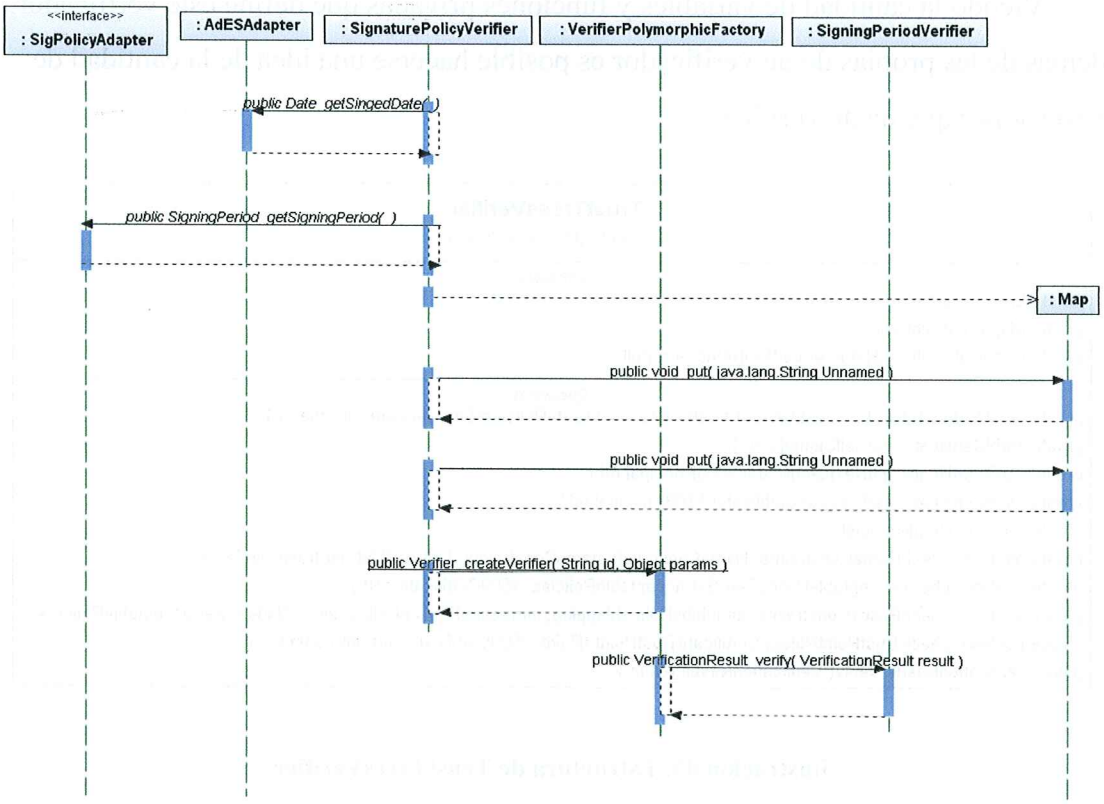


Ilustración 41: Diagrama de secuencia de la llamada a SigningPeriodVerifier dentro de SignaturePolicyVerivier

4.7.4. Estudio concreto de SignerTrustTreesVerifier

Este verificador es completamente distinto al anterior, ya que va a ser reutilizado en diversas ocasiones ha de recibir toda la información necesaria al ser creado. Es decir, no puede obtener la información directamente de las capas de acceso a datos ya que será llamado para verificar distintas restricciones de la política, y en cada caso utilizará datos distintos. También contendrá mucha más lógica de verificación de restricciones, ya que la política es capaz de definir gran cantidad de restricciones sobre el camino de certificación.

El único elemento presente también en otros lugares de la política, de los que componen las reglas sobre el camino de certificación, son las restricciones aplicables al espacio de nombres que, como ya se ha explicado, compondrán otro verificador aparte.

