CAPÍTULO 10
INSTRUMENTOS Y MÉTODOS DE LA INGENIERÍA DE SOFTWARE

ACRÓNIMOS

CASE	Computer Assisted Software Engineering

INTRODUCCIÓN

Los instrumentos de desarrollo de software son los instrumentos asistidos por ordenador que son requeridos para ayudar a los procesos de ciclo de vida de software. Los instrumentos permiten a acciones repetidas, bien definidas para ser automatizadas, reduciendo la carga cognoscitiva sobre el ingeniero de software que es entonces libre de concentrarse en los aspectos creativos del proceso. Los instrumentos a menudo son diseñados para apoyar el software particular métodos de la ingeniería, reduciendo cualquier carga administrativa asociada con la aplicación del método a mano. Como los métodos de la ingeniería de software, ellos son queridos para hacer el software que trama más sistemático, varían en el alcance de apoyar tareas individuales que abarcan el ciclo de vida completo.

Los métodos de la ingeniería de software imponen la estructura a la actividad de la ingeniería de software con el objetivo de hacer la actividad sistemática y en última instancia más probablemente de ser acertado. Los métodos por lo general proporcionan la notación y el vocabulario, procedimientos para realizar tareas identificables, y directrices para comprobar tanto el proceso como el producto. Ellos varían extensamente en el alcance, de una fase única del ciclo de vida al ciclo de vida completo. El énfasis en esta Área de Conocimiento está sobre los métodos de la ingeniería de software que abarcan múltiples fases de ciclo de vida, ya que métodos específicos de fase son cubiertos por otras áreas de conocimiento.

Mientras hay manuales detallados sobre instrumentos específicos y numerosos papeles de investigación sobre instrumentos innovadores, escrituras genéricas técnicas sobre instrumentos de la ingeniería de software son relativamente escasas. Una dificultad es la alta tarifa de cambio de instrumentos de software en general. Detalles específicos cambian con regularidad, haciendo difícil de proporcionar ejemplos concretos y actualizados.

Los Instrumentos de Ingeniería de Software y los Métodos del Área de Conocimiento cubren los procesos de ciclo de vida completos, y por lo tanto son relacionados con cada área de conocimiento en la Guía.

