

Trabajo de grado

Licenciatura en Informática (Plan 90)

Facultad de Informática, Universidad Nacional de La Plata



Mecanismos de rastreo en el desarrollo de software dirigido por modelos

Alumno: Mariano Gabriel Gili

Directora: Dra. Claudia Pons

Diciembre de 2014

Trabajo de grado

Licenciatura en Informática (Plan 90)

Facultad de Informática, Universidad Nacional de La Plata



Introducción

Definición de traceability



Según el Glosario Estándar de Términos de la Ingeniería de Software del IEEE:

El grado o nivel en el cual una relación puede ser establecida entre dos o más productos del proceso de desarrollo, especialmente entre productos que tengan una relación de predecesor-sucesor o principal-secundario; por ejemplo el grado en el cual el requerimiento y el diseño de un componente de software se corresponden.

Desde un punto de vista más cercano al contexto de MDD:

cualquiera de las complejas relaciones lógicas que existen entre los distintos artefactos que se presentan en cualquier momento del ciclo de vida del desarrollo de software, el establecimiento de estas relaciones y/o el mantenimiento de las mismas.

Beneficios



- ✓ **Análisis de Sistemas:** entender la complejidad de un sistema, navegando el modelo de tracelinks resultante de la ejecución de las distintas cadenas de transformación.
- ✓ **Análisis de Cobertura:** el uso de traceability es crucial a la hora de determinar si todos los requerimientos fueron cubiertos.
- ✓ **Análisis de Impacto de Cambios:** permite como los cambios en un modelo repercutirán en los modelos relacionados; también, saber la dependencia de las entidades relacionadas, lo cual ayuda a determinar la necesidad de un cambio.
- ✓ **Análisis de Huérfanos:** serán aquellos artefactos que no se encuentren relacionados mediante ningún tracelink.
- ✓ **Análisis de Requerimiento:** por ejemplo ayuda a identificar el artefacto particular que demanda una propiedad específica; como así también encontrar y resolver un conjunto de requerimientos contradictorios.

Beneficios



- ✓ **Comprensión del Software e Ingeniería Inversa:** crucial cuando se necesita identificar todas las entidades relacionadas a una en particular, entender el tipo de relación existente, identificar las abstracciones, es decir los patrones de diseño y/o estilos de arquitectura, etc.
- ✓ **Apoyo en la toma de decisiones:** para justificar una decisión dado que facilita el entendimiento de los factores y metas que influyen en la misma; también resulta muy útil en el análisis y evaluación de distintas propuestas.
- ✓ **Configuración del sistema y Versionado:** para identificar las restricciones entre los componentes, los cambios necesarios para resolver una restricción, las diferencias entre dos versiones distintas de un mismo artefacto y el impacto que estas diferencias tendrán.

Inconvenientes



- ✖ **Alto costo** de la creación de la información de traceability.
- ✖ La necesidad de **mantenimiento manual** de la información.
- ✖ La falta de **heurísticas** que determinen qué información de los links deben ser guardadas.
- ✖ Discrepancias entre los distintos **roles** que ejercen los usuarios de la información de traceability.
- ✖ La carencia de **soporte adecuado de las herramientas**.
- ✖ El uso de **diferentes lenguajes**. Los requerimientos se escriben en lenguaje natural, los programas en algún lenguaje formal.
- ✖ Diferencias entre los distintos **niveles de abstracción** en los cuales los artefactos describen el sistema de software.

Trabajo de grado

Licenciatura en Informática (Plan 90)

Facultad de Informática, Universidad Nacional de La Plata



Teoría de traceability

Meta-modelos de traceability



Para llegar a la definición de cualquier propuesta de traceability se necesita de un **modelo** en el cual se especifiquen los conceptos, las reglas y las relaciones que existen, por ejemplo entre los artefactos y los tracelinks.

Meta-modelo de traceability de

propósito general

caso específico

Meta-modelo de traceability de propósito general



Meta-modelo genérico que permite capturar tracelinks de cualquier tipo de artefactos de un determinado modelo.

Un tracelink se puede conectar con cualquier número de elementos, de cualquier tipo y de cualquier modelo.

