitos. Entendemos que esta investigación debe realizarse con proyectos reales, tanto en situaciones de laboratorio como en proyectos industriales, para mejorar nuestro conocimiento sobre cómo abordar problemas de calidad desde el principio.

Agradecimientos. Agradecemos los comentarios de Barbara Paech sobre un borrador anterior, que nos ayudaron significativamente a mejorar el documento de una manera más comprensible.

Referencias

1. Mylopoulos, J., Chung, L., Nixon, B.: Representación y uso de requisitos no funcionales. mentos: un enfoque orientado a procesos. Traducción IEEE. Software. Ing. 18(6), 483–497 (1992), http://dx.doi.org/10.1109/32.142871
2. van Lamsweerde, A.: Ingeniería de requisitos orientada a objetivos: una visita guiada. En: Actas del 5º Simposio internacional IEEE sobre ingeniería de requisitos, 27 al 31 de agosto de 2001, p. 249. Sociedad de Computación IEEE, Washington (2001)

3. Chung, L., Nixon, BA, Yu, E., Mylopoulos, J.: Requisitos no funcionales en ingeniería de software. Serie internacional en ingeniería de software, vol. 5, pág. 476. Springer, Heidelberg (1999)
4. Barbacci, M., Longstaff, TH, Klein, MH, Weinstock, CB: Quality Attributes, Informe técnico CMU/SEI-95-TR-021, ESC-TR-95-021 (diciembre de 1995)
5. Estándar IEEE 1061-1992 Estándar para una metodología de métricas de calidad del software. Instituto de Ingenieros Eléctricos y Electrónicos, Nueva York (1992)
6. Freeman, PA: Perspectivas del software: el sistema es el mensaje. Addison-Wesley, Lectura (1987)
7. Instituto QFD, Despliegue de la Función de Calidad,http://www.qfdi.org/
8. Hauser Jr., Clausing, D.: La casa de la calidad. Harvard Business Review 66(3), 63–73 (1988)
9. Yeh, RT, Zave, P., Conn, AP, Cole, GE: Análisis de requisitos de software: nuevas direcciones y perspectivas. En: Vick, CR, Ramamoorthy, CV (eds.) Manual de ingeniería de software, Van Nostrand Reinhold Co. (1984)
10. Glinz, M.: Sobre requisitos no funcionales. En: 15ª Conferencia Internacional de Ingeniería de Requisitos del IEEE (RE 2007), págs. 21-26 (2007)
11. Paech, B., Kerkow, D.: Ingeniería de requisitos no funcionales: la calidad es esencial. En: Taller internacional del décimo aniversario sobre ingeniería de requisitos: Fundación para la Calidad del Software, REFSQ 2004 (2004),http://www.sse.uni-essen.de/refsq/downloads/toc-refsq04.pdf
12. Landes, D., Studer, R.: El tratamiento de los requisitos no funcionales en MIKE. En: Botella, P., Schäfer, W. (eds.) ESEC 1995. LNCS, vol. 989, págs. 294–306. Springer, Heidelberg (1995)
13. Una base de datos léxica de inglés,http://wordnet.princeton.edu/
14. ISO/IEC 9126-1:2001(E): Ingeniería de software - Calidad del producto - Parte 1: Modelo de calidad (2001)
15. Jureta, IJ, Faulkner, S., Schobbens, P.-Y.: Una conceptualización de objetivos blandos más expresiva para el análisis de requisitos de calidad. En: Embley, DW, Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, págs. 281–295. Springer, Heidelberg (2006)
16. Roman, G.-C.: Una taxonomía de problemas actuales en ingeniería de requisitos. Computadora IEEE, 14-21 (abril de 1985)
17. Boehm, BW, Brown, JR, Kaspar, H., Lipow, M., MacLeod, GJ, Merritt, MJ: Características de la calidad del software. Holanda Septentrional, Ámsterdam (1978)
18. Grady, R., Caswell, D.: Métricas de software: establecimiento de un programa para toda la empresa. Prentice-Hall, acantilados de Englewood (1987)
19. Bowen, TP, Wigle, GB, Tsai, JT: Especificación de atributos de calidad del software, Informe RADC-TR-85-37, vol. I (Introducción), vol. II (Guía de especificaciones de calidad del software), vol. III (Guía de evaluación de la calidad del software), Rome Air Development Center, Griffiss Air Force Base, Nueva York (febrero de 1985)
20. Bass, L., Nord, R., Wood, W., Zubrow, D.: Temas de riesgo descubiertos a través de evaluaciones de arquitectura, Informe técnico CMU/SEI-2006-TR-012, ESC-TR-2006-012 (2006)
21. Robertson, S., Robertson, J.: El proceso de requisitos de Volere, Dominar el proceso de requisitos. Addison-Wesley, Londres (1999)
22. Ross, DT: Análisis estructurado (SA): un lenguaje para comunicar ideas. Traducción IEEE. Software. Ing. 3(1), 16–34 (1977), http://dx.doi.org/10.1109/TSE.1977.229900
23. Chung, L., Supakkul, S.: Representación de nFR y fR: un enfoque orientado a objetivos y basado en casos de uso. En: Dosch, W., Lee, RY, Wu, C. (eds.) SERA 2004. LNCS, vol. 3647, págs. 29 a 41. Springer, Heidelberg (2006)
24. Herrmann, A., Paech, B.: MOQARE: ingeniería de requisitos de calidad orientada al mal uso. Requerir. Ing. 13(1), 73–86 (2008)

