Revista EIA, ISSN 1794-1237 Número 4 p. 95-102. Noviembre 2005
Escuela de Ingeniería de Antioquia, Medellín (Colombia)

ANÁLISIS DE LA TRAZABILIDAD DESDE LA PERSPECTIVA DE LA ORIENTACIÓN A ASPECTOS

MARTA SILVIA TABARES*

RESUMEN

En este artículo se presenta un análisis acerca de la trazabilidad de los intereses de un sistema de software, desde el enfoque del desarrollo de software orientado a aspectos. Para esto se abordarán los conceptos generales de trazabilidad, de la orientación a aspectos y de las características que esta técnica puede proveer para verificar la fiabilidad y evolución de los requisitos en el proceso de desarrollo de software. Tratar la trazabilidad desde este enfoque pretende orientar al desarrollador a identificar elementos generales para determinar la traza de los intereses de corte transversal cuando estos afectan otros intereses del sistema y a su vez cómo diferentes piezas de un artefacto de software asociadas a este tipo de intereses han evolucionado y afectado a otras a lo largo del ciclo de vida de desarrollo.

PALABRAS CLAVE: trazabilidad; orientación a aspectos; intereses; intereses de corte transversal; ingeniería de software; aspectos.

ABSTRACT

In this article an analysis about traceability concerns from the aspect-oriented software development over a software system is presented. To achieve that, the general concepts about traceability and the aspect oriented approaches is treated; also the way of this technique can provide elements to verify reliability and evolution of the requirements in the software development process. To treat traceability from this way will drive to developer to identify general elements necessary to determine trace of crosscutting concern when these affect other concerns, and in turn how different pieces of a software artifact associated to this concern type have evolved and affected others along the development of life cycle.

KEY WORDS: traceability; aspect-oriented; concerns; crosscutting concerns; software engineering; aspects.

* Ph. D(c) en Ingeniería de Sistemas de la Universidad Nacional de Colombia. Docente del Área de Ingeniería de Software y Bases de Datos de la Escuela de Ingeniería de Antioquia. pfmstabare@eia.edu.co

Artículo recibido 22-VIII-2005. Aprobado con revisión 8-XI-2005

Discusión abierta hasta mayo 2006

1. INTRODUCCIÓN

La complejidad de los modelos de negocio y su evolución no controlada dentro de la organización son un reto constante para la ingeniería de software, ya que los modelos y programas de software que soportan el negocio deberán ayudar a controlar dicha complejidad. Esto motiva a la innovación de nuevos métodos, técnicas y herramientas de desarrollo que permitan a los desarrolladores enfrentar diferentes situaciones de una forma más sencilla, sin perder consistencia y calidad en el proceso de desarrollo y en los productos de software.

Para controlar la complejidad adquirida en los sistemas de software, Dijkstra [1] y Parnas [2] introducen la separación de intereses (*separation of concerns* [1]) como un principio general de la ingeniería de software. Este significa que un problema es separado en diferentes intereses, que pueden ser resueltos separadamente sin requerir conocimiento detallado de las otras partes, y así combinarlos finalmente en un resultado.

Los diferentes enfoques de desarrollo han girado alrededor de este principio tratando de encontrar mejores formas de descomposición del problema. Inicialmente fue la técnica de la descomposición funcional para estructurar la funcionalidad de los programas, pero la carencia de características como la orientación al reuso, trabajar los datos aparte de las funciones, entre otros, hicieron necesario indagar otras alternativas de descomposición. Así surgen la orientación a objetos (mejorando la reutilización de un sistema por medio de características como la generalización, ocultamiento de información y polimorfismo) y la separación de requisitos funcionales y no funcionales [4].

No obstante, desde la perspectiva orientada a objetos se encuentra que los intereses son difíciles de factorizar en módulos separados. Por ejemplo, el código del comportamiento de "seguridad" no está bien separado en una clase o método, pero podría ser tratado de una forma que independientemente cortara (atravesara) a muchos de ellos. La separación de este tipo de intereses, intereses de corte transversal[2], es tratada por la orientación a aspectos y provee a los desarrolladores una técnica para identificar y conservar intereses de comportamiento común que puedan cruzar transversalmente diferentes módulos del núcleo del sistema.

