SQuaRE* y la Orientación a Aspectos**

Isi Castillo[1,2,3], Francisca Losavio[2], Alfredo Matteo[2]

[1] Laboratorio de Informática, Universidad Nacional Experimental Sur del Lago "Jesús María Semprum"
[2] Centro ISYS, Escuela de Computación, Universidad Central de Venezuela
{castilloi, flosav, almatteo}@cantv.net

Resumen. El desarrollo de software orientado a aspectos (AOSD*), representa un nuevo paradigma de ingeniería de software basado en los conceptos de la Programación Orientada a Aspectos. La investigación se centra en el tratamiento temprano de las incumbencias (concerns) transversales en combinación con los procesos clásicos de ingeniería de requisitos, donde los requisitos de calidad son relevantes. A pesar del interés en esta línea de investigación, no es evidente encontrar una visión compartida y homogénea sobre los términos utilizados. El objetivo de este trabajo es integrar los principales conceptos del AOSD con las nociones de ingeniería de requisitos y la reciente terminología sobre requisitos de calidad del nuevo estándar ISO/IEC 25030 (SQuaRE); para establecer un mejor entendimiento y consenso hacia un vocabulario común para la emergente disciplina de la Ingeniería de Requisitos de Calidad Orientada a Aspectos. El resultado principal de esta integración es un modelo conceptual, expresado en UML.

Palabras Claves: aspectos, incumbencias, incumbencias transversales, ingeniería de requisitos, calidad de software, ISO/IEC 25030, SQuaRE

Abstract. Aspect Oriented Software Development (AOSD), based on Aspect Oriented Programming, is part of the post-object paradigm of software engineering. The early treatment of non functional requirements elicitation and specification, combining quality properties and crosscutting concerns at classical requirements engineering, are open research subjects. In spite of the increasing interest, a homogenous vision of the terminology involved is still missing. The goal of this work is to integrate the main concepts used in AOSD, with requirements engineering notions and the recent ISO/IEC 25030 software quality requirements terminology, setting the basis for a better understanding and consensus towards a common vocabulary for the emerging Aspect Oriented Quality Requirements Engineering discipline. The main result of this integration is an UML conceptual model.

Keywords: aspects, concerns, crosscutting concerns, requirements engineering, software quality, ISO/IEC 25030, ISO/IEC 25010, SQuaRE.

---

* SQuaRE: Software Quality Requirements and Evaluation, AOSD: Aspect-Oriented Software Development

** Este trabajo es parcialmente financiado por el Consejo de Desarrollo Científico y Humanístico de la Universidad Central de Venezuela, proyectos MODABAC Project No. 03-005821-2008 y GEMOCLASS No. 03-006051-2005

3 Autor de correspondencia

1 Introducción

Una de las mejores prácticas de la ingeniería del software es seguir un proceso de desarrollo maduro. Así, el proceso de desarrollo ha evolucionado buscando mejoras para garantizar la calidad del producto de software resultante. Sin embargo, como describir y garantizar un producto de software de calidad, de acuerdo a la madurez del proceso es una actividad aún poco clara. Los métodos de desarrollo de software existentes son dirigidos más por la especificación de la funcionalidad del sistema que por la consideración de sus aspectos no funcionales. En general, estos aspectos no funcionales son considerados en etapas posteriores del desarrollo, contribuyendo con el enmarañamiento y la dispersión del código final, contradiciendo la propiedad de flexibilidad a cambios del desarrollo orientado a objetos. Por otra parte, la importancia de tener un entendimiento claro y una correcta especificación de requisitos se ve influenciada por la complejidad de los actuales sistemas de software, en particular si estos tienen que responder a incumbencias o *concerns* tales como interoperabilidad, adaptabilidad, disponibilidad y seguridad, propiedades que van más allá de la funcionalidad principal del sistema o del servicio ofrecido por el componente de software. Estas incumbencias que en general no son exigencias directas del usuario, pueden afectar a múltiples componentes [3], de aquí la importancia de que deban ser consideradas temprano, al mismo tiempo que la funcionalidad principal. En la actualidad, la disciplina de ingeniería de requisitos está tratando de establecer un razonamiento preciso para considerar estos tipos de requisitos que afectan múltiples componentes. La ingeniería de requisitos orientada a aspectos (AORE[4]) y el diseño arquitectural orientado a aspectos (AOAD[5]) [6][7], estudian la identificación y el manejo temprano de las incumbencias transversales ("*crosscutting concerns*") para mejorar la calidad del código [2][3][6][7][16]. Este enfoque se conoce como aspectos tempranos ("*early aspects")*.