25. Cysneiros, LM, do Prado Leite, JC: Uso de UML para reflejar requisitos no funcionales. En: Stewart, DA, Johnson, JH (eds.) Actas de la Conferencia de 2001 del Centro de Estudios Avanzados sobre Investigación Colaborativa. Conferencia del Centro IBM de Estudios Avanzados, vol. 2. Prensa IBM (2001)
26. Alexander, I.: Los casos de uso indebido ayudan a generar requisitos no funcionales. Revista de ingeniería de control y computación 14 (1), 40–45 (2003)
27. de Sousa, TGMC, Castro, JFB: Hacia una metodología de requisitos orientada a objetivos basada en el principio de separación de preocupaciones. En: Anais do WER 2003 - Taller en Engenharia de Requisitos, Piracicaba-SP, Brasil, 27 y 28 de noviembre de 2003, págs. 223–239 (2003), http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER03/georgia_souza.pdf
28. Leite, JC, Hadad, G., Doorn, J., Kaplan, G.: Un proceso de construcción de escenarios. Revista de ingeniería de requisitos 5 (1), 38–61 (2000)
29. Gunter, C., Gunter, E., Jackson, M., Zave, P.: Un modelo de referencia para requisitos y especificaciones. Software IEEE, 37–43 (2000)
30. Yu, ESK: Hacia el soporte de modelado y razonamiento para la ingeniería de requisitos en las primeras fases. En: Actas del Tercer Simposio Internacional IEEE sobre Ingeniería de Requisitos, págs. 226-235 (1997)
31. Castro, J., Kolp, M., Mylopoulos, J.: Hacia la ingeniería de sistemas de información basada en requisitos: el proyecto Tropos. Sistemas de información 27 (6), 365–389 (2002)
32. Amyot, D., Mussbacher, G.: URN: Hacia un nuevo estándar para la descripción visual de requisitos. En: Sherratt, E. (ed.) SAM 2002. LNCS, vol. 2599, págs. 21–37. Springer, Heidelberg (2003)
33. Dutoit, AH, Paech, B.: Especificación de casos de uso basada en fundamentos. Ingeniería de requisitos 7 (1), 1–3 (2002)
34. Potts, C., Bruns, G.: Registro de los motivos de las decisiones de diseño. En: Actas de la Décima Conferencia Internacional sobre Ingeniería de Software. Conferencia internacional sobre ingeniería de software, Singapur, 11 al 15 de abril de 1988, págs. Prensa de la Sociedad de Computación IEEE, Los Alamitos (1988)
35. Kunz, W., Rittel, HWJ: Issues as Elements of Information Systems, documento de trabajo núm. 131 (julio de 1970); Studiengruppe für Systemforschung, Heidelberg, Alemania (reimpreso en mayo de 1979)
36. Dutoit, AH, McCall, R., Mistrík, I., Paech, B. (eds.): Gestión racional en ingeniería de software. Springer, Heidelberg (2006)
37. Simon, HA: Las ciencias de lo artificial, 3ª ed. Prensa del MIT, Cambridge, MA (1977)
38. Cysneiros, LM, Leite, JC: Requisitos no funcionales: de la elicitación a lo conceptual Modelos. Traducción IEEE. Software. Ing. 30(5), 328–350 (2004), http://dx.doi.org/10.1109/TSE.2004.10
39. Liaskos, S., Lapouchnian, A., Yu, Y., Yu, ESK, Mylopoulos, J.: Sobre la adquisición y el análisis de la variabilidad basada en objetivos. En: RE 2006, págs. 76–85 (2006)
40. Yu, Y., Lapouchnian, A., Liaskos, S., Mylopoulos, J., Leite, JC: De los objetivos al diseño de software de alta variabilidad. En: An, A., Matwin, S., Ras,ZW,Syomizak, D. (eds.) Fundamentos de los sistemas inteligentes. LNCS, vol. 4994, págs. 1-16. Springer, Heidelberg (2008)
41. González-Baixauli, B., Laguna, MA, Leite, JC: Uso de modelos de objetivos para analizar la variabilidad. En: Primer taller internacional sobre modelado de variabilidad de sistemas intensivos en software, VaMoS 2007, Actas, Limerick, Irlanda, 16-18 de enero de 2007, págs. 101-107, Informe técnico de Lero 2007-01 2007 (2007)
42. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Razonamiento con modelos de objetivos. En: Spaccapietra, S., March, ST, Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, págs. 167–181. Springer, Heidelberg (2002)