Realizar el proceso de software desde esta perspectiva lleva a analizar cómo esta técnica puede facilitar el logro de factores de calidad como la trazabilidad a lo largo del ciclo de vida de desarrollo, ya que la vida de estos intereses y los requisitos asociados a ellos se irán refinando[3] a lo largo de la transformación que sufran los artefactos del sistema en diferentes niveles de granularidad.

En este artículo se realizará un análisis acerca de la trazabilidad de los intereses de un sistema de software, desde el enfoque del desarrollo de software orientado a aspectos, donde se muestran algunos factores que se deben tener en cuenta para realizar la traza de los intereses en diferentes niveles de granularidad, cómo un interés afecta a otros intereses, cómo un interés de corte transversal afecta otros intereses, cómo diferentes piezas de un artefacto de software (v. g. requisito) asociadas a un interés han evolucionado y afectado otras piezas o artefactos y qué elementos se tendrían que tener en cuenta para verificar la fiabilidad que proporciona un interés de corte transversal sobre otros intereses o módulos del núcleo del sistema a lo largo del ciclo de vida.

Con el fin de profundizar en la problemática enunciada y las teorías que la respaldan, en la sección 2 se introduce la orientación a aspectos y la concepción básica de la trazabilidad en el desarrollo de software; en la sección 3 se hace un análisis descriptivo de la trazabilidad desde la perspectiva de la orientación a aspectos; en la sección 4 se presentan algunos trabajos recientes que aportan a la trazabilidad en el desarrollo de software orientado a aspectos, y en la sección 5 se presentan conclusiones del tema tratado.

1 IEEE especifica que los intereses *(concerns)* para un sistema son "esas competencias que pertenecen al desarrollo del sistema, sus operaciones o cualquier otro aspecto que sea crítico o de otra forma importante para uno o más participantes [3]".

2 Este término corresponde a "*crosscutting concern*" en el idioma inglés.

3 El refinamiento (análisis *top-down*) tiende a resultar en una "relación entre elementos de diagramas", donde el refinamiento lleva a que elementos de alto nivel sean descompuestos y relacionados con uno o muchos elementos de bajo nivel (proceso inverso a la abstracción).

2. ANTECEDENTES

2.1 La orientación a aspectos

La orientación a aspectos (*Aspect-Oriented, AO*) emerge como una alternativa de separación de intereses o mecanismo de descomposición/composición de sistemas de software. Busca simplificar la complejidad del sistema y facilita la trazabilidad y la evolución de los sistemas de software.

La AO proporciona elementos que permiten a los desarrolladores la separación de "*intereses de corte transversal*", que puedan cruzar diferentes módulos del núcleo del sistema, ser identificados como ortogonales y actuar como un mecanismo de control sobre intereses comunes. Este comportamiento común podría corresponder a políticas, reglas de negocio, vistas de usuario, procesos de negocio y atributos de calidad tales como *logging*, trazabilidad, seguridad, persistencias, rendimiento, entre otros.

Con esta técnica surgen la programación orientada a aspectos (Aspect-Oriented Programming, AOP) y el desarrollo de software orientado a aspectos (*Aspect Oriented Software Development, AOSD)*.

La AOP se define como un mecanismo que ayuda a resolver problemas complementarios de código disperso (*scattered*) y enredado (*tangled*) que no se solucionan fácilmente con metodologías tradicionales, provee una unidad modular llamada *aspecto* y un mecanismo de *tejido*, que permite entremezclar unidades modulares de comportamiento común con otras unidades modulares básicas del sistema [5].

El AOSD permite tratar los aspectos a lo largo del ciclo de vida de desarrollo [6]. Se fundamenta en un conjunto de aproximaciones que tratan los intereses de corte transversal como elementos de primera clase y determinan cómo el comportamiento común puede ser modelado y trazado como artefactos aspectuales a lo largo del ciclo de vida de desarrollo.

2.2 Trazabilidad

Como en cualquier proceso de software, establecer la trazabilidad surge de la necesidad de poder hacer un seguimiento al cumplimiento de los requisitos a lo largo del ciclo de vida. Esto debido al hecho de ver con frecuencia que, a medida que el desarrollador avanza en sus tareas de diseño y codificación, se va dificultando la ubicación de los elementos del sistema que deben ser modificados ante una necesidad de cambio en los requisitos de un sistema que ya se encuentra en operación.