Viendo la cantidad de variables y funciones privadas que define este verificador, además de las propias de un verificador es posible hacerse una idea de la cantidad de restricciones que ha de verificar:

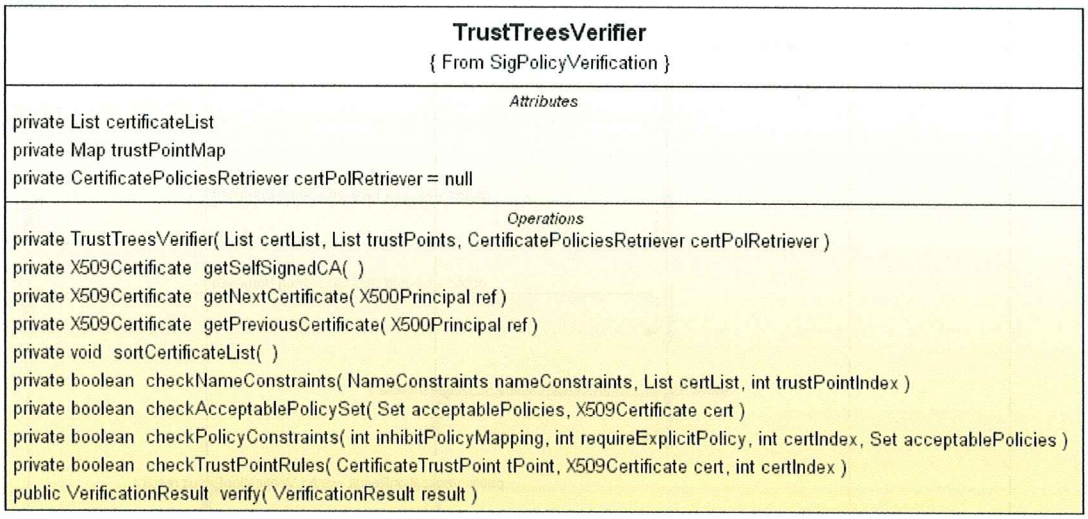


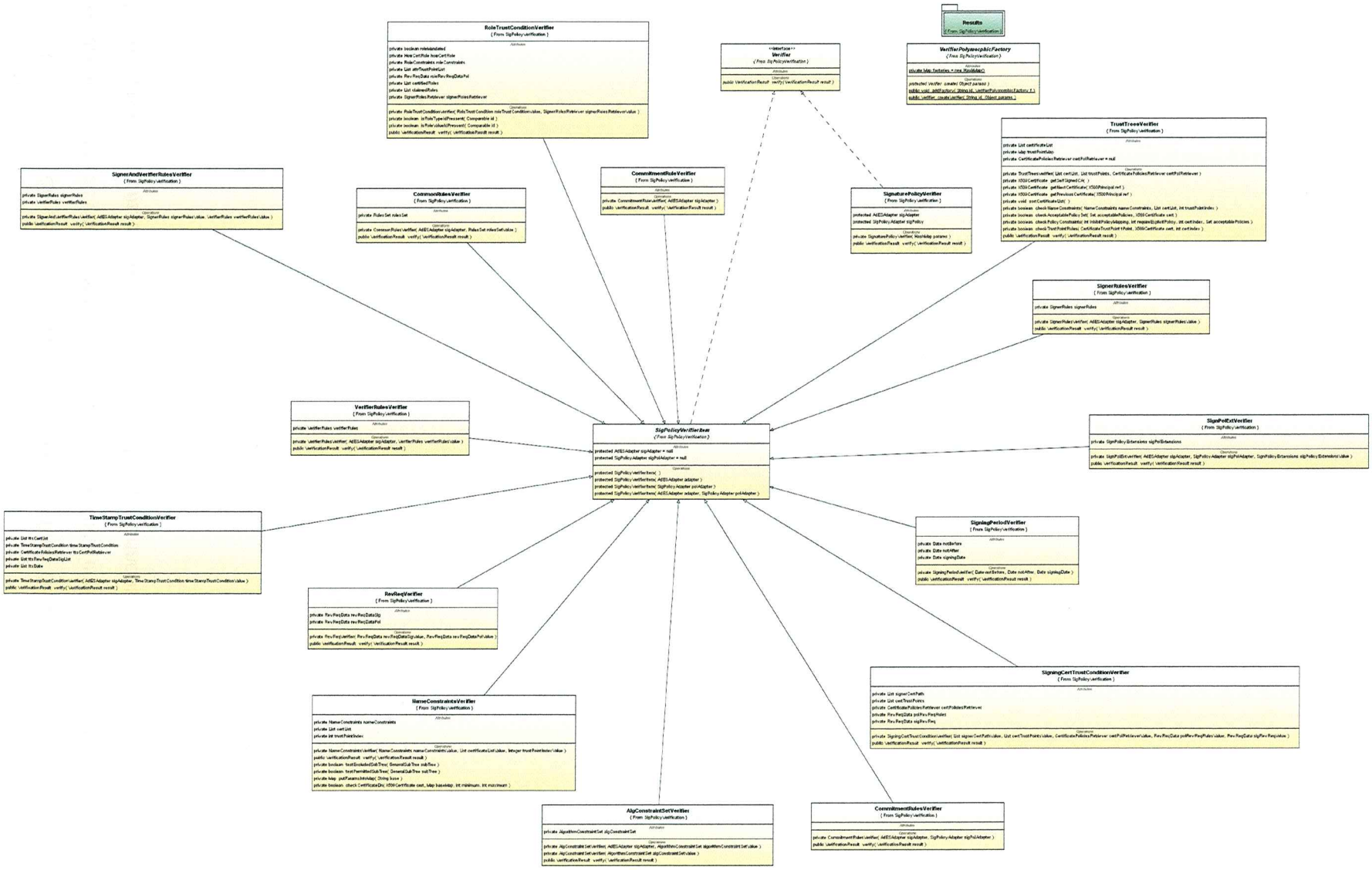
Ilustración 42: Estructura de TrustTreesVerifier

Este verificador incluye funciones privadas, siguiendo la nomenclatura de elementos definidos por la política de firma, para verificar algunas de las reglas concretas que ésta define, junto con otras destinadas a facilitar la tarea de estas. Algun:

- Ordenar la lista de certificados del camino de certificación para que sea más fácil la verificación de las reglas referentes a entidades de certificación y la distancia máxima permitida.
- Encontrar certificados de autoridades de certificación autofirmados. También sirve para realizar una ordenación de los certificados, pero este caso se trata por separado.
- Comprobar las restricciones sobre el espacio de nombres de cada certificado, donde se llamará al verificador correspondiente, *NameConstraintsVerifier*.
- Comprobar si los certificados siguen alguna de las políticas aceptables por la política de firma bajo la cual se verifica.
- Comprobar si los certificados siguientes al *TrustPoint* cumplen las exigencias sobre políticas de firma.