43. Kaiya, H., Horai, H., Saeki, M.: AGORA: Método de análisis de requisitos atribuidos orientado a objetivos. En: Actas de la Conferencia internacional conjunta del IEEE sobre ingeniería de requisitos del décimo aniversario, 9 al 13 de septiembre de 2002, págs. Sociedad de Computación IEEE, Washington (2002)
44. Liu, L., Yu, E., Mylopoulos, J.: Análisis de requisitos de seguridad y privacidad en un entorno social. En: Actas de la 11ª Conferencia internacional IEEE sobre ingeniería de requisitos, 8-12 de septiembre de 2003, IEEE Computer Society, Washington (2003)
45. González-Baixauli, B., Leite, JC, Mylopoulos, J.: Análisis de variabilidad visual para modelos de objetivos. En: Actas de la Conferencia de Ingeniería de Requisitos, 12ª edición internacional del IEEE tional, 6-10 de septiembre de 2004, págs. 198-207. Sociedad de Computación IEEE, Washington (2004), http://dx.doi.org/10.1109/RE.2004.56
46. Horkoff, J., Yu, ESK: Análisis cualitativo, interactivo y hacia atrás de modelos i*. En: iStar 2008, págs. 43–46 (2008)
47. Oliveira, APA, Leite, JC, Cysneiros, LM: AGFL - Objetivos de agentes de Lexicon - Obtención de intencionalidad de sistemas multiagente. En: iStar 2008, págs. 29-32 (2008)
48. Niu, N., Easterbrook, SM: Gestión de la interferencia terminológica en modelos de objetivos con Repertory Grid. En: RE 2006, págs. 296-299 (2006)
49. González-Baixauli, B., Leite, JC, Laguna, MA: Obtención de interacciones de requisitos no funcionales utilizando la teoría del constructo personal. En: RE 2006, págs. 340–341 (2006)
50. Cysneiros, LM, Werneck, V., Kushniruk, A.: Conocimiento reutilizable para satisfacer los requisitos de usabilidad. En: RE 2005, págs. 463–464 (2005)
51. Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., Christina, S.: Trazabilidad centrada en objetivos para la gestión de requisitos no funcionales. En: Actas de la 27ª Conferencia Internacional sobre Ingeniería de Software, ICSE 2005, St. Louis, MO, EE. UU., mayo 15-21, 2005, págs. 362–371. ACM, Nueva York (2005), http://doi.acm.org/10.1145/1062455.1062525
52. Cleland-Huang, J., Marrero, W., Berenbach, B.: Trazabilidad centrada en objetivos: uso de plomadas virtuales para mantener cualidades sistémicas críticas. Traducción IEEE. Software. Ing. 34(5), 685-699 (2008),http://dx.doi.org/10.1109/TSE.2008.45
53. Grundy, JC: Ingeniería de requisitos orientada a aspectos para sistemas de software basados en componentes. En: Actas del 4º Simposio internacional IEEE sobre ingeniería de requisitos, RE, 7 al 11 de junio de 1999, págs. Sociedad de Computación IEEE, Washington (1999)