La ingeniería de software estandariza la *trazabilidad* como un atributo de calidad que permite validar y controlar la transformación de los elementos y artefactos de software (referentes [8, 9, 10, 11]: IEEE Std. 830, 1984; IEEE Std.982.1, 1989; IEEE Std. 1219, 1992; ISO9000-3, 1991; SEI, 1994, EEA). La ISO 9000.2000 define la trazabilidad como "la habilidad para recuperar historia, uso y localización de ítem o actividades similares por medio de una identificación registrada" [12].

Se define también como "la habilidad para determinar realmente cómo una pieza de un artefacto de software (requisito, diseño, código) afecta a otros. La trazabilidad hace posible buscar un cambio a un requisito y encontrar sus partes en los detalles del diseño y el código que son afectados por el cambio. Es esencial para mantener los documentos de

requisitos y diseño al día con respecto a la evolución del código" [14]. En la figura 1 se muestra una representación de las trazas de los requisitos y piezas de artefactos que podrían aparecer en el refinamiento de un sistema de software.

En general, los métodos y modelos de trazabilidad [13, 15, 16] se basan en el proceso iterativo de desarrollo de software y sobre este se llegan a realizar, de forma general, actividades tales como determinar los elementos por trazar, trazar los elementos de un modelo a otro modelo, analizar los resultados de la trazabilidad, discutir incertidumbres con los desarrolladores y adaptar los modelos para la trazabilidad.

La trazabilidad está determinada por las diferentes dependencias de traza (relaciones entre elementos trazados) que son valoradas por los participantes (*stakeholders*) del sistema de software. Ellos determinarán la validez, flujo, evolución, metas y prioridades de los requisitos. Pero aun así hay que tener cuidado, puesto que la información vista desde los participantes puede ser muy confusa y dispersa.

Algunos problemas generales que podrían ser detectados al aplicar trazabilidad son: fallas en la especificación de las dependencias funcionales, tanto entre niveles de la jerarquía como entre módulos en los que se divida el sistema (vínculos y restricciones), mal análisis de dependencias de flujo entre componentes, surgimiento de subgrafos inconexos que llevará a generar grupos de funcionalidad separados, y falta de completitud y consistencia semántica entre los artefactos de diferentes niveles de granularidad.

Esto ha motivado que la concepción y aplicación de la trazabilidad evolucione a partir de nuevas técnicas de análisis y diseño que posibiliten la traza y el chequeo de los artefactos de una forma más dinámica, fiable y consistente, en diferentes niveles de granularidad.

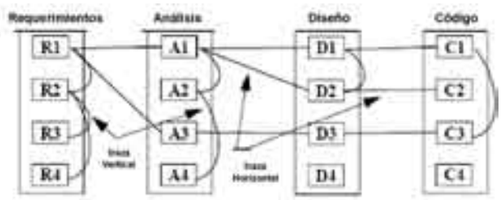

Figura 1. Representación de la trazabilidad en diferentes niveles de abstracción del ciclo de vida de desarrollo [13].

3. LA TRAZABILIDAD VISTA DESDE LA ORIENTACIÓN A ASPECTOS

Como se observa, la trazabilidad está relacionada directamente con el refinamiento de los requisitos y cómo un cambio en el sistema y su funcionalidad podrían afectar la evolución y consistencia de estos a lo largo del tiempo en el ciclo de vida de desarrollo.

Desde el punto de vista de la orientación a objetos, no siempre es posible mantener la traza claramente para capturar de forma aislada la transformación de intereses que no son ortogonales o que cortan transversalmente otros módulos y cuya representación se esparce en otros intereses. Esto lleva a inferir que un cambio sobre un interés que corta a otros intereses se realizaría de forma invasiva en la descripción de ellos, estará altamente acoplado con otros intereses. Además habrá una alta probabilidad de eliminación o adición descontrolada de elementos o artefactos de software al descomponer y componer modularmente, no pudiendo controlar situaciones tales como descartar elementos de los módulos considerados "poco significativos" en el nivel de diseño, sin haber sido contrarrestados con estructuras de módulos altamente cohesivos pero con comportamiento semejante.