4.7.5. Diagrama del conjunto de verificadores

El siguiente diagrama de clases muestra el conjunto de todos los verificadores y las relaciones que se establecen entre ellos y la interfaz *Verifier* y el elemento *SigPolicyVerifierItem*, tal como se ha explicado en apartados anteriores:


RoleTrustConditionVerifier
{ From SigPolicyVerification }
«Interface»
Verifier
{ From SigPolicyVerification }
Results
{ From SigPolicyVerification }
VerifierPolymorphicFactory
{ From SigPolicyVerification }
TrustTreeeVerifier
{ From SigPolicyVerification }
SignerAndVerifierRulesVerifier
{ From SigPolicyVerification }
CommonRulesVerifier
{ From SigPolicyVerification }
CommitmentRuleVerifier
{ From SigPolicyVerification }
SignaturePolicyVerifier
{ From SigPolicyVerification }
SignerRulesVerifier
{ From SigPolicyVerification }
VerifierRulesVerifier
{ From SigPolicyVerification }
SigPolicyVerifierItem
{ From SigPolicyVerification }
SignPolExtVerifier
{ From SigPolicyVerification }
TimeStampTrustConditionVerifier
{ From SigPolicyVerification }
SigningPeriodVerifier
{ From SigPolicyVerification }
RevReqVerifier
{ From SigPolicyVerification }
SigningCertTrustConditionVerifier
{ From SigPolicyVerification }
NameConstraintsVerifier
{ From SigPolicyVerification }
AlgConstraintSetVerifier
{ From SigPolicyVerification }
CommitmentRulesVerifier
{ From SigPolicyVerification }