Una incumbencia o "*concern"* es una propiedad o punto de interés de un sistema [7]. En [8] la IEEE define el término como aquellos intereses relacionados con el desarrollo del sistema y su operación, o cualquier aspecto crítico e importante para los stakeholders o participantes en el proyecto de software. En ISO/IEC 9126-1 [11], un concern es un requisito del sistema, es decir una consideración que debe ser tomada en cuenta para satisfacer una meta del sistema. Kiczales en [12], en el enfoque denominado Programación Orientada a Aspectos (conocido como AOP[6]), define a los concerns transversales como los concerns que son encontrados dispersos en múltiples módulos o enmarañados en un único módulo, por ejemplo persistencia de datos, control de acceso. Un aspecto o "*aspect*" en AOP, es definido como una unidad modular diseñada para implementar un concern; puede contener código con instrucciones de dónde, cómo y cuándo invocarlo [7]. En la ontología presentada en [18], un aspecto es definido como una unidad que modulariza un concern transversal. Kiczales define un aspecto como la estructura que encapsula un concern transversal [12]. En nuestro trabajo, un aspecto es considerado como un concern transversal de acuerdo a la definición presentada en la ontología de orientación a aspectos [18]. La

4 AORE: Aspect-Oriented Requirements Engineering

5 AOAD: Aspect-Oriented Architectural Design

6 AOP: Aspect-Oriented Programming

temprana separación de los concerns transversales y su composición en aspectos son las principales tendencias de investigación en la disciplina *Early Aspects*.

Por otra parte, en la literatura los conceptos *aspectos* y *concern* son frecuentemente considerados sinónimos. En la AORE, los términos *requisitos* y *concern* son utilizados indistintamente y las incumbencias transversales o *concerns transversales* son asociados indistintamente a un alto nivel con las propiedades de calidad o requisitos de calidad del producto de software tales como disponibilidad, confiabilidad y en un bajo nivel con los atributos de calidad (o atributos medibles) como tiempo de respuesta, en el sentido de ISO/IEC 9126-1 [10]. En [16] un requisito es considerado un tipo especial de concern. Nótese que los principales conceptos y definiciones utilizadas en el dominio del AOSD son extensiones de la terminología de la AOP [12][13].En la actualidad, son pocos los trabajos presentados donde se asocie la orientación a aspectos, el estándar ISO/IEC 9126-1 para especificar requisitos de calidad y la ingeniería de requisitos. En [15] Navarro et al. presenta una propuesta para organizar los requisitos integrando técnicas orientadas a aspectos dentro de un enfoque orientado a metas. Van den Berg et al. en [18] presenta una primera versión de una ontología sobre la orientación a aspectos, en particular en esta ontología se incluye un glosario para términos comunes del AOSD y una propuesta para un framework conceptual.

El objetivo principal de este trabajo es establecer un modelo conceptual integrado para clarificar y facilitar el uso de la emergente terminología del AOSD, el cual integrará: - ISO/IEC 25030, nuevo estándar de la familia SQuaRE [9][10], que se refiere a una clasificación de requisitos, - ISO/IEC 25010, también de SQuaRE, que contempla el modelo de calidad (aún denominado ISO/IEC 9126-1 mientras no se apruebe el nuevo estándar), - un modelo de clasificación de requisitos (RECLAMO[7]) [4]. Este nuevo modelo integrado puede ser usado en el contexto de una emergente disciplina de Ingeniería de Requisitos de Calidad Orientada a Aspectos (AOQuaRE[8]). Además de la introducción, este trabajo está estructurado de la siguiente manera: la sección 2 introduce la familia de estándares SQuaRE: ISO/IEC 25010 para especificar el modelo de calidad del producto de software y ISO/IEC 25030 para la identificación de los requisitos de software. La sección 3 presenta el modelo conceptual integrado propuesto. La sección 4 presenta brevemente un caso de estudio y para finalizar las conclusiones y trabajo futuro.