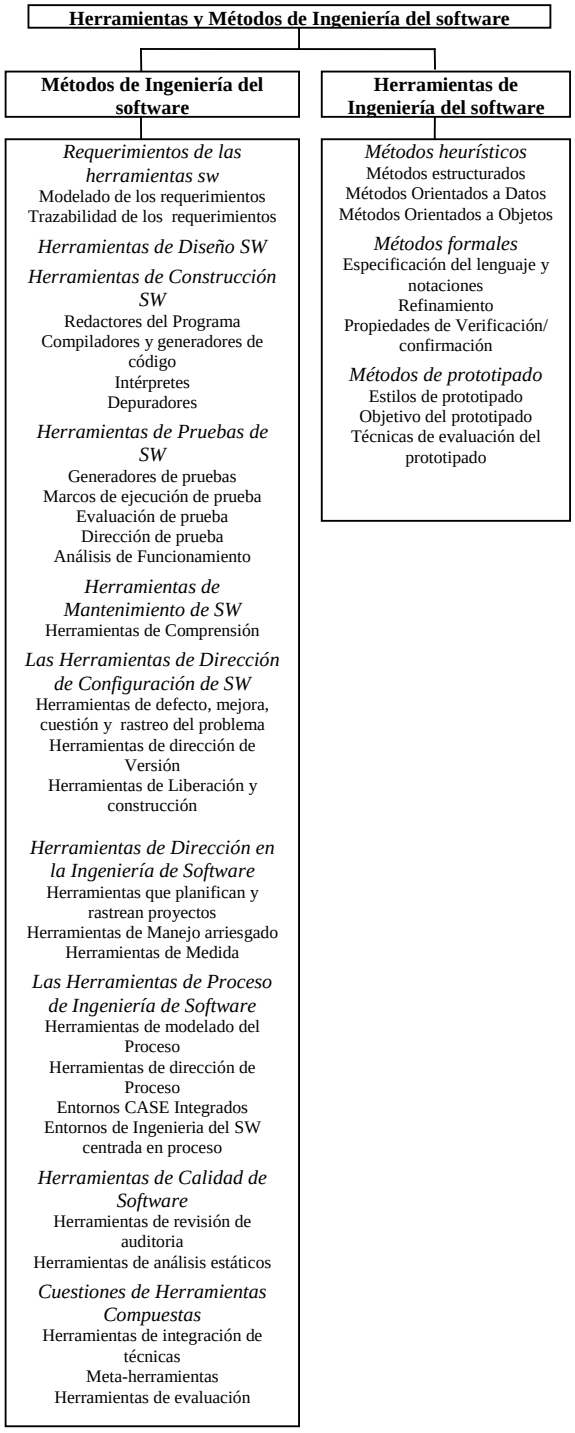

Figura 1 Desglose de tópicos de Instrumentos de Ingeniería del software y los Métodos del Área de Conocimiento.

ESTUDIO DE LAS HERRAMIENTAS Y MÉTODOS DE LA INGENIERÍA DE SOFTWARE

1. Las Herramientas de Ingeniería de Software

Los cinco primeros asuntos del subárea de los Instrumentos de Ingeniería de Software corresponden a las cinco primeras áreas del conocimiento de la Guía (Exigencias de Software, el Diseño de Software, la Construcción de Software, Pruebas de Software, y el Mantenimiento de Software). Los cuatro siguientes asuntos corresponden a las áreas de conocimiento restantes (la Dirección de Configuración de Software, la Dirección de la Ingeniería de Software, el Proceso de Ingeniería de Software, y la Calidad de Software). Proporcionan un asunto adicional, dirigiendo áreas como las técnicas de integración de instrumento que son potencialmente aplicables a todas las clases de instrumentos

1.1 *Las herramientas de Exigencias de Software* [Dor97, Dor02]

Los instrumentos para tratar con exigencias de software han sido clasificados en dos categorías: modelado e instrumentos de capacidad de rastreo.

- Exigencias de los Instrumentos de modelado. Estos instrumentos son usados para la obtención, el análisis, la especificación, y validez de las exigencias de software.
- Exigencias de los Instrumentos de capacidad de rastreo. [Dor02] Estos instrumentos se hacen cada vez más importante debido a que la complejidad de software crece. Ya que ellos son también relevantes en otros procesos de ciclo de vida, son presentados separadamente de los instrumentos de modelado.

1.2. Las herramientas Diseño de Software [Dor02]

Este asunto cubre instrumentos para crear y comprobar diseños de software. Hay una variedad de tales instrumentos, con la mayor parte de esta variedad siendo una consecuencia de la diversidad de notaciones de diseño de software y métodos. A pesar de esta variedad, ninguna división convincente para este asunto ha sido encontrada.

1.3. Las Herramientas de Construcción de Software [Dor02, Rei96]

Este asunto cubre instrumentos de construcción de software. Estos instrumentos son usados para producir y traducir la representación de programa (por ejemplo, el código original) que suficientemente es detallado y explícito para permitir la ejecución de máquina.

- Redactores del Programa. Estos instrumentos son usados para la creación y la modificación de programas, y posiblemente los documentos asociados con ellos. Pueden ser el texto de uso general o redactores de documento, o pueden ser especializado para un idioma de llegada.
- Compiladores y generadores de código. Tradicionalmente, los compiladores han sido los traductores no interactivos de código original, pero hubo una tendencia para integrar compiladores y redactores de programa para proporcionar ambientes de programa integrados. Este asunto también cubre preprocesadores, enlazadores/cargadores, y generadores de código.
- Intérpretes. Estos instrumentos proporcionan la ejecución de software por la emulación. Pueden apoyar actividades de construcción de software proporcionando un ambiente más controlable y observable para la ejecución de programa.
- Depuradores. Estos instrumentos son considerados en una categoría separada ya que ellos apoyan el proceso de construcción de software, pero son diferentes de redactores de programa y recopiladores.

