UN ESTUDIO INICIAL DE REQUERIMIENTOS NO FUNCIONALES A TRAVES DEL CASO DE USO[1]

DIGIÓN, Leda Beatriz
Univ. Nacional de Santiago del Estero, Facultad de Ciencias Exactas y Tecnologías.

Abstract
En este trabajo se investiga la definición inicial de requerimientos no funcionales, como parte del proceso de educción de requerimientos de un producto informático. Si no se considera la educción, documentación y registro de requerimientos no funcionales en el proceso de desarrollo de producto software, se hace más difícil priorizar el tratamiento de las diferencias de calidad que puedan surgir en el sistema en estudio. Recuérdese que la calidad es el grado de concordancia entre los requerimientos del producto y el producto actual; también, se define la calidad desde el punto de vista de la percepción del usuario, expectativas y objetivos o necesidades. Se toman las lecciones presentadas en el documento de Bredemeyer Consulting [4], y también se exponen conceptos introductorios de medición En este trabajo, se brinda conocimiento para identificar fuentes de requerimientos no funcionales y clasificarlos con la técnica ampliada de caso de uso, para lo cual se muestra un ejemplo. Así también, se pueden deducir atributos de calidad asociados a los objetivos técnicos y organizacionales del sistema en estudio.

Palabras clave
Requerimientos no funcionales, caso de uso, atributos del sistema, calidad del sistema.

1. Introducción

Un equipo de proyecto informático inicia su trabajo con el objetivo de desarrollar un sistema que satisfaga los requerimientos actuales del usuario. Luego de un tiempo de trabajo del equipo, lamentablemente puede suceder que se demore por "cuellos de botella" surgidos en el análisis y diseño del sistema, y el equipo comienza a sentir la presión de la próxima fecha impostergable de terminación del proyecto; además arrastra un importante desvío temporal en su ejecución. El riesgo de no poder terminar el proyecto en tiempo y forma, hace que comiencen a aparecer y crezcan rápidamente problemas de calidad en el producto y proceso, con la posible consecuencia del abandono del proyecto. Este es un escenario común que se mantiene en muchos equipos de proyectos informáticos, aún hasta el día de hoy. Entonces, ya sea que los requerimientos no funcionales no fueron especificados en el tiempo, o estudiados con la atención explícita de los riesgos que involucran, se dificulta el tratamiento de los problemas de calidad que puedan surgir.

Sin objetivos de calidad que guíen a arquitectos e ingenieros, las chances de mejorar el producto son arbitrarias, y es duro medir al sistema durante las revisiones de diseño y arquitectura, y test del sistema. Por ello, se presenta aquí otra definición de calidad como *la totalidad de rasgos y características de un producto o servicio que se refiere a su habilidad para satisfacerlos en forma regular, o necesidades implicadas* [9].

En este trabajo, se realiza una investigación sobre la definición inicial de requerimientos no funcionales, como parte del proceso de educción de requerimientos. Se tomarán las lecciones expuestas en el documento de Bredmeyer Consulting [4]. Si no se considera la educción, documentación y registro de requerimiento no funcionales, se hace más difícil priorizar los problemas de calidad en el sistema en estudio. Recuérdese que la calidad es el grado de

[1] Trabajo realizado en el marco del proyecto CICYT Nº 23/C062, desarrollado en la Facultad de Ciencias Exactas y Tecnologías, Universidad Nacional de Santiago del Estero.

concordancia entre los requerimientos del producto y el producto actual; entonces, se define la calidad desde el punto de vista de la percepción del usuario, expectativas y objetivos o necesidades.

La idea de este trabajo es, en una etapa inicial, poder identificar fuentes de requerimientos no funcionales, clasificarlos con el modelo de casos de uso, y poder deducir los atributos, o cualidades del sistema, que permitan más adelante, evaluar al sistema acorde a dichos requerimientos. Para ello, en la sección dos se presenta una clasificación de requerimientos no funcionales; luego en la sección tres, se identifican estos requerimientos por categoría, según la fuente de información que los produce; mientras que en la sección cuatro se muestran definiciones emitidas por autores varios sobre requerimientos; y finalmente en la sección cinco se presenta el caso de uso como herramienta de análisis del ámbito del requerimiento no funcional. A continuación se muestran las conclusiones del trabajo y la bibliografía utilizada.