2 SQuaRE

De acuerdo al estándar sobre requisitos de calidad SQuaRE [9], el sistema de software es usualmente parte de un sistema más grande y complejo, donde los requisitos de software y requisitos de calidad están estrechamente relacionados y los requisitos de software no pueden ser considerados de manera aislada. En consecuencia, es importante considerar los requisitos de calidad de software en las primeras etapas del ciclo de vida, como parte importante de la especificación general requisitos. Este estándar se enfoca en los requisitos de calidad bajo una perspectiva del sistema. En el nuevo estándar, los requisitos son categorizados en ISO/IEC 25030 y especificados mediante un modelo de calidad según el estándar ISO/IEC 25010 (formalmente es el mismo estándar para modelo de calidad ISO/IEC 9126-1). El modelo de calidad está

[7] RECLAMO: Requirements Classification Model

[8] AOQuaRE: Aspect-Oriented Quality Requirements Engineering

organizado jerárquicamente en características y subcaracterísticas hasta los atributos, los cuales son los elementos medibles. Los atributos especifican los requisitos de calidad del software en términos de medidas y valores objetivos. El estándar proporciona una serie de recomendaciones y una guía para especificar estos requisitos, para así garantizar que estén en conformidad con las necesidades de los participantes en el proyecto de software o stakeholders. Formalmente, SQuaRE está conformado por una familia de estándares, en varias divisiones, bajo el titulo general *Software Product Quality Requirements and Evaluation* [9][10]. En particular esta investigación se centra en los estándares ISO/IEC 25010 (modelo de calidad) y ISO/IEC 25030 (requisitos de calidad), presentados a continuación.

2.1 ISO/IEC 25010

Los productos de software son construidos conforme con necesidades especificas, requeridas por sus usuarios. Su calidad es determinada en la medida que estas necesidades son alcanzadas. La especificación de la calidad del software debe ser detallada y precisa. El modelo de calidad es una manera de formalizar esta especificación. ISO/IEC 25010 - *Quality Model* [10] será una actualización del actual estándar ISO/IEC 9126-1[11], con el mismo propósito de definir un modelo de calidad y de proveer una guía práctica sobre el uso del modelo de calidad. De acuerdo a ISO/IEC 25010, un modelo de calidad está definido como un conjunto de características de calidad, subcaracterísticas y sus relaciones; proporciona un framework para especificar los requisitos de calidad y evaluar la calidad de un producto de software, ofreciendo un entendimiento común y la terminología de calidad de software. Las características son refinadas en subcaracterísticas, en una jerarquía multinivel y éstas en atributos los cuales son los elementos medibles a los que se les asignan métricas [10]. ISO/IEC 25010 [10] es una revisión de ISO/IEC 9126-1 [11], con cambios menores. Mantiene las mismas definiciones y estructura de [11], sin embargo, a las 6 características (funcionalidad, eficiencia, confiabilidad, mantenibilidad, usabilidad y portabilidad de [11] agrega dos más, seguridad e interoperablidad para un total de 8 características; las características agregadas eran subcaracterísticas de funcionalidad en [11]. Efectivamente, para tratar con las aplicaciones actuales de software, por ejemplos orientadas a servicios Web, resulta más natural considerar estas propiedades como características de primer nivel. La calidad del producto está definida en general por tres modelos de calidad: modelo de calidad interna, externa y de calidad en uso. En lo que sigue, mantendremos en general la terminología de los estándares de calidad en inglés, para compatibilidad con la literatura existente.

2.2 ISO/IEC 25030

ISO/IEC 25030 – *Requisitos de Calidad* [9] proporciona una guía para identificar los requisitos de calidad del software, para validar la completitud de la especificación de los requisitos y para identificar criterios de aceptación y aseguramiento de calidad del producto de software. Enfoca principalmente los requisitos de software; la categorización completa de los requisitos en SQuaRE se muestra en la fig.1.