1.4. Herramientas de Pruebas de Software [Dor02, Pfl01, Rei96]

- Generadores de pruebas. Estos instrumentos ayudan en el desarrollo de casos de prueba.
- Marcos de ejecución de prueba. Estos instrumentos permiten la ejecución de casos de prueba en un ambiente controlado donde el comportamiento del objeto bajo prueba es observado.
- Herramientas de evaluación de prueba. Estos instrumentos apoyan la evaluación de los resultados de ejecución de prueba, ayudando a determinar si realmente el comportamiento observado se conforma al comportamiento esperado.
- Herramientas de dirección de prueba. Estos instrumentos proporcionan el apoyo a todos los aspectos del proceso de pruebas de software.
- Herramientas de análisis de Funcionamiento. [Rei96] Estos instrumentos son usado para medir y analizar el funcionamiento de software, que es una forma especializada de pruebas donde el objetivo es de evaluar el comportamiento de funcionamiento más bien que el comportamiento funcional (la corrección).

1.5. Herramientas de Mantenimiento de Software [Dor02, Pfl01]

Este asunto abarca los instrumentos que son en particular importantes en el mantenimiento de software donde el software existente está siendo modificado. Dos categorías son identificadas: instrumentos de comprensión e instrumentos de reingeniería.

- Herramientas de Comprensión. [Re196] Estos instrumentos ayudan en la comprensión humana de programas. Los ejemplos incluyen instrumentos de visualización como rebanadores de programa y animadores.
- Herramientas de reingeniería. En el Mantenimiento de las áreas de conocimiento de Software, reingeniería es definido como el examen y la alteración del software sustancial para reconstituirlo en una nueva forma, e incluye la puesta en práctica subsiguiente de la nueva forma. Los instrumentos de reingeniería apoyan aquella actividad.

Al revés herramientas de la ingeniería ayudan al proceso trabajando hacia atrás de un producto existente a crear artefactos como la especificación y descripciones de diseño, que entonces pueden ser transformadas para generar un nuevo producto de uno anterior.

1.6. Las herramientas de Dirección de Configuración de Software

[Dor02, Rei96, Som05]

Las herramientas para la dirección de configuración han sido divididos en tres categorías: rastreo, dirección de versión, e instrumentos de liberación.

- Defecto, mejora, cuestión, e instrumentos que rastrean problema. Estos instrumentos son usados en la conexión con las cuestiones que rastrean problema asociadas con un producto de software particular.
- Herramientas de dirección de Versión. Estos instrumentos están implicados en la dirección de múltiples versiones de un producto.
- Herramientas de liberación y construcción. Estos instrumentos son usados para las tareas de liberación y construcción de software. La categoría incluye los instrumentos de instalación que se han hecho extensamente usados para configurar la instalación de productos de software.

Más información adicional en Software Configuration Management KA, topic 1.3 *Planning for SCM*.

1.7. Herramientas de Dirección en la Ingeniería de Software

[Dor02]

herramientas de Dirección en la Ingeniería de Software esta subdividido en tres categorías: planificación de proyecto y rastreo, manejo arriesgado, y medida.

- Herramientas que planifican y rastrean proyectos. Estos instrumentos son usados en la medida de esfuerzo de proyecto de software y cuentan la valoración, así como la planificación de proyecto.
- Herramientas de Manejo arriesgado. Estos instrumentos son usados en la identificación, la estimación, y riesgos de supervisión.
- Herramientas de Medida. Los instrumentos de medida asisten en la realización de las actividades relacionadas con el programa de medida de software.

1.8. Las Herramientas de Proceso de Ingeniería de Software

[Dor02, Som05]

Las herramientas de proceso de ingeniería de Software están divididos en instrumentos que modelan, instrumentos de dirección, y ambientes de desarrollo de software.