✓ Simplicidad
✓ Uniformidad } mejora la **interoperabilidad de las herramientas**

✗ No capta casos específicos de tracelinks fuertemente tipados



links ilegítimos

Meta-modelo de traceability de caso específico



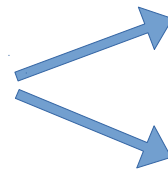
Para cada escenario de traceability se define un **meta-modelo específico**. Este meta-modelo captura links fuertemente tipados para casos específicos con una semántica bien definida.

- ✓ tipado fuerte
- ✓ restricciones asociadas



links legítimos

- ✗ Un meta-modelo para cada caso específico



Esfuerzo de construcción

Herramientas específicas

Tipos de tracelinks

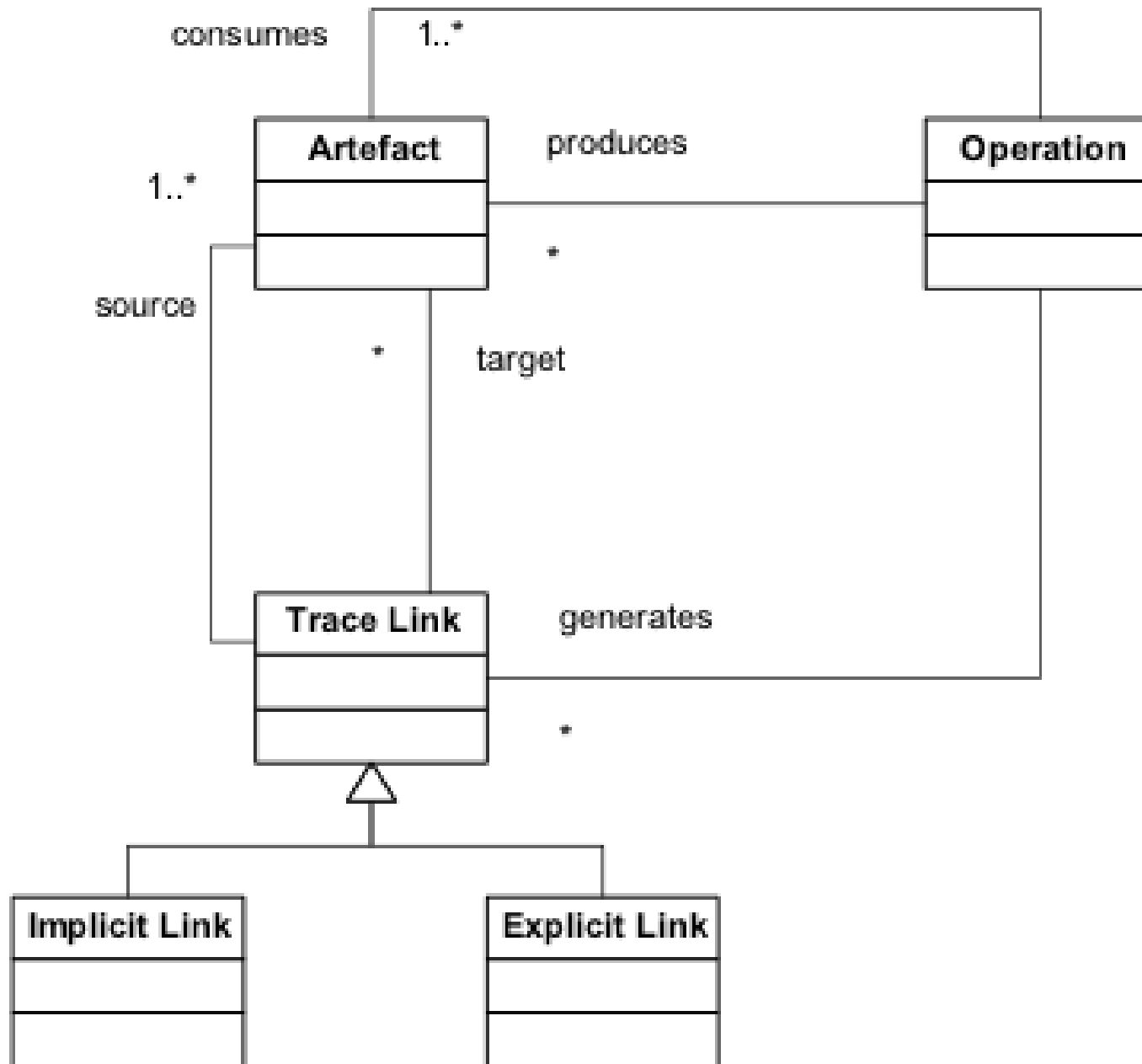


La definición semántica de los tracelinks permite descubrir los distintos tipos de links que se pueden presentar, entender su significado y uso.