2. Clasificación de requerimientos no funcionales

Es necesario referirse a los conceptos de requerimientos de sistemas y realizar distinciones entre ellos, en lo que concierne a requerimientos funcionales y, lo que generalmente en la práctica se conoce como requerimientos no funcionales [4].

Los requerimientos funcionales describen el comportamiento, funciones o servicios del sistema, y realizan los objetivos, tareas o actividades solicitadas por el usuario. Los requerimientos no funcionales incluyen condiciones (restricciones) y cualidades del sistema. Las cualidades son propiedades o características del sistema que sus *stakeholders* (o quienes tienen a su cargo la administración del proyecto informático, como por ejemplo la gestión del proyecto o la supervisión del riesgo del proyecto) toman en cuenta, y que afectará el grado de satisfacción del sistema. Las condiciones no están sujetas a negociación, y contrario a las cualidades, están (en todo caso en forma teórica) fuera del alcance durante el diseño. Las restricciones de contexto son las características del "super sistema", es decir, del gran sistema en el que encajará el sistema en desarrollo, o la organización que condiciona el proyecto en alguna manera. Como ejemplos se incluyen el sistema operativo destino, o la plataforma de hardware en el caso del medio ambiente del usuario, o el conjunto de habilidades de los desarrolladores a cargo del equipo de proyecto.

Otra distinción útil que se realiza, es entre el sistema y los productos que se generan en su desarrollo, es decir, arquitectura, diseño y código. En el caso del sistema que se desarrolla, las cualidades de interés se refieren a los objetivos del usuario, por lo que [4] refiere a ellas como cualidades *run-time*. En el caso de los productos, las cualidades son dirigidas por los objetivos de la organización en que se encuentra el proyecto, y se refiere como cualidades *development-time*; Bennet 1997 [2] denomina a estas últimas, los requerimientos *build-time*.

2.1 Cualidades *run-time*

Informalmente se puede pensar que los requerimientos funcionales capturan *lo que el sistema debe hacer*, y las cualidades *run-time* se describen como "*que tan bien estos requerimientos funcionales son satisfechos"*, donde el "*que tan bien*" es definido por observadores externos ó propiedades medidas del comportamiento del sistema, no sobre su implementación interna. En otras palabras, "*el que tan bien"* debe ser evaluado por el usuario en términos de alguna característica que el mismo valore o se interese. Estas cualidades *run-time* incluyen:

- Usabilidad: significa que la aplicación es fácil de usar, de aprender, de memorizar, de ser eficiente, etc.
- Configuración y soporte.

- Corrección, confiabilidad, disponibilidad.
- Calidad de requerimientos de servicio: como perfomance, tiempo de respuesta, demora de tránsito, latencia, etc.
- Propiedades de seguridad: así llamadas porque previenen la ocurrencia de cosas no gratas para el sistema, como confiabilidad y tolerancia de falla.
- Escalabilidad operativa: que incluye soporte para usuarios adicionales, o sitios, o volúmenes altos de transacción.

El alcance de estos requerimientos *run-time* puede ser para todo el sistema, o ser local, para un comportamiento específico.

2.2 Cualidades *development-time*

En adición al desarrollo de sistemas que satisface a sus usuarios, el equipo de desarrollo tiene un gran interés en las propiedades de los productos del proceso de desarrollo, como arquitectura, diseño, código, etc.. Las cualidades de estos productos influencian el esfuerzo y costo asociado al actual desarrollo, así como también soportan los futuros cambios o usos (mantenimiento, mejora o reuso). Ejemplos de requerimientos de calidad *development-time* son:

- Localización: se define como la habilidad de hacer adaptaciones debido a diferencias regionales.
- Modificabilidad o extensibilidad: es la habilidad para agregar funcionalidad futura, sin especificaciones.
- Evolucionabilidad: se entiende como el soporte para nuevas capacidades, o la habilidad para explotar nuevas tecnologías.
- Composición: es la habilidad para integrar sistemas de componentes *plug-and-play*.
- Reusabilidad: es la habilidad para el uso (reuso) en futuros sistemas.