- Herramientas de modelado del Proceso. [Pfl01] Estos instrumentos son usados para modelar e investigar los procesos de la ingeniería de software.
- Herramientas de dirección de Proceso. Estos instrumentos proporcionan el apoyo a la dirección de la ingeniería de software.
- Entornos CASE Integrados. [Rei96, Som05] (ECMA55-93, ECMA69-94, IEEE1209-92, IEEE1348-95, Mul96) el software Integrado automatiza instrumentos de la ingeniería o ambientes que cubren múltiples fases del software el ciclo de vida de la ingeniería pertenece a este subtema. Tales instrumentos realizan múltiples funciones y de ahí potencialmente actúan recíprocamente con el proceso de ciclo de vida de software siendo ejecutado.
- Entornos de Ingeniería del SW centrada en proceso. [Rei96] (Gar96) Estos ambientes explícitamente incorporan la información sobre los procesos de ciclo de vida de software y dirigen y supervisan al usuario según el proceso definido.

1.9. Las Herramientas de Calidad de Software

[Dor02]

Las herramientas de Calidad son divididas en dos categorías: inspección e instrumentos de análisis.

- Herramientas de revisión de auditoria. Estos instrumentos son usados para apoyar revisiones y revisiones de cuentas.
- Herramientas de análisis estáticos. [Cla96, Pfl01, Rei96] Estos instrumentos son usados para analizar artefactos de software, como analizadores sintácticos y semánticos, así como

datos, el flujo de control, y analizadores de dependencia. Tales instrumentos son queridos para comprobar artefactos de software para la conformidad o para verificar propiedades deseadas.

1.10. Cuestiones de Instrumento Compuestas
[Dor02]

Este asunto cubre el tema aplicable a todas las clases de instrumentos. Tres categorías han sido identificadas: técnicas de integración de instrumento, meta-instrumentos, y evaluación de instrumento.

- Herramientas de integración de técnicas [Pfl01, Rei96, Som01] (Bro94) la integración de Instrumento es importante para hacer a instrumentos individuales cooperar. Esta categoría potencialmente se solapa con la categoría de ambientes de CASO integrada donde las técnicas de integración son aplicadas; sin embargo, es suficientemente distinto para merecer una categoría de su propiedad. Las clases típicas de integración de instrumento son la plataforma, la presentación, el proceso, datos, y el control.
- Meta-herramientas. Los Meta-instrumentos generan otros instrumentos; recopilador de recopiladores son el ejemplo clásico.
- Herramientas de evaluación. [Pfl01] (IEEE1209-92, IEEE1348-95, Mos92, Val97) A causa de la evolución continua de los instrumentos de la ingeniería de software, la evaluación de instrumento son un tema esencial.

2. Los Métodos de la Ingeniería de Software

Los Métodos de la Ingeniería de Software están dividido en tres temas: métodos heurísticos que tratan con accesos informales, métodos formales que tratan con accesos matemáticamente basados, y métodos de prototipado que tratan con software que trama accesos basados en varias formas de prototipado. Estos tres temas no son inconexos; más bien representan preocupaciones distintas. Por ejemplo, un método orientado por objeto puede incorporar técnicas formales y confiar en prototipado para la verificación y la validación. Como los instrumentos de la Ingeniería de Software, las metodologías continuamente se desarrollan. Por consiguiente, la descripción del área de conocimiento evita en la medida de lo posible llamar metodologías particulares.

2.1. Métodos heurísticos
[Was96]

Este tema contienen cuatro categorías: estructurado, orientado a datos, orientado a objetos, y específico de dominio. La categoría específica de dominio incluye métodos especializados para desarrollar los sistemas que implican en tiempo real, de seguridad, o aspectos de seguridad.

