

Este protocolo hace referencia a los elementos que el IUED debe valorar acerca de la **adecuación a la metodología a distancia de los Libros de texto didácticos** de las asignaturas de Grado y Máster de la UNED.

## PROTOCOLO DE REVISIÓN DE LIBROS DE TEXTO DIDÁCTICOS

PROTOCOLO DE REVISIÓN DE LIBROS DE TEXTO DIDÁCTICOS

IDENTIFICACIÓN DEL LIBRO DE TEXTO											
Título de Grado: Ingeniería Informática											
Facultad / Escuela: E. T. S. de Ingeniería Informática											
Departamento:											
Asignatura: Procesadores de lenguaje II									Código: 71013118		
Curso				Carácter			Duración			ECTS	
1º	2º	3º X	4º	FB	Ob X	Op	Anual	Sem 1º	Sem 2º X	6	
Equipo Docente: Anselmo Peñas y Álvaro Rodrigo											
Título del libro: Compiladores. Principios, técnicas y herramientas											
Autor(es): Alfred V. Aho, Monica S. Lam, Ravi Sethi y Jeffrey D. Ullman											
Editorial: Pearson - Addison Wesley									Año de publicación: 2008		

ESCALA DE VALORACIÓN				
(NF) <u>No figura</u>	(1) <u>Insuficiente</u>	(2) <u>Suficiente</u>	(3) <u>Bien</u>	(4) <u>Muy bien</u>
(añádase el comentario que corresponda en la columna <b>Observaciones</b> , si procede)				

1. PRESENTACIÓN DEL LIBRO						
	NF	1	2	3	4	Observaciones
Introducción general del libro de texto					X	Excelente
Contextualización de sus contenidos					X	
Justificación de la relevancia de sus contenidos					X	
Presentación del(los) autor(es)					X	En prefacio y al final del libro
Referencia a los conocimientos previos necesarios					X	
2. ESTRUCTURA Y OBJETIVOS						
	NF	1	2	3	4	Observaciones
Estructura general de sus contenidos clara y ordenada (orienta y sitúa al estudiante)					X	Amplia explicación en el Prefacio
División de sus contenidos en capítulos, apartados, etc. (conforme a su índice)					X	
Exposición de epígrafes con orden de nivel					X	
Objetivos generales concretos y bien expresados	X					Se recomienda incluir en guía de estudio
Objetivos por tema o bloque temático concretos y bien expresados			X			Mínimamente dentro de la introducción de algunos temas. Recomendamos tratar en epígrafe aparte o incorporar en guía de estudio.
3. ORIENTACIONES PARA EL APRENDIZAJE						
<i>(el texto puede incluir orientaciones generales para su estudio, pudiendo ofrecerse otras más detalladas y específicas para el estudiante de la asignatura en la Guía de estudio. La valoración de este apartado puede por tanto complementarse con la información incluida en la Guía de estudio)</i>						
	NF	1	2	3	4	Observaciones
Explica con claridad cómo abordar el estudio de sus contenidos					X	En el prefacio "Cómo utilizar este libro"

# MADI Protocolo de revisión de Libros de texto didácticos



Orienta sobre cómo superar las posibles dificultades que puede encontrar el estudiante en su proceso de aprendizaje	X					Recomendamos tratar en guía de estudio
Proporciona pautas de estudio para cada tema o bloque temático	X					Recomendamos tratar en guía de estudio
Prioriza los aspectos más relevantes del texto, de los temas o bloques temáticos				X		Recomendamos utilizar más recursos gráficos, color (si no se imprime), sangrías con viñetas...