5. Implementación

5.1 – Implementación en Java

La implementación del módulo se ha llevado a cabo completamente en Java, por su facilidad a la hora de trabajar con documentos XML e información referente a infraestructuras de clave pública. Además la herramienta de generación y verificación de firma electrónica avanzada existente en el departamento está desarrollada también en este lenguaje, por lo tanto, para llevar a cabo la integración de forma sencilla parece la mejor opción.

Concretamente, el código generado es totalmente compatible con la versión JDK 1.4.2 o superiores, para conseguir una mayor compatibilidad con máquinas virtuales antiguas o de otros proveedores.

5.2 – Alcance de la implementación

Como se ha explicado en apartados anteriores, la implementación no cubre todos los apartados del diseño. Concretamente faltan los siguientes apartados:

- Capa de acceso a la política de firma: Se ha definido el adaptador como un interfaz, con todos los métodos necesarios para la capa de verificación definidos, pero falta su implementación para algún caso concreto de documento de política de firma. Los formatos más importantes para los que se debería implementar son XML y ASN.1.
- Capa de acceso a la firma y datos de verificación: Igual que la capa de acceso a la política de firma, se ha definido la interfaz del adaptador en profundidad, pero falta su implementación. Esto forma parte de la integración con el módulo de generación y verificación de política de firma, ya que gran parte de estos datos se obtienen después de la verificación de la firma.
- Instrucciones de generación y verificación: También forma parte de la integración con el otro módulo. Se ha definido una interfaz para permitir el acceso a esta información por parte del otro módulo, pero seguramente se tengan que hacer modificaciones para una mejor integración en cooperación con las partes encargadas del desarrollo del otro módulo.

- Formato de salida del informe de verificación: Aunque está preparado para ello, no se ha implementado ninguna clase para algún formateado concreto, seguramente por falta de tiempo.

El resto del diseño ha sido implementado completamente, en especial la capa de verificación, que ha sido la más costosa y extensa de implementar, es capaz de verificar todas las reglas que se pueden definir en una política de firma independientemente de su formato de representación, tal como se pretendía en el planteamiento de este proyecto.

5.3 – Clases de test

Para controlar la calidad del código desarrollado y poder demostrar el funcionamiento de los verificadores desarrollados también se han implementado toda una serie de clases de test JUnit que comprueban a fondo el funcionamiento de los distintos verificadores.

Estas clases de test se han ido desarrollando en paralelo con el código de la capa de verificación, de forma que han ido sirviendo para detectar y corregir algunos errores de programación cometido en algunos verificadores a medida que se iban desarrollando. Estas clases de test no cubren todos los verificadores diseñados, sino un 40 o 50% aproximadamente. No obstante hay que comentar que los verificadores que se han comprobado en primer lugar son los de menor nivel, que tienen una lógica mayor de verificación y comprobación de reglas o restricciones, lo que les convierte en las clases más críticas en cuanto a posibles errores de programación, dejando para el final las clases de test de aquellos verificadores que simplemente se limitan a recoger datos y llamar a otros verificadores.