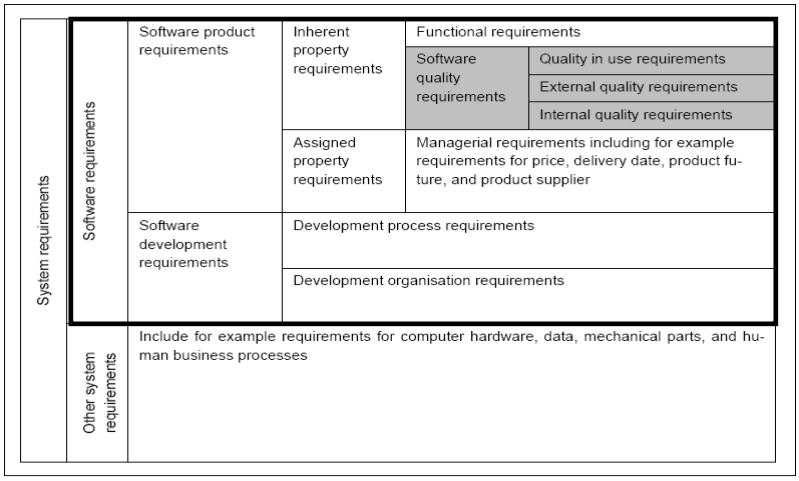

Fig.1. Categorización de los requisitos en ISO/IEC 25030 [9].

De acuerdo a la categorización propuesta en la fig. 1 [9], los requisitos de software dirigen el proceso de desarrollo de software (producto de software). Los requisitos del producto de software incluyen requisitos funcionales y de calidad (*inherent property requirements*) y requisitos gerenciales o *managerial requirements* (*assigned property requirements*). Los requisitos funcionales incluyen requisitos específicos del dominio de aplicación así como también los requisitos funcionales de los usuarios, estos requieren el alcance de metas de calidad las cuales son también un tipo de requisito de calidad. Los requisitos de calidad pueden también implicar requisitos estructurales y arquitecturales. Los requisitos de propiedades asignadas o *assigned property requirements* representan los requisitos que pueden cambiar sin que ocurran cambios en el software, incluye por ejemplo requisitos por el precio, fecha de entrega, suplidor del producto. En ISO/IEC 25030 los *requisitos funcionales* determinan lo que el software es capaz de hacer y los *requisitos de calidad* determinan como se ejecuta el software. En otras palabras, los requisitos de calidad muestran el grado con el cual el software es capaz de proveer y mantener sus servicios y tienen tres vistas diferentes: *requisitos de calidad en uso*, *requisitos de calidad interna* y *requisitos de calidad externa*.

3 Modelo integrado para la ingeniería de requisitos de calidad orientada a aspectos

El modelo conceptual integrado propuesto facilita el razonamiento sobre las principales nociones inherentes a una disciplina de requisitos de calidad orientada a aspectos. En nuestro modelo los elementos requisitos de software, concern y características de calidad (*software requirement*, *concern* y q*uality characteristic*) son las principales nociones utilizadas para interrelacionar la terminología más resaltante de tres ámbitos: REQUISITOS DE SOFTWARE, ORIENTACION A ASPECTOS y CALIDAD DE SOFTWARE, en donde se integra ISO/IEC con RECLAMO para la categorización de los requisitos [4][9][10][17] (ver fig. 2).