Con base en lo anterior, tratar la trazabilidad desde el enfoque AOSD pretende dar fiabilidad a sistemas de software orientando al desarrollador a identificar elementos para determinar la traza desde los intereses de corte transversal cuando estos afectan otros intereses del sistema. Así, se hace necesario identificar el impacto del cambio que motiva la evolución del sistema validando algunas situaciones tales como:

- ¿Qué intereses de corte transversal son afectados?
- ¿Qué modelos y artefactos están o estarán asociados a los intereses afectados?
- ¿Cuál es el nivel de granularidad donde los intereses y sus artefactos son afectados directamente?
- ¿Qué reglas de composición se hace necesario modificar o crear?
- ¿En qué sentido una traza hacia atrás puede proveer elementos aspectuales que obliguen a la construcción de intereses o artefactos no identificados en etapas tempranas?
- ¿A qué aspectos candidatos sólo es posible llevarles la traza en algunas etapas del ciclo de vida?
- ¿Existen semánticas apropiadas para llevar la trazabilidad de los intereses a lo largo del ciclo de vida sin importar los modelos de descomposición y composición utilizados en la construcción del sistema?
- ¿Puede afectar la trazabilidad el hecho de que nuevos artefactos se consideren subsistemas del sistema de software actual?

En la figura 2 se muestra la representación gráfica de una evolución probable de los intereses durante el ciclo de vida. Aquí se propone establecer dos tipos de dependencias de traza durante el refinamiento: entre artefactos del núcleo del sistema ("***traza***", color negro) y entre los artefactos aspectuales ("***traza aspectual***", color gris y raya punteada). La *traza aspectual* representa la trama o argumento que permitirá validar la evolución de los intereses de corte transversal por cada iteración que se tenga de ellos y los posibles cambios en el tejido con los intereses del sistema que estos controlen.

Cada etapa identificará sus artefactos, dependiendo del modelo de análisis y diseño utilizado por el desarrollador. Estos artefactos podrían ser puntos de vista, escenarios, prototipos, casos de uso, diagramas de secuencia y diagramas de clases especificados por UML u otros de acuerdo con el lenguaje o técnica de modelamiento utilizada.

En esta figura se muestra una trazabilidad hacia delante. Un cambio podría darse en cualquier momento del ciclo de vida; por esto es necesario conocer los elementos, artefactos y reglas de composición (posibles tejidos) que podría mantener una *traza aspectual*. En la primera fase se identifican requisitos provistos por diferentes vistas de los participantes en el dominio del problema y se separan los intereses agrupados en intereses funcionales, del sistema y de corte transversal (aún descritos en lenguaje natural). Aquí se establecen trazas verticales (algunas de ellas trazas aspectuales). Cuando ocurra el cambio es posible crear una nueva traza aspectual (o generar una nueva versión de una existente) a partir de la incorporación de un nuevo interés, un nuevo requisito o una regla de composición que afecta a otros intereses. Esta traza aspectual es la base para validar la evolución de intereses de corte transversal en otras etapas del ciclo de vida con sus respectivos artefactos.

En las siguientes fases, en el dominio de la solución, la traza aspectual se asocia a trazas horizontales de artefactos que representan intereses de corte transversal. En este punto será significativo para esta traza el cómo se especifiquen los artefactos que representarán los intereses de corte transversal (esto dependerá en gran medida del lenguaje de representación utilizado, en este caso UML).

De esta forma será posible establecer los parámetros necesarios para construir o mantener la traza de artefactos aspectuales y del núcleo del sistema que sean afectados por cambios a lo largo del ciclo de vida.