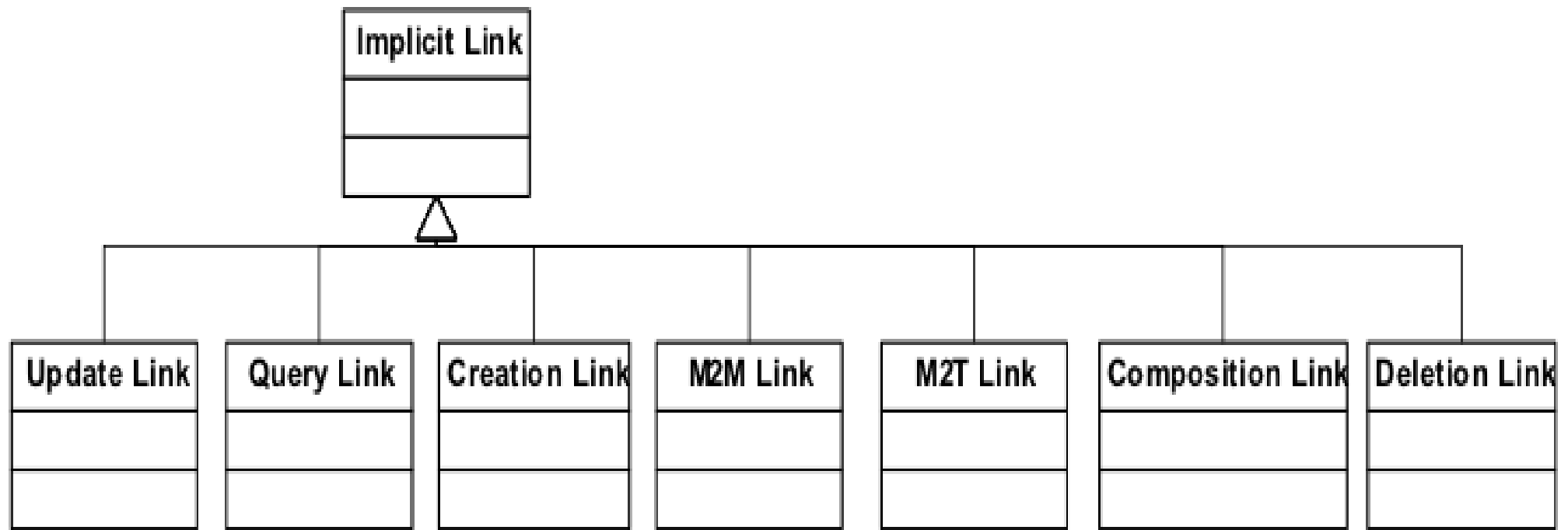
No predefinir la semántica de un tracelink correctamente, puede resultar en fallas de razonamientos y/o entendimientos.

Ejemplo: una jerarquía que representa una **clasificación genérica de traceability**