Esta distinción de cualidades *run-time* y cualidades *development-time* tiene implicancias importantes en la especificación de los requerimientos no funcionales. También, algunas diferencias pueden ser hechas entre las cualidades *run-time* y las cualidades *development-time*. Por ejemplo, la perfomance y la modificabilidad pueden estar en conflicto en el diseño del sistema, ya que el deseo de los usuarios (a menudo influenciados por una presión competitiva) para la perfomance, puede estar amenazada por el objetivo de la organización de desarrollo de tener una arquitectura más sostenible.

En general, las cualidades *run-time* proporcionan valor al usuario y tienen relación con una diferenciación competitiva a corto plazo. Las cualidades *development-time*, para la mayor parte, proporcionan valor de negocio (como opuesto al valor directo del usuario final), y tienen que ver con la competitividad a largo plazo del negocio. Todo aquello que sea medido en sucesos a corto plazo pondrá requerimientos a corto plazo, contra aquello que contribuyen a la efectividad a largo plazo. En particular, desarrolladores y administradores de proyectos están trabajando contra el reloj de entrega (en el mercado) del producto. Esto es porque la arquitectura debe ser vigilada tomando en consideración las cualidades *development-time*.

Andreozzi et. all [1] distinguen los requerimientos no funcionales de las condiciones (restricciones), porque las condiciones principalmente expresan rasgos y características fácilmente de cuantificar. Por ejemplo, es muy difícil definir condiciones y constantes relacionadas con la “usabilidad” de un servicio o producto. Por supuesto, las condiciones pueden ser muy útiles para modelar, por ejemplo el tiempo de entrega de un servicio o producto en un flujo de trabajo. Cada requerimiento no funcional tiene relaciones lógicas e ingenieriles con ciertos atributos de calidad.

En [1] también se realiza una descripción interesante (según quien suscribe), sobre la evaluación de la calidad total basada en los conceptos de medición, modelos y servicios accesibles en la web (por razones de espacio no se presenta en este documento). También se considera que la información de los servicios es relevante para los aspectos del diseño y arquitectura del sistema en estudio, e influyen en la definición de requerimientos no funcionales.

Según [10] existe un debate sobre el rol que los requerimientos no funcionales deben desempeñar en la educción de requisitos y el proceso de desarrollo del sistema. Los ingenieros en sistemas se refieren a dos visiones de trabajo cuando se hacen esta pregunta, estando el límite de ellas en la naturaleza "cualitativa" de los requerimientos no funcionales. Un requerimiento cualitativo es análogo a un requerimiento no medible, lo que en cualquier disciplina se convierte en algo que se quiere evitar a toda costa. En los años recientes, hubo un empuje para hacer que los requerimientos no funcionales sean en su naturaleza más cuantitativos, como lo hicieron autores como Chung 1993, French 1998, Yacobs 1999.

En [7] se presenta una relación gráfica (diagrama) interesante entre los métodos de colección de datos, métricas, mediciones, y juicio humano. Este diagrama muestra como las mediciones de los cuatro niveles operativos identificados en un sistema colaborativo, están "encajados uno dentro del otro", enfatizando el mapeo o direccionamiento (relación) entre estos niveles. Se ilustra métodos de colección de datos, como la herramienta de log o el video tape, para obtener métricas del sistema. Las métricas proporcionan el input necesario para refinar las mediciones de los requerimientos, capacidades, servicios y tecnologías. El juicio humano, tanto de expertos como de usuarios, está asociado con los cuatro niveles.

3 Fuentes de requerimientos no funcionales

3.1 Para los requerimientos *run-time*

Según [4] los requerimientos no funcionales de tipo *run-time* surgen de fuentes de información como el ambiente operativo, el(los) usuario(s), y productos competitivos. Dentro de estas fuentes, se identifican distintas categorías de estudio, como se detallan a continuación:

Las restricciones de sistema

Aquí se observan a elementos del ambiente en los cuales el sistema debe encajar, eso sirve como restricción (o condición) del sistema. Esto puede ser propio de la infraestructura instalada (por ejemplo hardware y plataforma del sistema operativo) o aplicación a entregar en el mercado, o puede estar en la forma de factores organizacionales o el proceso que el sistema soportará.