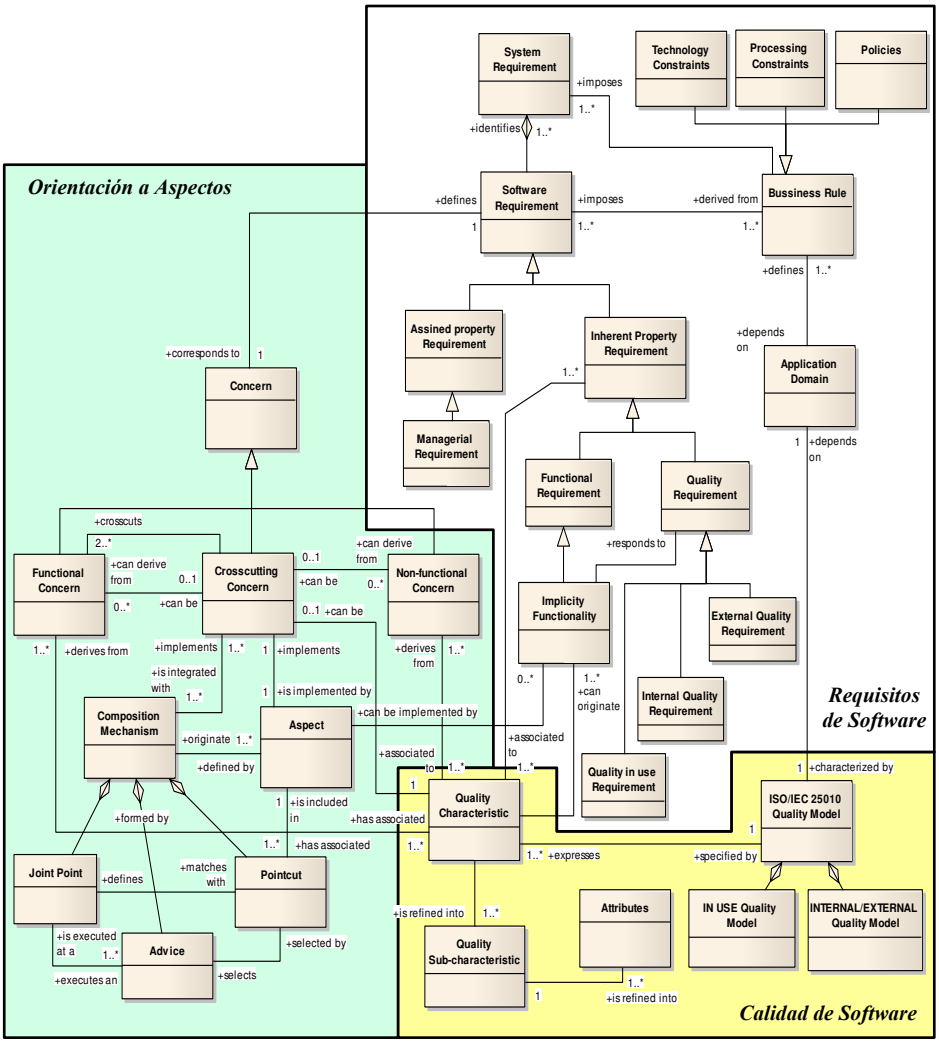

Fig. 2. El modelo integrado. Nótese que en la figura se han dejado en inglés los términos de cada ámbito para facilitar su reutilización.

De manera general, se consideran los requisitos como descripciones del comportamiento principal del sistema o de sus propiedades o atributos y/o de las restricciones sobre el proceso de desarrollo y/o del entorno de ejecución [17][19]. Los *requisitos de software* son especificados y derivados de los requisitos del sistema, en el sentido de ISO/IEC 25030. Sin embargo, consideramos que las reglas de negocio o "*business rules"* impuestas por la organización que requiere el producto de software, pueden imponer algunos de los requisitos del sistema. Las reglas de negocio pueden venir también de una organización externa por ejemplo una institución gubernamental y pueden estar relacionadas con componentes existentes, a menudo restringen quien pueda ejecutar determinados casos de usos o estipular que el sistema de software debe contener cierta

funcionalidad para cumplir con las políticas de la organización. Por otro lado, las reglas de negocio son el origen de metas de calidad específicas que son implementadas como un mecanismo o componente (*funcionalidad implícita*), en adición a la funcionalidad principal. Así, el modelo de metas de Lamswerdee [14], el framework de Requisitos No Funcionales (NFR[9]) de Chung y Yu [5] o los recientes enfoques de Brito y Moreira [2][3], pueden ser usados para especificar tempranamente esta funcionalidad implícita o "*implicit functionality"*. Una funcionalidad implícita puede ser considerada como un requisito funcional no expresado directamente por el usuario. Según el acuerdo general [4][17][19], los requisitos de software son clasificados en requisitos funcionales y no funcionales, en ISO/IEC 25030 los requisitos no funcionales corresponden directamente a los requisitos de calidad o *Quality Requirements* (ver fig.1). Así, en el modelo se tiene la relación de herencia para expresar que los requisitos funcionales y de calidad son una especialización de los requisitos de propiedades inherentes o *inherent property requirements* (ver figura 2). Los requisitos funcionales capturan el comportamiento del sistema y pueden ser expresados como servicios o componentes que deben cumplir metas de calidad específicas. Los requisitos de calidad especifican las condiciones que el sistema debe satisfacer para restringir, condicionar o controlar la ejecución de los componentes del sistema. De acuerdo a ISO/IEC 25030, un requisito de calidad está asociado a una característica de calidad o "*quality characteristic"* y es definido por un modelo de calidad (ISO/IEC 25010), el cual es usado para caracterizar el dominio de aplicación. Adicionalmente los requisitos de calidad según ISO/IEC 25030 están categorizados en requisitos de calidad en uso, requisitos de calidad interna y externa (expresados por las relaciones de herencia en el modelo).