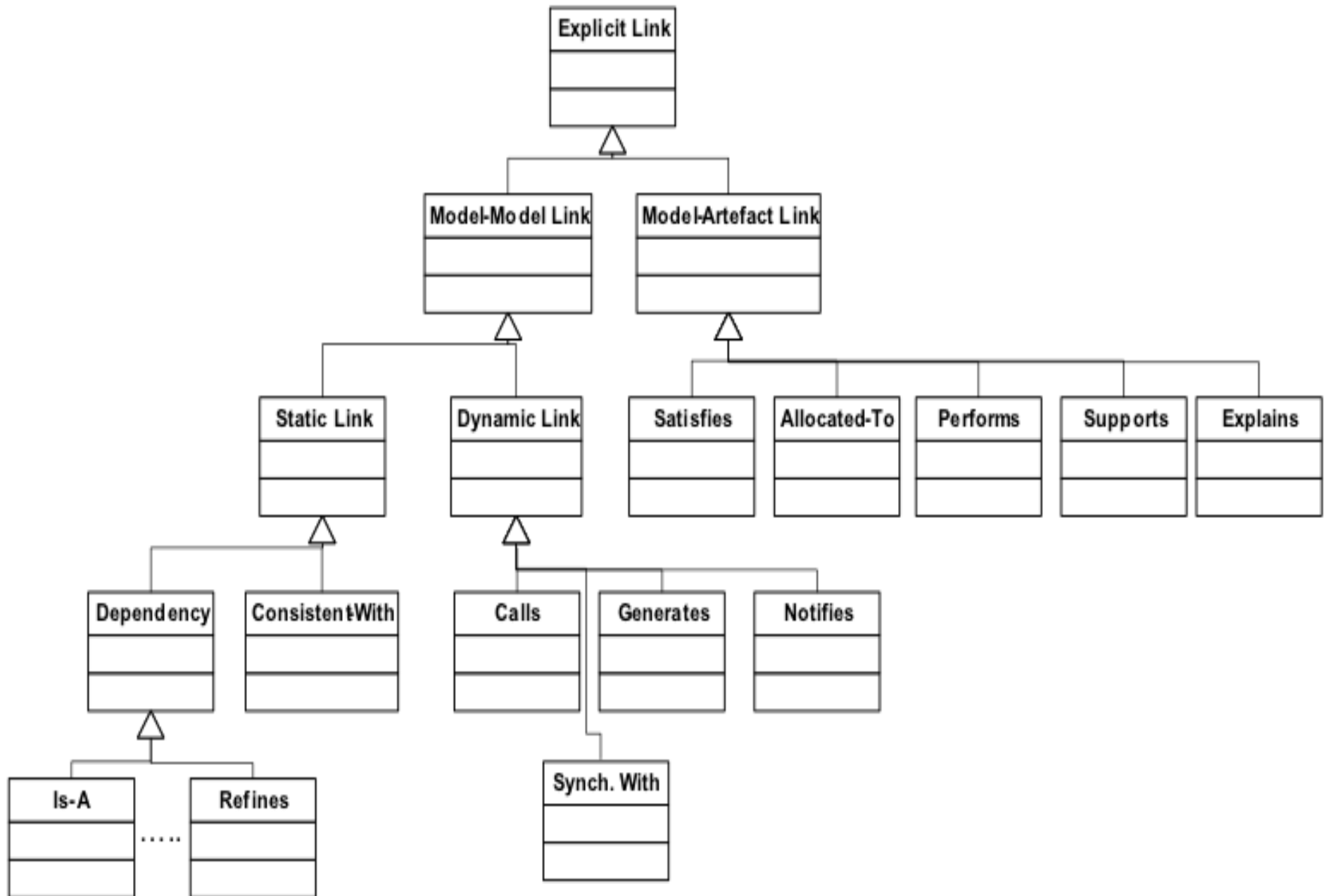
Clasificación genérica



Links implícitos



Links explícitos

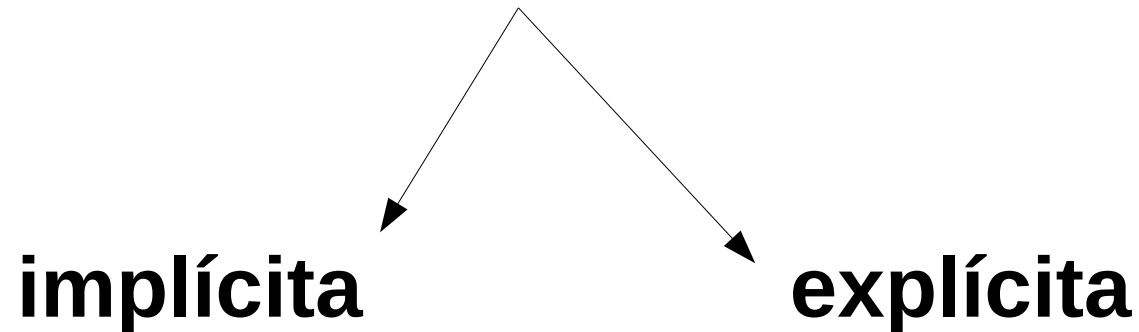


Generación de trancelinks



Durante una transformación de modelos se pueden presentar dos enfoques de generación de trancelinks:

Generación de trancelinks



Generación de tracelinks implícita



- ✓ no es necesario **ningún esfuerzo adicional** para obtener los tracelinks que relacionan los modelos de entrada y salida.
- ✗ El **meta-modelo de traceability** tiene que ser fijo, lograr un estándar es muy complejo.
- ✗ cuando se generan tracelinks para todos los elementos del modelo referenciado, una cantidad considerable de ellos puede volverse incomprensible e inútil.
- ✗ en el caso de transformaciones de modelos grandes y complejos, problemas de rendimiento.
- ✗ la configuración del nivel de trace granularity varía de un escenario de traceability a otro.
- ✗ un cliente, por ejemplo aduciendo a motivos de seguridad, puede requerir que para cierta información de un modelo no se le generen tracelinks.

Generación de tracelinks explícita

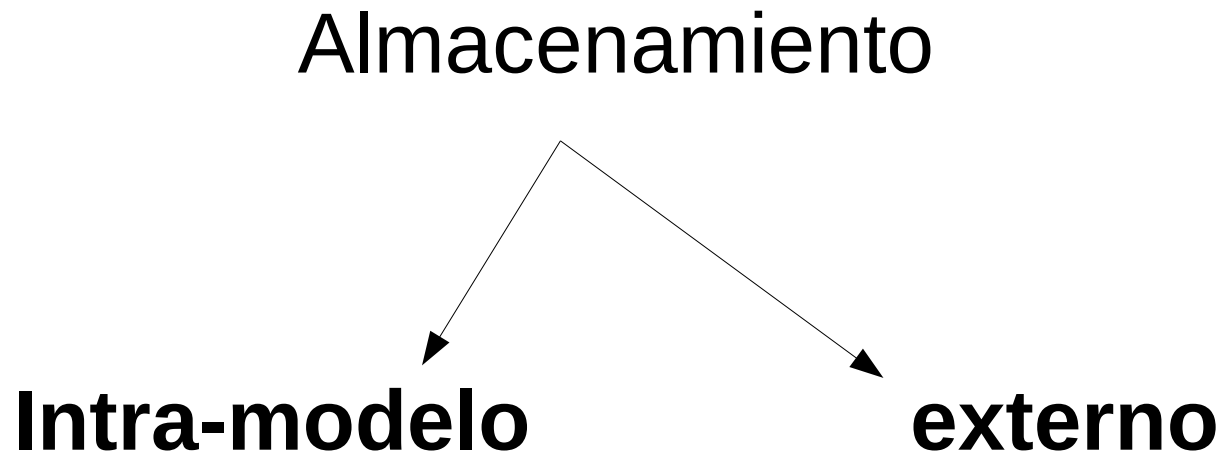


- ✓ Es posible el tratamiento de traceability como un modelo regular complementario al resultado de la transformación, que se puede obtener por la incorporación de reglas de transformación adicionales. Por lo cual, el **meta-modelo es flexible**.
- ✓ Dada la flexibilidad del meta-modelo, **el nivel de trace granularity** de los tracelinks es **fácil de adaptar** al dominio de la transformación
- ✗ **Esfuerzo adicional** para definir las reglas de transformación específicas para traceability.
- ✗ Se **contamina la implementación** de la transformación.
- ✗ Como la definición de las reglas de traceability es responsabilidad del programador, es **propensa a errores** y puede llegar a demandar mucho **tiempo**.

Estrategias de almacenamiento




Dos tipos principales de estrategias a seguir para almacenar y administrar la información de traceability:



Almacenamiento intra-modelo



- ✓ Estrategia **sencilla y amigable**.
 - ✗ si se almacenan los tracelinks sólo en el modelo origen, no son **visibles** en el modelo destino (ídem a la inversa).
 - ✗ si la información es almacenada en ambos modelos, desafío de mantenerla **consistente** por cada cambio que se suceda.
 - ✗ problema de **polución** que se genera en el modelo con la información de traceability.
 - ✗ compleja la tarea de **diferenciar la información** de traceability de los objetos que representan el modelo del dominio.
- 
- ✗ Muy difícil el **análisis automatizado** de la información de traceability.