Los objetivos, valores, e intereses de usuario

Al establecer las cualidades *run-time* para un sistema, es importante identificar todas las categorías de usuarios (incluyendo de otros sistemas) que interactuaran con el sistema, y entender que atributos de calidad se debe tener en cuenta. Un atributo de calidad como la perfomance puede surgir para un usuario como un interés, y para otro como un valor, por eso es útil una educción directa de requisitos para ambos, es decir interés y valor para cualquier (grupo) de usuario(s). Es importante dirigir a crear "justo lo que quieren los usuarios", con las cualidades que ellos toman en cuenta – los rasgos *whiz-bang* de baja prioridad o cualidades solo incrementan la complejidad para la organización de desarrollo y /o el usuario [4]. El equipo de requerimientos debería sin embargo, estar alerta a requerimientos que el usuario presupone o que no estén disponibles para articularse directamente. Entender el objetivo de

los usuarios y forzar a que impacte su suceso y sentido de utilidad, ayudará a surgir y establecer las prioridades de cualidades del sistema, así como la funcionalidad por supuesto.

En adición a descubrir que cualidades son importantes al usuario en el nivel de sistema, las cualidades asociadas a una función particular u objetivo de usuario debe ser educida. Las cualidades pueden necesitar ser trasladadas por los desarrolladores desde los objetivos de nivel usuario, valores e intereses, en requerimientos específicos de calidad técnica. Por ejemplo, un requerimiento de usuario no puede ser degradado o impedido por el rendimiento lento del sistema, al realizar una tarea que puede ser trasladada en requerimientos sobre el tiempo de transacciones y la potencia de red.

El análisis competitivo de rasgos
Las cualidades *run-time* están a menudo asociadas con los *rasgos* del producto. Los *rasgos* son generalmente pensados como las características del producto, que establece su competencia, frecuentemente distinguiendo las funciones del producto (diferenciadores únicos de productos y línea de base) con al menos un atributo de calidad. Por ejemplo, varios servicios de catálogo basados en la web tienen opciones de pago *on-line*. Para aliviar el interés de mercado, el rasgo de pago electrónico incluye seguridad en la transacción como un atributo esencial.

Los requerimientos de rasgos pueden provenir de productos pasados, estar dirigidos por productos competidores o en forma proactiva por el equipo de desarrollo. Productos competidores, o la evaluación amenazante de la prensa hacia ellos, puede provocar las expectativas del usuario, tanto en términos de funcionalidad como en términos de cualidades del sistema. Por esta razón, el mercadeo juega un rol importante en establecer los requerimientos no funcionales conducidos por el análisis competitivo para entender el archivo de competencias sobre cualidades que el cliente valora, o la prensa amenazante enfatiza (por la influencia de la revisión del producto la decisión de adquirir sin tener necesariamente los objetivos prioritarios top del usuario, valores e intereses).
El equipo de desarrollo también juega un rol en influenciar expectativas para estas cualidades, entendiendo que nuevas oportunidades se ofrecen por avances tecnológicos.

Los requerimientos *development-time* típicamente están dirigidos por la organización de desarrollo (aunque en el caso de desarrollo sin recursos, ellos puede surgir del cliente)

Las restricciones de la organización de desarrollo
En el desarrollo del producto, las restricciones se ubican en los niveles superiores de la administración; típicamente toman la forma del tiempo requerido del mercado para la aplicación o entrega y/o recursos fijos de desarrollo. Cuando ambas variables son fijas, los requerimientos de los rasgos deben ser estrictamente alcanzados. Esto muestra que funcionalidad se ha alcanzado para la entrega actual y qué se ha diferido, y el manejo de amenazas entre las cualidades de los atributos del sistema.

Otros factores del desarrollo de la organización, como el curriculum vitae y las habilidades de los ingenieros, puede también restringir en lo que la organización pueda cumplir dada otra restricción como el tiempo de mercadeo.