Con respecto al ámbito ORIENTACIÓN A ASPECTOS en el modelo, un *concern* define un requisito de software y este es especializado en concern funcional, no funcional y concern transversal o *crosscutting concern*. Obsérvese que los concern funcionales y no funcionales pueden ser omitidos en el modelo por la relación existente con los requisitos de propiedades inherentes. Sin embargo se han dejado para mostrar que un concern puede entrecruzar tanto a un concern funcional y a un no funcional. Por otra parte, un concern transversal es implementado por un aspecto mediante un mecanismo de composición o "*composition mechanism"*, un aspecto también puede implementar una funcionalidad implícita. Un mecanismo de composición define un aspecto y tiene asociado un punto de corte "*pointcut"*, el cual es una expresión que identifica un punto específico o conjunto de puntos en la ejecución de un sistema. Los puntos de unión o "*joint points"* que ejecutan los avisos o "*advice"*, son también definidos por el mecanismo de composición y representan los puntos de interés en algunos artefactos del ciclo de vida en el cual dos o más concern pueden ser compuestos. Los avisos proveen las acciones que ocurrirán en los puntos de unión y complementan o restringen otros concern en dichos puntos. Los puntos de corte, puntos de unión y los avisos son elementos esenciales para implementar un aspecto. Una característica de calidad derivada de un requisito de propiedad inherente o de una funcionalidad implícita puede ser un potencial concern transversal y por tanto un aspecto candidato. Este punto de muy importante para el tratamiento de los aspectos en las etapas tempranas del desarrollo de software, donde la noción de *aspecto candidato* es introducida [3]. El estándar ISO/IEC 25030 está relacionado solamente con el ámbito de los REQUISITOS DE SOFTWARE del sistema en general y no trata con el ámbito de ORIENTACIÓN A ASPECTOS; sin embargo, en el modelo las relaciones entre estos ámbitos se han establecido. Adicionalmente, el ámbito de la CALIDAD DE SOFTWARE del estándar ISO/IEC 25010 ha sido también modelado y relacionado con los otros ámbitos. Finalmente, para concluir sobre la terminología discutida sobre tres importantes

[9] NFR: Non-Functional Requirements

áreas de investigación de la ingeniería del software, puede indicarse que un concern corresponde a un requisito de software [16] que debe ser manipulado para resolver algún problema de software [13], puede ser usado en la disciplina de ingeniería de requisitos y en el modelado de la arquitectura. Usando el modelo se establece una correspondencia entre los estándares ISO/IEC SQuaRE y el emergente paradigma del AOSD. En la actualidad, los concerns no funcionales y los requisitos de calidad están relacionados con una o más características de calidad del modelo de calidad (potenciales concerns transversales) lo cual representa una de las principales metas del AOSD [1][7].

4 Caso de estudio

A continuación se ilustran los elementos más importantes del modelo propuesto, con el caso de estudio sobre una aplicación web para una tienda de juguetes en línea. La fig.3 presenta una versión simplificada del modelo de caso de uso para la aplicación, donde se identifican los principales requisitos funcionales (casos de usos) y requisitos no funcionales (casos de usos incluidos como aspectos candidatos). Este caso de estudio está basado en una versión simplificada de una tienda de juguetes en línea: un cliente (usuario web) pueda comprar un juguete en línea registrando una cuenta y especificando un login. El sistema permite al cliente navegar a través de un catalogo en línea, el cliente puede agregar artículos a su carro de compra o removerlos sin dificultad; cuando el cliente desee cerrar la compra o "*checkout*" su meta es comprar los artículos que se encuentren cargados en su carro de compras usando su tarjeta de crédito. Adicionalmente, un cliente puede verificar el estado de su orden y cancelar una orden solo si la misma no ha sido procesada. En general, un cliente espera del sistema operaciones seguras, procesamiento rápido, navegación fácil a través del sistema, el cual debe operar los 365 días del año.