Almacenamiento externo



- ✓ modelos origen y destino se totalmente **limpios**.
- ✓ el proceso de **análisis de la información** es mucho más fácil que en la otra estrategia.
- ▶ Única condición: los diferentes elementos del modelo tienen que tener **identificadores únicos**.

Trabajo de grado

Licenciatura en Informática (Plan 90)

Facultad de Informática, Universidad Nacional de La Plata



Manos a la obra

Requerimientos



Lograr un **meta-modelo bien simple** en el que se capturen solamente los **conceptos más relevantes** de traceability y se almacene la **información mínima** necesaria.



Independencia del modelo con respecto al dominio



Esquema de propósito general

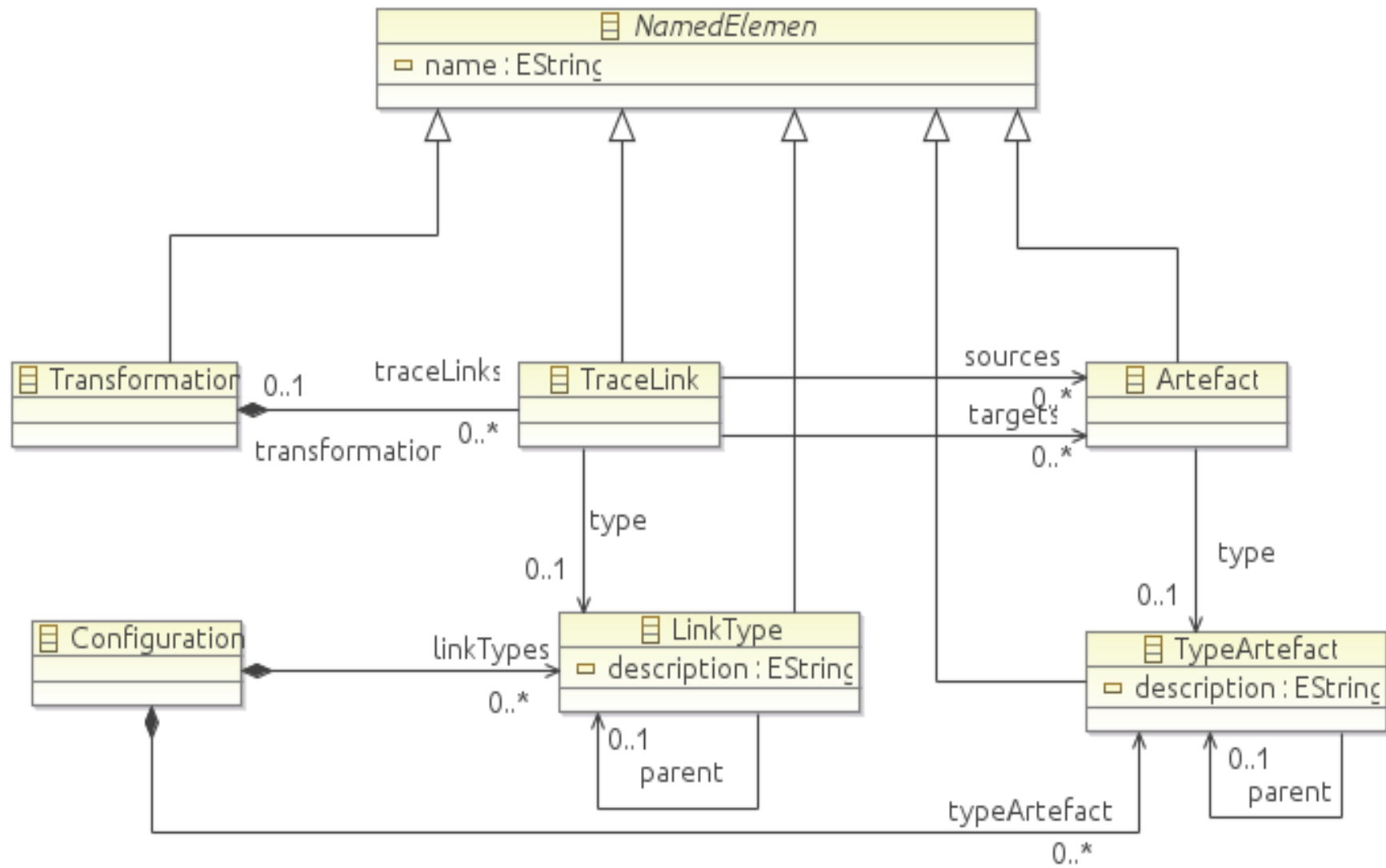
Requerimientos



Solución de los siguientes desafíos:

- ★ Lograr un **esquema** que represente el conjunto de tracelinks **independientemente de cómo fueron generados** (explícita o implícitamente), su **trace granularity** y/o **semántica**.
- ★ Mantener la **comunicación** lo más **simple** posible, entre usuarios y herramientas.
- ★ Tiene que ser **sencillo agregar la funcionalidad de traceability** a una herramienta con este esquema.
- ★ el esquema siempre tiene que ofrecer **información semántica** de los artefactos, con el fin de **facilitar la validación de links legítimos**.
- ★ **Identificar unívocamente** cada artefacto.

Esquema propuesto



El prototipo

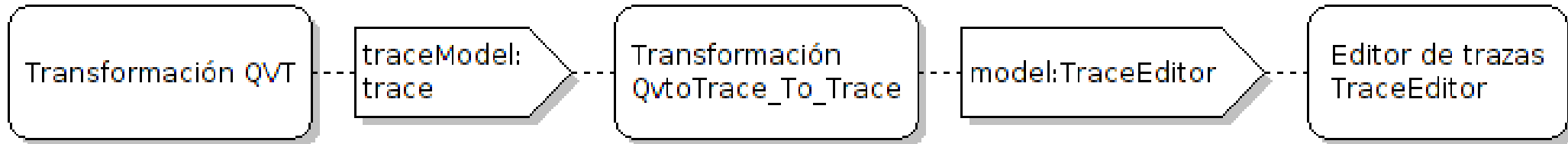


Es una herramienta que, dada la definición de una transformación acompañada con el conjunto de modelos de entrada/origen y de resultado/destino, retorna y muestra de una forma amigable el mapa de tracelinks que se generaron implícitamente producto de su ejecución.

También permite la edición de dicho mapa:

- ✓ crear, editar y eliminar tracelinks
- ✓ editar artefactos de los modelos origen y destino
- ✓ editar transformaciones

Secuencia de uso y arquitectura



Arquitectura

Dos módulos:

- ✓ **QvtoTrace_To_Trace** (Transformación QVT)
- ✓ **TraceEditor**

Editor gráfico y tabular TraceEditor



Ubuntu 9 B I A 100% Quick Access

Project Ex QvtoTrace_To_Trace.traceeditor_diagram

Artefactos orígenes

- NOMBRE: Source_48ce4df0
TIPO: Source_Model
- NOMBRE: Source_669b776
TIPO: Source_Package
- NOMBRE: Source_66917b72
TIPO: Source_Class

Transformaciones y trazas

- TRANSFORMACIÓN: model2RDBModel2
TRAZA: Ejemplo Generates
TRAZA: Trace_7ace664d
TIPO: Implicit
- TRANSFORMACIÓN: das
TRAZA: Trace_49e8838
TIPO: Implicit
- TRANSFORMACIÓN: persistentClass2ta...
TRAZA: Trace_440116e6

Artefactos destinos

- NOMBRE: Target_7813631e
TIPO: Target_Model
- NOMBRE: Target_3a9264b7
TIPO: Target_Schema
- NOMBRE: Target_63b6e435
TIPO: Target_Table

Palette

- Traza
- Origen de la traza
- Destino de la traza

Properties Lista de Trazas

Transformación	Enlace/Traza	Tipo del enlace	Artefacto origen	Tipo del origen	Artefacto destino	Tipo del destino
<< Sin transformación	Ejemplo	Generates	Source_48ce4df0	Source_Model	Target_3a9264b7	Target_Schema
model2RDBModel2	Trace_7ace664d	Implicit	Source_48ce4df0	Source_Model	Target_7813631e	Target_Model
das	Trace_49e8838	Implicit	Source_669b776	Source_Package	Target_3a9264b7	Target_Schema
persistentClass2table	Trace_440116e6	Implicit	Source_66917b72	Source_Class	Target_63b6e435	Target_Table
persistentClass2table	Trace_51108570	Implicit	Source_77e8d632	Source_Class	Target_604ac455	Target_Table
persistentClass2table	Trace_764d4324	Implicit	Source_5bb45e8	Source_Class	Target_604d4374	Target_Table