54. Moreira, A., Araújo, J., Brito, I.: Atributos de calidad transversales para la ingeniería de requisitos. En: Actas de la 14ª Conferencia Internacional sobre Ingeniería de Software e Ingeniería del Conocimiento, SEKE 2002, Ischia, Italia, 15-19 de julio de 2002, vol. 27, págs. 167-174. ACM, Nueva York (2002),http://doi.acm.org/10.1145/568760.568790
55. Yu, Y., Leite, JC, Mylopoulos, J.: De las metas a los aspectos: descubrimiento de aspectos a partir de modelos de metas y requisitos. En: 12ª Actas internacionales del IEEE de la Conferencia de Ingeniería de Requisitos, 6 al 10 de septiembre de 2004, págs. Sociedad de Computación IEEE, Washington (2004),http://dx.doi.org/10.1109/RE.2004.23
56. Brito, I., Moreira, A.: Integración del marco NFR en un modelo de ER. En: EA-AOSD 2004: Taller sobre aspectos iniciales: ingeniería de requisitos y diseño arquitectónico orientado a aspectos, celebrado junto con la 3ª Conferencia Internacional sobre Orientado a Aspectos Desarrollo de software, Lancaster, Reino Unido, 22 al 26 de marzo (2004), http://trese.cs.utwente.nl/workshops/early-aspects-2004/ Papers/ BritoMoreira.pdf
57. Alencar, F., Silva, C., Moreira, A., Araújo, J., Castro, J.: Identificación de aspectos candidatos con el enfoque I-star. En: Aspectos tempranos 2006: Trazabilidad de aspectos en el ciclo de vida temprano, págs. 4-10 (2006)

58. de Padua Albuquerque Oliveira, A., Cysneiros, LM, do Prado Leite, JC, Figueiredo, EM, Lucena, CJ: Integración de escenarios, i* y AspectT en el contexto de sistemas multiagente. En: Actas de la Conferencia de 2006 del Centro de Estudios Avanzados sobre Investigación Colaborativa, CASCON 2006, Toronto, Ontario, Canadá, 16 al 19 de octubre de 2006, pag. 16. ACM, Nueva York (2006),http://doi.acm.org/10.1145/1188966. 1188988

59. da Silva, LF, Leite, JC: Generación de vistas de requisitos: un enfoque impulsado por la transformación. ECEASST 3 (2006)

60. Yu, Y., Niu, N., González-Baixauli, B., Mylopoulos, J., Easterbrook, S., Leite, JC: Ingeniería de requisitos y aspectos. En: Ingeniería de requisitos de diseño: una perspectiva de diez años. Apuntes de conferencias sobre procesamiento de información empresarial, págs. Springer, Heidelberg (2009)

61. Parnas, DL, Madey, J.: Documentación funcional para sistemas informáticos. Ciencia de la programación informática 25 (1), 41–61 (1995)