6. Análisis de costes y presupuesto

6.1 – Distintas fases y costes temporales del proyecto

Con el fin de calcular el coste total de este proyecto y realizar un presupuesto se va a exponer en la siguiente lista el conjunto de fases y tareas en las que se puede dividir todo el proceso que ha llevado a la elaboración del mismo. Estas fases incluyen todos los puntos explicados en este documento, con los que se intenta hacer una analogía para una mejor comprensión del mismo. Junto a cada fase se indica el tiempo aproximado dedicado a su elaboración, en horas. Estas fases se han dividido según dos roles básicos: analista y programador.

- Análisis de requisitos y viabilidad → 10 horas de analista
- Análisis de infraestructuras y conceptos relacionados con la firma electrónica avanzada y política de firma → 30 horas de analista
- Análisis completo de la estructura, reglas y restricciones de la política de firma → 40 horas de analista
- Análisis de las capas presentes en el diseño → 10 horas de analista
- Diseño de la capa de acceso a los datos de la política de firma → 20 horas de analista
- Programación de la capa de acceso a datos de la política de firma → 20 horas de programador
- Diseño de la capa de acceso a datos de la firma electrónica avanzada → 10 horas de analista
- Diseño de la capa contenedora del informe de verificación → 10 horas de analista
- Diseño del conjunto de verificadores → 60 horas de analista
- Programación del conjunto de verificadores → 120 horas de programador
- Diseño de las clases de test para la capa de verificación → 10 horas de analista
- Programación de las clases de test de la capa de verificación → 30 horas de programador
- Elaboración de toda la documentación → 60 horas de analista y 60 horas de programador

6.2 – Tabla de costes por fases

Para calcular los costes de cada fase se han establecido los siguientes precios por hora para cada rol o perfil desarrollador:

- Perfil de Analista: 50 €/hora.
- Perfil de programador: 30 €/hora.

La tabla siguiente muestra el conjunto de fases que componen el proyecto, junto con el número de horas dedicadas y el perfil desarrollador asociado a la tarea. También incluye el coste de cada fase y el coste total del proyecto, así como el número total de horas dedicadas.

Fase o tarea	Horas estimadas	Perfil desarrollador	Coste total
Análisis de requisitos y viabilidad	10	Analista	**500 €**
Análisis de infraestructuras y conceptos relacionados con la firma electrónica avanzada y política de firma	30	Analista	**1500 €**
Análisis completo de la estructura, reglas y restricciones de la política de firma	40	Analista	**2000 €**
Análisis de las capas presentes en el diseño	10	Analista	**500 €**
Diseño de la capa de acceso a los datos de la política de firma	20	Analista	**1000 €**
Programación de la capa de acceso a datos de la política de firma	20	Programador	**600 €**
Diseño de la capa de acceso a datos de la firma electrónica avanzada	10	Analista	**500 €**
Diseño de la capa contenedora del informe de verificación	10	Analista	**500 €**
Diseño del conjunto de verificadores	60	Analista	**3000 €**
Programación del conjunto de verificadores	120	Programador	**3600 €**
Diseño de las clases de test para la capa de verificación	10	Analista	**500 €**
Programación de las clases de test de la capa de verificación	30	Programador	**900€**
Elaboración de la documentación	60	Analista	**3000 €**
Elaboración de la documentación	60	Programador	**1800 €**
Total	**490**		**19900 €**

7. Conclusiones

En este proyecto he aprendido a afrontar el diseño completo de un sistema software complejo, desde la búsqueda de información y puesta al día hasta el diseño y la implementación del sistema.

Desde el planteamiento inicial, me he visto obligado a realizar una búsqueda exhaustiva de información para obtener una buena base en los campos de firma electrónica e infraestructura de clave pública con la finalidad de poder entender y comprender en profundidad el problema planteado para afrontarlo con garantías. Luego he tenido que plantear posibles soluciones para cada uno de los requisitos y, en varias ocasiones, estas soluciones han ido cambiando, evolucionando y madurando a lo largo de la realización del proyecto al ver que no eran del todo satisfactorias o que se podían plantear alternativas mejores.