Editor gráfico y tabular TraceEditor



Ubuntu 9 B I A 100% Quick Access

Project Ex

- Ejemplo_Transform
- models
- >> results
- >> tr
- >> Simpleuml_To
- transforms
- > QvtoTrace to Trace
- >> meta-models
- >> models
- >> results
- QvtoTrace_To_Tr
- QvtoTrace_To_Tr

Outline

QvtoTrace_To_Trace.traceeditor_diagram

Artefactos orígenes

- NOMBRE: Source_48ce4df0
TIPO: Source_Model
- NOMBRE: Source_669b776
TIPO: Source_Package
- NOMBRE: Source_66917b72
TIPO: Source_Class

Transformaciones y trazas

- TRANSFORMACIÓN: model2RDBModel2
TRAZA: Trace_7ace664d
TIPO: Implicit
- TRANSFORMACIÓN: das
TRAZA: Trace_49e8838
TIPO: Implicit
- TRANSFORMACIÓN: persistentClass2ta...
TRAZA: Trace_440116e6

Artefactos destinos

- NOMBRE: Target_7813631e
TIPO: Target_Model
- NOMBRE: Target_3a9264b7
TIPO: Target_Schema
- NOMBRE: Target_63b6e435
TIPO: Target_Table

Palette

- Traza
- Origen de la traza
- Destino de la traza

Properties

Lista de Trazas

Transformación	Enlace/Traza	Tipo del enlace	Artefacto origen	Tipo del origen	Artefacto destino	Tipo del destino
<< Sin transformación	Ejemplo	Generates	Source_48ce4df0	Source_Model	Target_3a9264b7	Target_Schema
model2RDBModel2	Trace_7ace664d	Implicit	Source_48ce4df0	Source_Model	Target_7813631e	Target_Model
das	Trace_49e8838	Implicit	Source_669b776	Source_Package	Target_3a9264b7	Target_Schema
persistentClass2table	Trace_440116e6	Implicit	Source_66917b72	Source_Class	Target_63b6e435	Target_Table
persistentClass2table	Trace_51108570	Implicit	Source_77e8d632	Source_Class	Target_604ac455	Target_Table
persistentClass2table	Trace_764d434	Implicit	Source_6bb4500	Source_Class	Target_604d4774	Target_Table

Tecnologías usadas



- ✓ **Eclipse**
- ✓ **Eclipse Modeling Framework (EMF)**
- ✓ **Graphical Modeling Framework (GMF)**
- ✓ **Query/View/Transformation (QVT)**

Trabajo de grado

Licenciatura en Informática (Plan 90)

Facultad de Informática, Universidad Nacional de La Plata



Terminando

Conclusión



- ✓ Todo proceso de desarrollo debería hacer uso, durante el ciclo de vida completo del software, de una **metodología de traceability** que muestre de forma precisa la relación existente entre todos los artefactos que surgen y forman parte del mismo.
- ✓ La información que se obtenga de un buen y actualizado servicio de traceability es muy valiosa, ya que al disponer de ella en un proyecto será más difícil que el mismo tienda al fracaso.

Trabajos futuros



Lo que el esquema no ofrece

- ✗ Funcionalidad de **versionado**
- ✗ Métodos de **detección** o **información** de **errores**
- ✗ Grafo de tracelinks para **cadenas de transformación**

Mejoras para el prototipo

- ✓ Traceability en **Atlas Transformation Language (ATL)**

Trabajos futuros



- ✓ Mejorar el esquema de traceability propuesto con:
 - funcionalidad de **versionado**,
 - métodos de **detección e información de errores**,
 - organización de los links según la ejecución de una **cadena de transformaciones**.
- ✓ Implementar una solución para **transformaciones ATL**.
- ✓ Mejores opciones de **filtrado o vistas parciales**.
- ✓ Mejorar la **identificación** de un artefacto.

Trabajo de grado

Licenciatura en Informática (Plan 90)

Facultad de Informática, Universidad Nacional de La Plata



¡Muchas gracias!