- Métodos Estructurados. [Dor02, Pfl01, Pre04, Som05] el sistema es construido de un punto de vista funcional, que comienza con una vista de alto nivel y cada vez más la refinación de esto en un diseño más detallado.
- Métodos Orientados a datos. [Dor02, Pre04] Aquí, los puntos de partida son las estructuras de datos que un programa manipula más que la función que esto realiza.
- Métodos Orientados a objetos. [Dor02, Pfl01, Pre04, Som05] el sistema es visto como una colección de objetos más que de funciones.

2.2. Métodos Formales
[Dor02, Pre04, Som05]

Esta subdivisión trata con el software matemáticamente basado métodos de la ingeniería, y es subdividida según varios aspectos de métodos formales.

- Especificación del lenguaje y notaciones. [Cla96, Pfl01, Pre01] Este tema concierne la notación de especificación o la lengua usada. Las lenguas de especificación pueden ser clasificadas como orientado por modelo, orientado por característica, u orientado por comportamiento.
- Refinamiento. [Pre04] Este tema trata como el método refina (o transforma) la especificación en una forma que es más cercana a la forma deseada final de un programa ejecutable.
- Propiedades de Verificación/confirmación. [Cla96, Pfl01, Som05] Este tema cubre las propiedades de verificación que son específicas al acercamiento formal, incluyendo tanto confirmación de teorema como la comprobación del modelo.

2.3. Métodos de prototipado
[Pre04, Som05, Was96]

Esta subdivisión cubre métodos que implican el prototipazo de software y es subdividida en estilos de prototipado, objetivos, y técnicas de evaluación.

- Estilos de prototipado. [Dor02, Pfl01, Pre04] (Pom96) el tema de estilos de prototipado identifica varios accesos: especificación desechable, evolutiva, y ejecutable.
- Objetivo del prototipado. [Dor97] (Pom96) los Ejemplos de los objetivos de un método prototipado puede ser exigencias, el diseño arquitectónico, o el interfaz de usuario.
- Técnicas de evaluación del prototipado. Este tema cubre las razones por las cuales los resultados de un ejercicio de prototipo son usados.

MATRIZ DE TEMAS VS. REFERENCIAS

	[Cla96]	[Dor02] {Dor97}	[Pfl01] {PFL98}	[Pre04]	[Rei96]	[Som05]	[Was96]
1.Las Herramientas de Ingeniería de Software							
1.1Las Herramientas de Exigencias de Software		{c4s1} ,v2c8s4					
Exigencias de los Herramientas de modelado							
Exigencias de los Herramientas de capacidad de rastreo.		v1c4s2					
1.2 Los Herramientas de Diseño de Software		v2c8s4					
1.3. Los Herramientas de Construcción de Software		v2c8s4			c112s2		
Redactores del Programa							
Compiladores y generadores de código							
Intérpretes.							
Depuradores							
1.4. Herramientas *de Pruebas de Software*		v2c8s4	C8s7,c9s7		c112s3		
Generadores de pruebas							
Marcos de ejecución de prueba							
Herramientas de evaluación de prueba							
Herramientas de dirección de prueba.							
Herramientas de análisis de Funcionamiento					c112s5		
1.5. Herramientas *de Mantenimiento de Software*		v2c8s4	c11s5				
Herramientas de Comprensión					c112s5		
Herramientas de reingeniería							
1.6.Las Herramientas *de Dirección de Configuración de Software*		v2c8s4	c11s5		c112s3	c29	
Herramientas de defecto, mejora, cuestión y rastreo del problema							
Herramientas de dirección de Versión							
Herramientas de Liberación y construcción							