## 4. ELEMENTOS FACILITADORES DEL APRENDIZAJE

(pueden ampliarse o combinarse con lo ofrecido en la Guía de estudio, por lo que pueden valorarse conjuntamente)

	NF	1	2	3	4	Observaciones
Ofrece una presentación atractiva y motivadora			X			
Incluye resúmenes					X	
Dispone de glosario(s)					X	Índice onomástico al final
Emplea mapas conceptuales					X	
Posee cronologías						No procede
Da ejemplos					X	
Utiliza esquemas					X	
Introduce elementos de diálogo con el estudiante que le ayuden y animen a reflexionar					X	Se advierte que el libro se entrega con las tareas "Gradiance", tutoriales de ayuda al alumno en inglés.
Cuenta con ilustraciones, fotografías e imágenes alusivas	X					
Ofrece cuadros y tablas					X	
Hace uso de grafías y atributos de texto y formato que dirijan la atención						Cuadros con definiciones
Ofrece bibliografía recomendada, así como comentarios y actualizaciones					X	
Procura enlaces de interés					X	
Propone actividades complementarias que motiven y amplíen sus propios contenidos					X	Se habla de actividades on line, Gradiance con la compra del libro
Otros (especifíquense)						

## 5. ELEMENTOS DE AUTOEVALUACIÓN

(pueden ampliarse o combinarse con lo ofrecido en la Guía de estudio, por lo que pueden valorarse conjuntamente)

	NF	1	2	3	4	Observaciones
En qué medida se incluyen procedimientos de autoevaluación					X	Preguntas de respuesta corta en cada apartado y on line (Gradiance)
Las autoevaluaciones son coherentes con los contenidos						Carecemos de criterio para evaluar
Las autoevaluaciones son coherentes con las actividades propuestas						Carecemos de criterio para evaluar
En qué medida las autoevaluaciones proporcionan realimentación						Carecemos de criterio para evaluar

## 6. REDACCIÓN

(1) No adecuado      (2) Adecuado (recom.)      (3) Bien      (4) Excelente

(añádase el comentario que corresponda en la columna **Observaciones**, si procede)

	1	2	3	4	Observaciones
Claridad de la redacción				X	
Extensión adecuada de los párrafos				X	
Exposición ordenada de ideas		X			A veces, interrupciones del discurso sin marcas de puntuación. Por ej. pág. 506, segundo párrafo: el párrafo referido al capítulo 10 es un inciso que debería estar entre paréntesis o guiones.
Formulación correcta de citas y referencias bibliográficas					No figuran
Ortografía y signos de puntuación (acentuación, puntuación, siglas, símbolos, abreviaturas, mayúsculas, cursiva, etc.)				X	Algunos errores ortográficos, sobre todo de puntuación.

La valoración final del material de una asignatura contemplará la valoración del libro de texto y de su Guía de estudio. De esta manera, se garantizará que el conjunto reúne todos los elementos de orientación necesarios.

En ese sentido, los revisores del libro de texto deben referir qué elementos pueden no estar suficientemente desarrollados en el texto, para que se preste especial atención a los mismos en la Guía de estudio.

VALORACIÓN GENERAL DEL LIBRO DE TEXTO					
	1	2	3	4	Observaciones
Presentación del libro		X			-
Estructura y objetivos				X	-
Orientaciones para el aprendizaje				X	Autosuficiente GdE*
Elementos facilitadores del aprendizaje			X		Recomendamos complementar con guía. GdE*
Elementos de autoevaluación				X	GdE*
Redacción				X	-
<i>Motivador y atractivo</i>		X			
<i>Favorece la autonomía del estudiante</i>			X		Recomendamos complementar con guía.
<i>Favorece el estudio activo</i>			X		Recomendamos complementar con guía.
<i>Adecuación a la metodología de la enseñanza a distancia</i>			X		Recomendamos complementar con guía.

**\*GdE:** Con estas siglas se recuerda que los apartados señalados guardan una estrecha relación con la información ofrecida en la Guía de estudio, por lo que su valoración puede complementarse con la de su correspondiente Guía de estudio.

En el caso de algunos textos elaborados por autores externos a la UNED, o en el caso de textos que, por las características de la asignatura, no respondan al estilo de manual, será necesario hacer constar explícitamente que la guía deberá recoger determinados aspectos no presentes, o proporcionar mayor información.