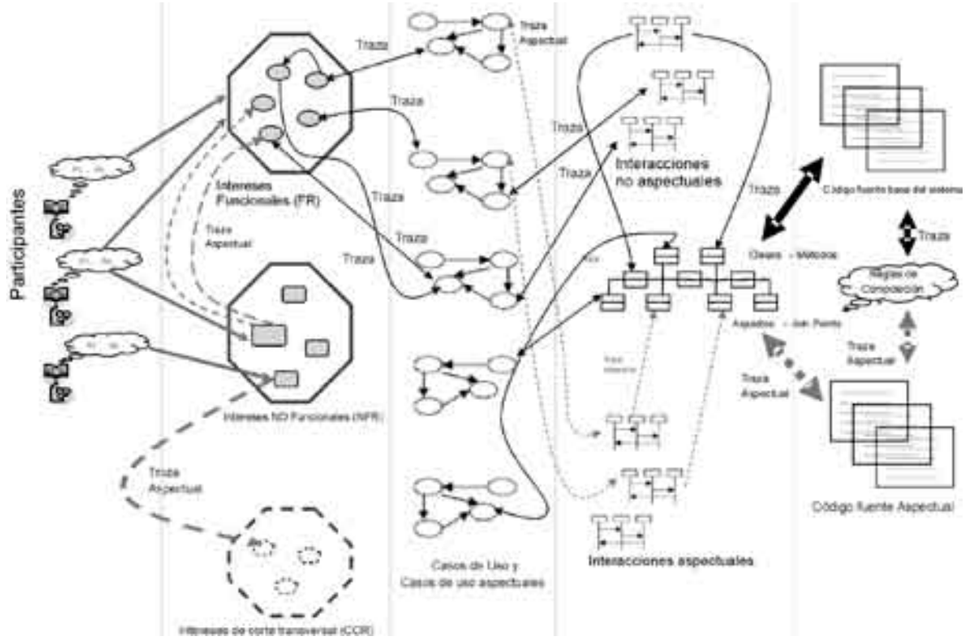

Figura 2. Representación del refinamiento y mapa de trazabilidad aspectual.

4. TRABAJOS RELACIONADOS

A continuación se relacionan las aproximaciones más recientes que aportan al logro de la trazabilidad en el proceso de desarrollo orientados a aspectos. Moreira et al. [17, 18] parten del concepto de la separación multidimensional de intereses en la ingeniería de requisitos. Proveen elementos de traza a partir de la identificación de la influencia de múltiples intereses en el sistema, el análisis de compensación de la especificación de requisitos y la solución de conflictos entre los participantes.

Katz y Rashid [19] identifican la necesidad de tener claridad de los vínculos de trazabilidad entre intereses y que sus compensaciones asociadas sean establecidas y mantenidas desde el nivel de requisitos a lo largo del diseño y la implementación, facilitándose la validación del sistema resultante.

Egyed [20, 21] hace un tratamiento a la trazabilidad desde un contexto genérico de diseño, el cual puede aplicar a cualquier otra estrategia dentro de la orientación a aspectos o la separación multidimensional de intereses. Se basa en escenarios de prueba, elementos de modelo y líneas de código. Establece un modelo para mantener las dependencias de traza a partir del principio de semejanzas entre diferentes elementos de modelo.

Clarke y Baniassad [22] presentan una aproximación de trazabilidad desde la elicitación de requisitos hasta su implementación. Para esto desarrolla un modelo de análisis léxico sobre los requisitos (theme/Doc) que son separados funcionalmente para luego obtener artefactos de diseño encapsulados en unidades llamadas "temas" (theme/UML). Este modelo permite la separación de temas de corte transversal que afectan a otros temas por medio de un patrón de composición que habilita la trazabilidad de estos y otros temas a lo largo del ciclo de vida.

Bakker [7] presenta la aproximación más concreta acerca de trazabilidad de intereses. Esta se crea, bajo XML, un "esquema de modelo de intereses" que permite determinar el impacto del escenario de cambio para los intereses arquitectónicos y para declaraciones de requisitos. Dicho esquema debe ser un "interés neutro", es decir, ser capaz de capturar intereses arbitrarios, debe soportar múltiples clasificaciones o dimensiones de intereses, debe ser neutral con respecto a los tipos de artefactos que puedan ser modelados y debe representar también relaciones entre los intereses.

5. CONCLUSIONES

Como se pudo observar, tratar la trazabilidad no es algo nuevo, pero sí un factor de calidad que debe acompañar la técnica de análisis y diseño utilizada en el desarrollo del software. Trabajarla desde la perspectiva de la AO motiva a explorar la posibilidad de establecer métodos de trazabilidad que provean un mayor número de factores que permitan determinar los posibles elementos de traza identificados en la separación de intereses de corte transversal; de esta forma podría constituirse en la base para plantear arquitecturas provistas de modelos aspectuales que soporten la adaptabilidad del sistema a cambios imprevistos generados por las organizaciones.

Especificar la traza desde esta perspectiva AOSD conducirá al desarrollador a tener elementos para validar cómo los intereses de corte transversal son comúnmente especificados en diferentes niveles de abstracción y cómo ellos afectan otros intereses del sistema. También cómo diferentes piezas de un artefacto de software asociadas a este tipo de intereses han evolucionado a otras piezas a lo largo del ciclo de vida de desarrollo.