	[Cla96]	[Dor02]{Dor97}	[Pfl01]{PFL98}	[Pre04]	[Rei96]	[Som05]	[Was96]
1.7. Herramientas de Dirección en la Ingeniería de Software		v2c8s4					
Herramientas que planifican y rastrean proyectos							
Herramientas de Manejo arriesgado							
Herramientas de Medida							
1.8. Las Herramientas de Proceso de Ingeniería de Software		v2c8s4					
Herramientas de modelado del Proceso			c2s3, 2s4				
Herramientas de dirección de Proceso							
Entornos CASE Integrados					c112s3, c112s4	c3	
Entornos de Ingenieria del SW centrada en proceso					c112s5		
1.9. Las Herramientas de Calidad de Software		v2c8s4					
Herramientas de revisión de auditoria							
Herramientas de análisis estáticos	*		C8s7		c112s5		
1.10. Cuestiones de Herramientas Compuestas		v2c8s4					
Herramientas de integración de técnicas			c1s8		c112s4		*
Meta-herramientas							
Herramientas de evaluación			C9s10				
2. Los Métodos de la Ingeniería de Software							
2.1. Métodos heurísticos							*
Métodos Estructurados		v1c5s1, v1c6s3	c4s5	c7-c9		c15	
Métodos Orientados a datos		v1c5s1, v1c6s3		c7-c9			
Métodos Orientados a objetos		v1c6s2, v1c6s3	c4s4, c6, c8s5	c7-c9		c12	
2.2. Métodos Formales		v1c6s5		c28		c9	
Especificación del lenguaje y notaciones	*		c4s5				
Refinamiento							
Propiedades de Verificación/ confirmación	*		c5s7, c8s3				
2.3. Métodos de prototipado						c8	*
Estilos de prototipado		v1c4s4	c4s6, c5s6				
Objetivo del prototipado		v1c4s4					
Técnicas de evaluación del prototipado							

REFERENCIAS RECOMENDADAS PARA HERRAMIENTAS Y MÉTODOS DE INGENIERIA DEL SOFTWARE

[Cla96] E.M. Clarke et al., "Formal Methods: State of the Art and Future Directions," *ACM Computer Surveys*, vol. 28, iss. 4, 1996, pp. 626-643.

[Dor97] M. Christensen, M. Dorfman and R.H. Thayer, eds., *Software Engineering*, IEEE Computer Society Press, 1997.

[Dor02] M. Christensen, M. Dorfman and R.H. Thayer, eds., *Software Engineering*, Vol. 1 & Vol. 2, IEEE Computer Society Press, 2002.

[Pfl01] S.L. Pfleeger, *Software Engineering: Theory and Practice*, second ed., Prentice Hall, 2001.

[Pre04] R.S. Pressman, *Software Engineering: A Practitioner's Approach*, sixth ed., McGraw-Hill, 2004.

[Rei96] S.P. Reiss, *Software Tools and Environments in The Computer Science and Engineering Handbook*, CRC Press, 1996.

[Som05] I. Sommerville, *Software Engineering*, seventh ed., Addison-Wesley, 2005.

[Was96] A.I. Wasserman, "Toward a Discipline of Software Engineering," *IEEE Software*, vol. 13, iss. 6, November 1996, pp. 23-31.

APÉNDICE A. LISTA DE LECTURAS COMPLEMENTARIAS

(Ber93) E.V. Berard, *Essays on Object-Oriented Software Engineering*, Prentice Hall, 1993.

(Bis92) W. Bischofberger and G. Pomberger, *Prototyping-Oriented Software Development: Concepts and Tools*, Springer-Verlag, 1992.

(Bro94) A.W. Brown et al., *Principles of CASE Tool Integration*, Oxford University Press, 1994.

(Car95) D.J. Carney and A.W. Brown, "On the Necessary Conditions for the Composition of Integrated Software Engineering Environments," presented at Advances in Computers, 1995.

(Col94) D. Coleman et al., *Object-Oriented Development: The Fusion Method*, Prentice Hall, 1994.

(Cra95) D. Craigen, S. Gerhart, and T. Ralston, "Formal Methods Reality Check: Industrial Usage," *IEEE Transactions on Software Engineering*, vol. 21, iss. 2, February 1995, pp. 90-98.

(Fin00) A. Finkelstein, ed., *The Future of Software Engineering*, ACM, 2000.

(Gar96) P.K. Garg and M. Jazayeri, *Process-Centered Software Engineering Environments*, IEEE Computer Society Press, 1996.