4.2 Requisitos funcionales y requisitos de calidad

Para este tipo de aplicación los concerns no funcionales (requisitos de calidad) son identificados y relacionados con los requisitos funcionales, posteriormente le son asociadas características de calidad para identificar los concerns transversales más representativos. La tabla 1 presenta algunos de los elementos del modelo propuesto (considerados de gran importancia) y sintetiza la composición de los principales requisitos funcionales y de calidad del sistema, en la misma se identifican los concerns transversales más importantes y se denotan como aspectos candidatos. Estos concerns transversales son objetivos de calidad identificados de los requisitos funcionales y pueden ser implementadas como un aspecto de allí el porqué denotarlos como aspectos candidatos. La fig.3 ilustra los concerns transversales identificados, los mismos son representados como un estereotipo denominado caso de usos "*aspecto candidato*". Nótese que precisión aparece en un solo requisito funcional, por lo tanto no es transversal y que los requisitos interoperabilidad y disponibilidad dependen del entorno de red y de la disponibilidad de los servidores respectivamente, en una arquitectura de estilo SOA[10]. La usabilidad es responsabilidad de la componente de Interfaz usuario.

[10] SOA: Service Oriented Architecture

Tabla 1. Composición: Requisitos funcionales, requisitos de calidad y concerns identificados

Requisito funcional	Objetivos de calidad	Concern transversal (aspecto candidato)
Login	-Acceso seguro y validación de usuarios. -Interfaz amigable.	seguridad usabilidad
Crear cuenta	-Comunicación con otros sistemas. -Interfaz amigable -Acceso seguro y validación de usuarios.	interoperabilidad usabilidad seguridad
Buscar juguetes	-Comunicación entre catálogos. -Respuestas rápidas. -Interfaz amigable.	interoperabilidad, disponibilidad tiempo de respuesta usabilidad
Cerrar compra	-Comunicación rápida con otros sistemas. -Operación de pago confiable y segura -Interfaz amigable.	tiempo de respuesta, interoperabilidad seguridad, precisión usabilidad

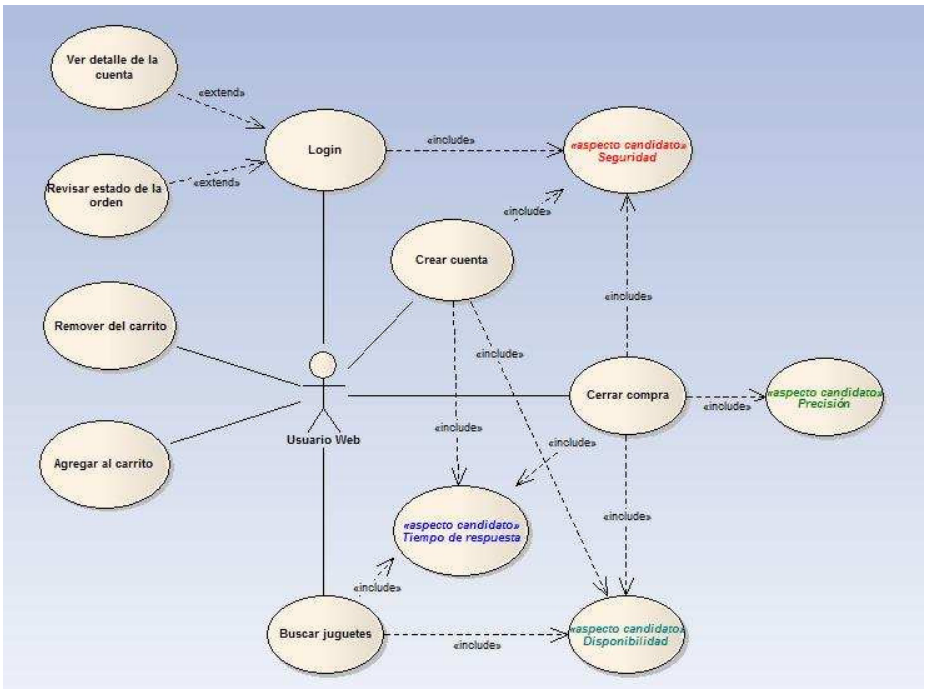

Fig. 3. Diagrama de Casos de Uso para la Vista Lógica de la tienda de juguetes en línea.