Es pertinente considerar un método genérico de trazabilidad que pueda ser aplicado sobre diferentes modelos orientados a aspectos, basado en los trabajos enunciados, que permita trazar cualquier tipo de evolución de los artefactos aspectuales en un sistema de software.

Por último, el análisis presentado en este artículo se constituye en una base teórica, que se tomará en cuenta para explorar la trazabilidad sobre la separación multidimensional de intereses en la ingeniería de requisitos y plantear, a partir de dicha investigación, un método formal de trazabilidad que aportará al AOSD.

6. BIBLIOGRAFÍA

[1] E. W. Dijkstra, A discipline of programming, Prentice Hall, Englewood Cliffs, NJ, 1976.

[2] Parnas D. L., On the criteria to be used in decomposing systems into modules, Communications of the ACM, Volume 15, No. 12, 1972.

[3] IEEE. IEEE Recommended practice for architectural description of software-intensive systems. IEEE Std. 1471-2000. Approved 21 September 2000.

[4] Sommerville I., Ingeniería de software, 6 ed., Addison Wesley, 2002.

[5] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W. G. Getting started with AspectJ. Comm. of the ACM, v. 44, n. 10, p. 59-65. Oct. 2001.

[6] Tekinerdogan B., Moreira A., Araújo J., Clements P. (eds.), Early aspects: aspect-oriented requirements Engineering and Architecture design. Workshop Proceedings 21 March, 2004 Lancaster, UK. http://early-aspects.net/.

[7] Bakker J. Traceability of concerns : An approach to support traceability of concerns across requirements analysis and architecture design life cycle phases. Magister Thesis, University of Twente - Julio 2005.

[8] IEEE Std. 1219 (1992). IEEE standard for software maintenance. Institute of Electrical and Electronic Engineers, New York, USA.

[9] IEEE Std. 830 (1984). IEEE Guide to software requirements specifications. Institute of Electrical and Electronic Engineers, New York, USA.

[10] IEEE Std. 982.1 (1989). IEEE standard dictionary of measures to produce reliable software. Institute of Electrical and Electronic Engineers, New York, USA.

[11] ISO9000-3 (1991). Quality management and quality assurance standards. International Organization for Standardization, Geneva, Switzerland.

[12] Icontec. Sistema de gestión de la calidad : fundamentos y vocabulario. Norma Técnica Colombiana NTC-ISO 9000. 2000-12-15.

[13] Lindvall M., A study of traceability in object-oriented systems development. Thesis (Licenciated 462, Department of Computer and Information Science), Linkping University, Sweden, 1994.

[14] Clarke S., Walker R. J. Towards a standard design language for AOSD. In: Proceedings of the 1st International Conference on Aspect-Oriented Software Development (AOSD), Enschede, The Netherlands, April 2002.

[15] Piattini M. G., García F. O., Calidad en el desarrollo y mantenimiento del software. Alfaomega, RA-MA.

[16] Roggio R. Use cases and traceability: a marriage for improved software quality. Proceedings of the 16th Annual NACCQ, Palmerston North, New Zealand, July, 2003 (eds) Mann, S. and Williamson, A. www.naccq.ac.nz.

[17] Moreira A., Rashid A., Araújo J., Multidimensional separation of concerns in requirementes engineering. Internacional Conference on Requirements Engineering (RE), IEEE Computer Society (accepted to appeear). (2005).

[18] Moreira A., Araújo J., Rashid A., A concern-oriented requirements engineering model. International Conference on Advanced Information Systems Engineering, Porto, Portugal, June 13-17. Editors: O. Pastor, J. Falcão e Cunha, Springer-Verlag. Volume 3520, p. 293-308. (2005).

[19] Katz S., Rashid A., From aspectual requirements to proof obligations for aspect-oriented systems, Proc. RE, 2004, IEEE CS Press, pp. 43-52.

[20] Egyed, A. Reasoning about trace dependencies in a multi-dimensional space, Proceedings of the 1st International Workshop on Traceability, co-located with ASE 2002, Edinburgh, Scotland, UK, September 2002.

[21] Egyed, A. Resolving uncertainties during trace analysis. Proccedings of 12th ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE), Irvine, CA, November 2004, p. 3-12.

[22] Clarke S., Baniassad E. Aspect-oriented analysis and design. The theme approach. Addison-Wesley, Object Technology Series, 2005.