(Har00) W. Harrison, H. Ossher, and P. Tarr, "Software Engineering Tools and Environments: A Roadmap," 2000.

(Jar98) S. Jarzabek and R. Huang, "The Case for User-Centered CASE Tools," *Communications of the ACM*, vol. 41, iss. 8, August 1998, pp. 93-99.

(Kit95) B. Kitchenham, L. Pickard, and S.L. Pfleeger, "Case Studies for Method and Tool Evaluation," *IEEE Software*, vol. 12, iss. 4, July 1995, pp. 52-62.

(Lam00) A. v. Lamsweerde, "Formal Specification: A Roadmap," *The Future of Software Engineering*, A. Finkelstein, ed., ACM, 2000, pp. 149-159.

(Mey97) B. Meyer, *Object-Oriented Software Construction*, second ed., Prentice Hall, 1997.

(Moo98) J.W. Moore, *Software Engineering Standards, A User's Roadmap*, IEEE Computer Society Press, 1998.

(Mos92) V. Mosley, "How to Assess Tools Efficiently and Quantitatively," *IEEE Software*, vol. 9, iss. 3, May 1992, pp. 29-32.

(Mül96) H.A. Muller, R.J. Norman, and J. Slonim, eds., "Computer Aided Software Engineering," special issue of *Automated Software Engineering*, vol. 3, iss. 3/4, Kluwer, 1996.

(Mül00) H. Müller et al., "Reverse Engineering: A Roadmap," *The Future of Software Engineering*, A. Finkelstein, ed., ACM, 2000, pp. 49-60.

(Pom96) G. Pomberger and G. Blaschek, *Object-Orientation and Prototyping in Software Engineering*: Prentice Hall, 1996.

(Pos96) R.M. Poston, *Automating Specification-based Software Testing*, IEEE Press, 1996.

(Ric92) C. Rich and R.C. Waters, "Knowledge Intensive Software Engineering Tools," *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, iss. 5, October 1992, pp. 424-430.

(Son92) X. Song and L.J. Osterweil, "Towards Objective, Systematic Design-Method Comparisons," *IEEE Software*, vol. 9, iss. 3, May 1992, pp. 43-53.

(Tuc96) A.B. Tucker, *The Computer Science and Engineering Handbook*, CRC Press, 1996.

(Val97) L.A. Valaer and R.C.B. II, "Choosing a User Interface Development Tool," *IEEE Software*, vol. 14, iss. 4, 1997, pp. 29-39.

(Vin90) W.G. Vincenti, *What Engineers Know and How They Know It — Analytical Studies from Aeronautical History*, John Hopkins University Press, 1990.

(Wie98) R. Wieringa, "A Survey of Structured and Object-Oriented Software Specification Methods and Techniques," *ACM Computing Surveys*, vol. 30, iss. 4, 1998, pp. 459-527.

APÉNDICE B. LISTA DE ESTANDARS

(ECMA55-93) ECMA, *TR/55 Reference Model for Frameworks of Software Engineering Environments*, third ed., 1993.

(ECMA69-94) ECMA, *TR/69 Reference Model for Project Support Environments*, 1994.

(IEEE1175.1-02) IEEE Std 1175.1-2002, *IEEE Guide for CASE Tool Interconnections—Classification and Description*, IEEE Press, 2002.

(IEEE1209-92) IEEE Std 1209-1992, *Recommended Practice for the Evaluation and Selection of CASE Tools, (ISO/IEC 14102, 1995)*, IEEE Press, 1992.

(IEEE1348-95) IEEE Std 1348-1995, *Recommended Practice for the Adoption of CASE Tools, (ISO/IEC 14471)*, IEEE Press, 1995.

(IEEE12207.0-96) IEEE/EIA 12207.0-1996//ISO/IEC12207:1995, *Industry Implementation of Int. Std. ISO/IEC 12207:95, Standard for Information Technology—Software Life Cycle Processes*, IEEE Press, 1996.