3.2 Para los requerimientos *development-time*
Por otra parte, en [4] los requerimientos no funcionales de tipo *development-time* pueden identificarse en las siguientes categorías de estudio, como se detallan a continuación:

Los objetivos, valores e intereses de la organización en desarrollo
Los *stakeholders* [4] de la organización en desarrollo incluyen administración estratégica (es decir, el gerente general y el gerente de desarrollo/tecnología), administradores de proyecto y programa, evaluadores (medidores) de calidad, ingenieros de manufacturación y marketing etc. Sus objetivos, valores y aspectos pueden relacionarse con el rendimiento del negocio, planificación, productividad y efectividad, balance de trabajo diario, etc.
Por ejemplo, la planificación estratégica establece que el plan de portfolio (desarrollo) del producto, que incluye las entregas planeadas (se refiere a que productos en que cronograma).

Los arquitectos y los administradores técnicos pueden traducir aquellos objetivos del portfolio en requerimientos de calidad de tipo *development-time* tales como extensibilidad, evolucionabilidad, y reuso, sabiendo que el portfolio no puede ser ejecutado sin estas características. Los desarrolladores pueden estar interesados en que los rasgos de reuso, de hecho, entreguen las cualidades que sus productos particulares requieren.

Los competidores y las amenazas de la industria
El benchmark [4] de los procesos de los competidores (es decir, cuantos productos entregan por año, con cuanta gente) y tendencias de la industria, pueden dirigir a la industria para adoptar objetivos de la producción más agresivos, y que pueden oportunamente traducirse en cualidades *development-time,* tales como la evolucionabilidad y el reuso. Cuando se ha trabajado con los *satakeholders* para obtener los requerimientos, estos necesitan ser documentados de tal modo que los arquitectos, diseñadores, e implementadores puedan entenderlos y crear un sistema que satisfaga esos requerimientos. Al final de cada iteración el sistema emergente debe ser validado contra los requerimientos.

3.3 Requerimientos SMART
En general, los requerimientos no funcionales han sido especificados en términos pobres y difusos, abiertos a un amplio rango de interpretaciones subjetivas. Debido a esto, ellos proporcionan poca guía a los arquitectos e ingenieros, de modo que realizan el “trueque” necesario para cumplir presiones de la planificación y objetivos de funcionalidad. En cambio, los requerimientos no funcionales necesitan ser precisos y accionables. Los denominados requerimientos “SMART” de Mannion y Keepence, 1995 [4] tienen las siguientes características:

- Específico: significa sin ambigüedad, usando terminología consistente, simple y con el apropiado nivel de detalle.
- Medible: es posible verificar que el requerimiento ha sido encontrado. Entonces, ¿qué tests deben ser realizados? ¿O que criterio debe tomarse para verificar que se encuentra el requerimiento?
- Accesible: se refiere a técnicamente factible. Cabe la pregunta ¿cuál es su juicio profesional sobre la “habilidad técnica para hacer” del requerimiento?
- Realizable: es realista, si se dan los recursos. ¿Tiene Ud. el don de mando? ¿Tiene Ud. la habilidad de dirigir? ¿Tiene Ud. el acceso a la infraestructura de desarrollo necesaria? Tiene Ud. acceso a la infraestructura *run-time* necesaria? ¿Tiene Ud. suficiente tiempo?
- Rastreable (ubicable): se lo puede vincular desde su concepción (a través de su especificación), a su diseño siguiente, implementación y test.

El primero de la lista definido como “específico y medible” proporciona un criterio para cada requerimiento de calidad, no está bien especificado si se refiere a “vago o ambiguo” o “no medible”. El siguiente “accesible y realizable” proporciona un chequeo que se debería realizar con cada requerimiento, no es un requerimiento si falla al encontrar cualquiera de estos

criterios. Nótese que cuando un requerimiento no está especificado ni medido, es difícil saber si será accesible y realizable.

4 Enfoques para el estudio de requerimientos no funcionales

Muchos de los primeros trabajos sobre requerimientos no funcionales se dirigieron a medir al sistema software según el conjunto de requerimientos no funcionales que debía satisfacer; se usó para ello algún tipo de análisis cuantitativo [12], ofreciendo métricas predefinidas para medir el grado de encuentro de un objeto software con un requerimiento no funcional particular.