Además durante el proceso de codificación e implementación de las clases he ido generando en paralelo clases de test para ir comprobando el correcto funcionamiento e ir depurando posibles fallos a medida que se iba generando el código.

En definitiva, he aprendido a plantear, desarrollar y resolver un problema de diseño mediante el método iterativo de desarrollo de software.

Además he podido desarrollar y aprender una serie de conocimientos sobre un tema que hoy en día es de total actualidad: la firma electrónica avanzada y política de firma tienen un gran futuro en las comunicaciones, el papeleo electrónico y en las tecnologías de la información en general. En los últimos años su utilización ha crecido exponencialmente y se espera un mayor crecimiento en el futuro gracias a la expansión masiva de algunos de sus usos, como el comercio electrónico, o la incorporación de nuevos usos como el documento nacional de identidad electrónico en nuestro país.

8. Futuro del proyecto

Como ya se ha comentado en el planteamiento, este proyecto no termina aquí, sino que continuará hasta cumplir todas las especificaciones iniciales, para integrarse con un módulo de generación y verificación de firma electrónica avanzada y formar así una herramienta mucho más completa.

A partir de aquí, los pasos a seguir para llegar a cumplir todos los objetivos marcados del proyecto pasan por ir cubriendo, poco a poco las carencias definidas en el apartado 5.2. Luego, una vez alcanzados los objetivos iniciales se podrían plantear algunas mejoras.

Se pueden clasificar las tareas pendientes según si son de integración con el otro módulo o no. Las tareas de integración, requieren un desarrollo conjunto con el equipo encargado del otro módulo, mientras que las tareas que no lo son podrían ir desarrollándose independientemente.

- Tareas de integración:
  - Implementar la capa de acceso a la firma avanzada y a los datos de verificación.
  - Implementación del instructor para dar las instrucciones de generación y verificación necesarias al módulo de firma.
- Tareas Independientes:
  - Implementación de la capa de acceso a los datos de la política de firma en diferentes formatos (XML y ASN.1).
  - Implementación de diferentes formateadores para el informe de verificación, como por ejemplo los indicados en el apartado 4.2.4.
  - Completar el conjunto de clases de test para comprobar en cualquier momento el correcto funcionamiento de todos los verificadores.

Más allá de estas tareas aún por completar podríamos pensar en algunas posibles mejoras al módulo de verificación de política de firma. Algunas de estas mejoras podrían ser la internacionalización del módulo, pudiendo generar informes en cualquier idioma, añadirle utilidades para la generación de documentos de política de firma, etc.

9. Referencias bibliográficas

- RFC-1779: String Representation of Distinguished Names
- RFC-2253: UTF-8 String Representation of Distinguished Names
- RFC-3852, RFC-3369, RFC-2630: Cryptographic Message Syntax (CMS)
- ETSI TS 101 903: XML Advanced Electronic Signatures (XAdES)
- ETSI TS 101 733: Electronic Signature and Infrastructures (ESI); Electronic Signature Formats
- ETSI TR 102 272: Electronic signaturas and Infrastructures (ESI); ASN.1 format for signature policies
- ETSI TR 102 038: Electronic signaturas and Infrastructures (ESI); XML format for signature policies
- W3C Recommendation part 1: "XML Schema Part 1: Structures"
- W3C Recommendation part 2: "XML Schema Part 2: Datatypes"
- W3C/IETF Recommendation: "XML-Signature Syntax and Processing"
- DIRECTIVA 1999/93/CE DEL PARLAMENTO EUROPEO Y DEL CONSEJO de 13 de diciembre de 1999: marco comunitario para la firma electrónica
- LEY 59/2003, de 19 de diciembre, de firma electrónica
- "Thinking in Patterns", Bruce Eckel: libro sobre patrones de software en Java.
- "A student guide to object-oriented development", Carol Britton – Jill Doake: libro sobre modelos UML aplicados a Java
- Wikipedia: www.wikipedia.org