4 Conclusiones y trabajo futuro

Este trabajo presenta un modelo conceptual en UML integrado para el razonamiento, entendimiento y manejo de las principales nociones relacionadas con el paradigma de desarrollo de software orientado a aspectos. EL modelo integra tres ámbitos de

investigación principales en la ingeniería del software: la ingeniería de requisitos clásica, el estándar de requisitos de calidad de software SQuaRE y el emergente paradigma del AOSD. Creemos que el uso de estándares, aunque puede parecer restrictivo es en general un paso hacia un entendimiento común para interactuar con grupos de trabajo con diferentes intereses y naturaleza; lo cual es el caso de un equipo de desarrollo de software y la principal causa de fallas en el software. Un ejemplo en la adopción de estándares por la comunidad es el uso de UML. El modelo presentado representa una herramienta útil en las etapas tempranas del desarrollo, por ejemplo durante la modelación de la arquitectura del sistema, cuando los requisitos deben ser claramente identificados, clasificados y cuantificados. Su utilización como soporte en la definición de un proceso de ingeniería de requisitos de calidad es parte de nuestro trabajo en progreso. En particular el modelo será aplicado en la caracterización del dominio de los sistemas fiables, donde el manejo temprano de las incumbencias transversales es de crucial importancia.

Referencias

1. AOSD Home Page: http://www.aosd.net
2. Brito, I. and Moreira, A.: Advanced Separation of Concerns for Requirements Engineering. VIII Jornadas de Ingeniería del Software y Base de Datos, JISBD 2003. Alicante, España (2003). Disponible en: http://citi.di.fct.unl.pt/member/member_act.php?id=30&type=publications
3. Brito, I. and Moreira, A.: Towards a Composition Process for Aspect-Oriented Requirements. Early Aspects Workshop at AOSD Conference. Boston (2003). Disponible en: http://citi.di.fct.unl.pt/member/member_act.php?id=30&type=publications
4. Chirinos, L., Losavio, F. and Matteo, A.: Identifying Quality-based Requirements. Information Systems Management (ISYM). Auerbach Publications, 21(1), 15-21, (2004).
5. Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers (2000).
6. Early Aspect Home Page: http://www.early-aspects.net
7. Filman, R., Elrad, T., Clarke, S. and Aksit, M. Aspect-Oriented Software Development. Addison Wesley, Boston. (2005).
8. IEEE. Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Std. 1471-2000.
9. ISO/IEC 25030: Software Engineering. Software Product Quality Requirements and Evaluation (SQuaRE). Quality Requirements (2007).
10. ISO/IEC WD 25010: Software Engineering. Software Product Quality Requirements and Evaluation (SQuaRE). Quality Model and guide. Draft (2006).
11. ISO/IEC 9126-1: Information Technology - Software Engineering Product Quality. Part 1: Quality Model (2001).
12. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M. and Irwin, J.: Aspect-Oriented Programming. ECCOP'97 Object-Oriented Programming, 11th European Conference, M. Aksit y S. matsouka, Eds. LNCS 1241, 220-242. (1997).
13. Laddad R. AspectJ IN ACTION. Practical Aspect-Oriented Programming. Manning Publications. 2003
14. Lamsweerde, A.: From system goals to software architecture. In M. Bernardo and P. Inverardi, editors, SFM, Formal Methods for Software Architectures. LNCS Springer-Verlag, 25-43 (2003).
15. Navarro, E., Letelier, P. and Ramos, I.: Goals and Quality Characteristics: Separating Concerns. In Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop (AOSD). Lancaster (2004).
16. Rosenhainer, L.: Identifying Crosscutting Concerns in Requirements Specifications. In Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design Workshop (AOSD). Lancaster (2004).
17. Sommerville, I. and Sawyer, P. Requirements Engineering. A Good Practice Guide. John Wiley and Sons, New Cork (1997).
18. Van den Berg, K., Conejero, J., and Chitchyan, R.: AOSD Ontology 1.0 - Public Ontology of Aspect-Orientation. Technical Report AOSD-Europe-UT-01, AOSD-Europe, May (2005).
19. Wiegers, K. Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. Microsoft Press, Washington, USA (2003).