Recientemente, algunos trabajos propusieron usar enfoques que explícitamente traten con requerimientos no funcionales antes que las métricas sean aplicables [6] [2] [8]. Estos trabajos proponen el uso de técnicas para justificar decisiones sobre la inclusión o exclusión de requerimientos que impactarán en el diseño del software. Contrariamente al enfoque de las métricas, luego estos enfoques están interesados en hacer que los requerimientos no funcionales sean una parte importante e interesante del proceso de desarrollo de software.

Boehm e In [2] proponen una base de conocimiento donde los requerimientos no funcionales son priorizados a través de las perspectivas de los *stakeholders*, tratando con los requerimientos no funcionales a un alto nivel de abstracción. Kirner [6] describe propiedades para seis requerimientos no funcionales del dominio de sistemas de tiempo real que son: perfomance, confiabilidad, seguridad, mantenibilidad y usabilidad. Este trabajo proporciona heurísticas en como aplicar las propiedades identificadas para encontrar requerimientos no funcionales y luego medirlos. De todos modos, goza de ser un enfoque amplio que puede ser aplicado a otros requerimientos no funcionales, en el dominio mencionado o en otros.

Se introdujo un avance significativo cuando los requerimientos no funcionales fueron tratados como objetivos competitivos extensivamente refinados y amenazantes entre cada uno como un intento para arribar a la solución aceptable. El *NFR Framework* [6] es uno de los pocos trabajos que tratan con requerimientos no funcionales, iniciando desde etapas tempranas del desarrollo de software a través de una amplia perspectiva. El *NFR Framework* ve a los objetivos no funcionales como objetivos que pueden generar conflictos entre ellos y deben ser representados como objetivos del soft a ser satisfechos. El concepto de *softgoal* (objetivo del software) fue introducido para alcanzar con la naturaleza abstracta e informal del requerimiento no funcional. Cada *softgoal* será descompuesto en objetivos menores representado por un estructura tipo grafo inspirada en “arboles y/o”, usados en la solución de problemas. Este proceso continúa hasta que el ingeniero de requerimientos considere el *softgoal* como satisfecho[2] (en el sentido de operacionable). Una de las ventajas del enfoque orientado al objetivo es que puede ser usado para modelar y razonar tanto sobre los requerimientos funcionales como no funcionales.

Inicialmente vagos, los requerimientos no funcionales son eventualmente operacionables en términos de técnicas que pueden ser implementadas. Las operaciones pueden ser vistas como requerimientos funcionales que surgen del reconocimiento de requerimientos no funcionales.

De todos modos, como es importante la obtención de un conjunto de requerimientos bien formado y tan completo como sea posible, necesitamos entender y sistematizar como los

[2] El término satisfacer fue usado por Herbert Simon para expresar una “alternativa suficientemente buena”.

requerimientos conducen el resto del desarrollo del sistema, especialmente durante la fase de diseño. Ninguno de los trabajos mencionados antes salvaron este problema.

5 El Caso de Uso aplicado al análisis de requerimientos no funcionales

Los casos de uso han sido ampliamente usados para especificar requerimientos funcionales. Por ejemplo, ampliando el caso de uso con un registro para los requerimientos no funcionales asociados a ese caso de uso, las cualidades *run-time* asociadas a una función particular pueden ser capturadas convenientemente. En este trabajo, para poder definir mediciones de calidad de requerimientos no funcionales, se realiza un análisis inicial del ámbito de un requerimiento no funcional, a partir del esquema de Coleman, 1998 [5], que se traduce en la Tabla 1.

Tabla 1: Modelo de caso de uso

Caso de Uso	# Identificar el caso de uso a partir de un número de referencia.
Descripción	Se refiere al objetivo a cumplirse a través del caso de uso y las fuentes del requerimiento.
Actores	Se define una lista de actores que participan en el caso de uso.
Suposiciones	Son las condiciones que deben ser verdaderas para que el caso de uso se ejecute con éxito
Pasos	Son las acciones interactivas necesarias entre los actores y el sistema para que se alcance el objetivo
Variaciones (optativo)	Indica cualquier variación de los pasos en el caso de uso.
No funcionales	Define la lista de requerimientos no funcionales que el caso de uso debe reconocer. Los requerimientos no funcionales se listan en la forma: <palabra-clave>:<requerimiento>, donde palabra-clave incluye (pero no está limitado) a Perfomance, Confiabilidad, Tolerancia de falla, Frecuencia, Prioridad. Cada requerimiento se expresa en lenguaje natural o con un formalismo apropiado.
Aspectos	Se refiere a situaciones puntuales pendientes de resolver.

Ya que los requerimientos *development-time* como extensibilidad o reuso generalmente se relacionan con funcionalidad futura de tipo *run-time*, ellos pueden ser teóricamente "capturados" también usando los casos de uso. De todos modos, la exhaustiva exploración de requerimientos sobre futuras entregas o variantes de productos, está fuera de la cuestión (el primer producto demoraría indefinidamente), por ello los requerimientos *development-time* necesitan ser tratados en forma diferente.

Debido al desconocimiento e impredecibilidad de los requerimientos *development-time* y necesidades de usuarios, lo "que pasa si en este escenario", es usado para explorar la robustez del sistema a futuros requerimientos y la facilidad de adaptarlo a tecnologías alteradas y objetivos de usuarios. Estos son descriptos menos formalmente que los casos de uso, reconociendo que ellos son solamente representativos de categorías de cambios que pueden tener para realizar sobre la vida del producto o familia.

Precisamente el caso de uso captura el *quien* (actor) que *hace* (interacción) con el sistema, para que *propuesta* (objetivo), sin dejar de relacionarse con el sistema interno. Se aconseja priorizar los requerimientos no funcionales y no caer en la tarea de expresar cada cualidad posible como un requerimiento del sistema. Se recomienda recoger los objetivos de usuario según Stake-Holder Profiles, Malan and Bredemeyer, 2000 [6] [5], extraer los requerimientos de calidad, y priorizarlos. Es útil por lo menos categorizar los requerimientos no funcionales en: *"se debe hacer","se necesita hacer", y "se puede hacer".*

Se explica en la tabla 2 un caso de uso concreto como ejemplo de análisis de un requerimiento no funcional.

Tabla 2: Ejemplo de caso de uso aplicado a RNF

Caso de Uso	**Nº 82.** *Reparación de la red de celular.*
Descripción	El **operador** rectifica un reporte cambiando los parámetros de la célula. (Fuente: Manual de Operaciones).
Actores	**Operador** (primario); **Red de celulares**; Ingeniero de Mantenimiento.
Suposiciones	Los cambios en la red siempre son satisfactorios cuando son solicitados por el estado de la red.
Pasos	1. El operador se entera de un problema de la red. 2. El operador inicia la sesión de reparación. **3. REPITA** 3.1 El operador debe ejecutar la aplicación de diagnóstico de la red. 3.2 El operador identifica las células a ser cambiadas y sus nuevos parámetros. 3.3. En operación tipo paralelo: 3.3.1. El ingeniero de mantenimiento testea las células de la red. 3.3.2. El ingeniero de mantenimiento envía el reporte de fallas. **HASTA** que no se reporte problemas. 4. El operador cierra la sesión de reparación.
Variaciones (optativo)	#1 El sistema puede detectar falla y notificar al operador, **O**; #2 El ingeniero de mantenimiento puede avisar la falla al operador.
No funcionales	El *rendimiento (perfomance)* significa que el tiempo para reparar la falla en la red debe ser menor a tres horas. La *tolerancia de falla* significa que una sesión de reparación debe poder tolerar falla de la consola del Operador.
Aspectos	¿Cuáles son los modos de comunicación entre el ingeniero de mantenimiento y el operador?

Finalmente, los objetivos funcionales son el punto de partida para los casos de uso, es decir la especificación de requerimiento funcionales. Mientras que los objetivos de calidad son el punto de partida para la especificación de requerimientos de calidad, es decir de requerimientos no funcionales, según [3].

Un trabajo que define un repositorio de métricas para medición de sitios web, como un caso de uso profesionalmente ampliado con información de medición de calidad en la web, es el de Olsina et. all [8]. Este repositorio permite a evaluadores y *stakeholders*, consultar y disponer de un mecanismo de disponibilidad y reuso de la métrica. Se lee en la conclusión del trabajo que *"se proyecta en un futuro incluir los aspectos de diseño de arquitectura en el ambiente del catálogo, como también su utilidad para el proceso de ingeniería de la empresa basado en la web"*. Se toma referencia de dicho trabajo, ya que en este documento se mencionan aspectos de la arquitectura de sistemas en la definición de requerimientos no funcionales.

Conclusión

Un sistema informático tiene propiedades que "emergen" de la combinación de sus partes. Estas propiedades serán "materia de accidente" seguramente y no de diseño, si los requerimientos no funcionales o cualidades del sistema no se especificaron previamente. Se presenta aquí un modelo extendido de caso de uso como una técnica de análisis inicial, para resumir la funcionalidad del sistema y facilitar la comunicación con los *stakeholder*s, incluyendo clientes y desarrolladores.

Más aún, si se entiende la arquitectura del sistema del sistema para proporcionar una fuente de diferenciación competitiva, los arquitectos del diseño deben tener en cuenta los objetivos a

largo plazo del negocio, y los requerimientos no funcionales asociados con la arquitectura y el desarrollo de los rasgos del sistema. Entonces, se propone complementar en otra instancia de trabajo, este estudio inicial de requerimientos no funcionales con la aplicación del concepto de *softgoal*, ya que los objetivos pueden ser usados como un mecanismo importante para conectar los requerimientos con el diseño del sistema.

Finalmente, la técnica ampliada de caso de uso más la organización y conceptos aquí presentados, constituyen el paso inicial de un esquema de trabajo para el análisis, descripción, negociación y evaluación de calidad de requerimientos no funcionales, además de poder relacionar las decisiones del diseño con los objetivos técnicos y organizacionales de la unidad de negocio involucrada.

Referencias

[1] ANDREOZZI S., MONTESI D., Y MORETTI R. *"Web Services Quality"*. Many Universities, Italy. 2000.
[2] BENNET, D. *"Designing Hard Software: The Essencial Tasks"*. Prentice-Hall 1997.
[3] BREDEMEYER CONSULTING. *"Architecture Requirements Action Guide #2: Stakeholder Profile*", 1999.
[4] BREDEMER CONSULTING. *"Architecture Resources For Enterprise Advantage"*. 2000. Disponible en http:/www.bredemeyer.com
[5] COLEMAN, D.*"Use Case Template"*.1998. Disponible en: http:/www.bredemeyer.com/papwers.htm
[6] MALAN, R., BREDEMEYER, D. *"Stakeholder Profile Action Guide"*, 2000. Disponible en http:/www.bredemeyer.com.papers.htm
[7] NATIONAL INSTITUTE OF STANDARD AND TECHNOLOGY. Visualization and Usability Group (VUG). 2000. Disponible en http:/zing.ncsl.nist.gov/nist-icv/documents/section3.htm
[8] OLSINA L., LAFUENTE G. y PASTOR O. *"Towards a Reusable Repository for Web Metrics"*.GIDIS, Department of Informatics, Engineering School at National University of La Pampa, 1999. Argentina.
[9] QUALITY MANAGMENT AND QUALITY ASSURANCE. Vocabulary ANSI/ASQ Standard. 1998.
[10] RYAN, A. *"An approach to quantitative non-functional requierements in software development"*. School of Information Technology and Engineering. George Mason University. 2001.
[11] YU E., CYSNEIROS L. M. *" Non-functional requirements elicitation"*. Department of Mathematics and Statistic, York University. Fac. of Information Studies, Univ. of Toronto. 2002.
[12] YU E., CHUNG L. *"Using non-functional requirements to systematically support change"*. Department of Computer Science, Univ. Toronto. Computer Science Program, Univ. of Texas, Dallas. 2002.

Datos de contacto

Ing. Leda Beatriz Digión de Grimaldi. Departamento de Informática. Fac. de Ciencias Exactas y Tecnologías, Universidad Nacional de Santiago. del Estero. Belgrano 1900. Santiago del Estero (CP. 4200).
E-mail: ldigion @unse.edu.ar