### VALORACIÓN GLOBAL

<b>Excelente</b>	
<b>Adecuado y sin modificaciones</b>	
<b>Adecuado. Requiere complementar con la Guía de Estudio</b>	
<b>Con propuestas de mejora</b>	<b>X</b>
<b>Con modificaciones necesarias</b>	

#### **Comentario final**

*(añádase tantas líneas como sean necesarias)*

El material entregado es un PDF de más de mil páginas, que solo se ha podido ver en pantalla.

En general las herramientas didácticas del libro son suficientes para valorar positivamente su adecuación a los requerimientos del EEES. En su prefacio se incluyen pautas y orientaciones de estudio, contiene ejercicios de autoevaluación en cada uno de sus apartados, e invita al estudiante a entrar en el tutorial que se incluye con la compra del libro, facilitando el trabajo autónomo.

No obstante, advertimos la dificultad de manejar el manual por su volumen, sin hipervínculos o separación para cada una de las dos asignaturas de Procesadores.

También recomendamos, independientemente de otros recursos electrónicos, aportar en la guía de estudio los objetivos de cada tema, la solución de los ejercicios, o incorporar otras fórmulas para la autoevaluación, aspectos sumamente importantes para el estudiante a distancia.

**Madrid, 25 de mayo de 2016**

**VºBº Comisión y fecha**

Este protocolo hace referencia a los elementos que el IUED debe valorar acerca de la **adecuación a la metodología a distancia de los Libros de texto didácticos** de las asignaturas de Grado y Máster de la UNED.

## PROTOCOLO DE REVISIÓN DE LIBROS DE TEXTO DIDÁCTICOS

PROTOCOLO DE REVISIÓN DE LIBROS DE TEXTO DIDÁCTICOS

IDENTIFICACIÓN DEL LIBRO DE TEXTO											
Título de Grado: Ingeniería Informática											
Facultad / Escuela: E. T. S. de Ingeniería Informática											
Departamento:											
Asignatura: Procesadores de lenguaje I									Código: 71013130		
Curso				Carácter			Duración			ECTS	
1º	2º	3º X	4º	FB	Ob X	Op	Anual	Sem 1º X	Sem 2º	6	
Equipo Docente: Anselmo Peñas y Álvaro Rodrigo											
Título del libro: Compiladores. Principios, técnicas y herramientas											
Autor(es): Alfred V. Aho, Monica S. Lam, Ravi Sethi y Jeffrey D. Ullman											
Editorial: Pearson - Addison Wesley									Año de publicación: 2008		

ESCALA DE VALORACIÓN				
(NF) <u>No figura</u>	(1) <u>Insuficiente</u>	(2) <u>Suficiente</u>	(3) <u>Bien</u>	(4) <u>Muy bien</u>
(añádase el comentario que corresponda en la columna <b>Observaciones</b> , si procede)				

1. PRESENTACIÓN DEL LIBRO						
	NF	1	2	3	4	Observaciones
Introducción general del libro de texto					X	Excelente
Contextualización de sus contenidos					X	
Justificación de la relevancia de sus contenidos					X	
Presentación del(los) autor(es)					X	En prefacio y al final del libro
Referencia a los conocimientos previos necesarios					X	
2. ESTRUCTURA Y OBJETIVOS						
	NF	1	2	3	4	Observaciones
Estructura general de sus contenidos clara y ordenada (orienta y sitúa al estudiante)					X	Amplia explicación en el Prefacio
División de sus contenidos en capítulos, apartados, etc. (conforme a su índice)					X	
Exposición de epígrafes con orden de nivel					X	
Objetivos generales concretos y bien expresados	X					Se recomienda incluir en guía de estudio
Objetivos por tema o bloque temático concretos y bien expresados			X			Mínimamente dentro de la introducción de algunos temas. Recomendamos tratar en epígrafe aparte o incorporar en guía de estudio.
3. ORIENTACIONES PARA EL APRENDIZAJE						
<i>(el texto puede incluir orientaciones generales para su estudio, pudiendo ofrecerse otras más detalladas y específicas para el estudiante de la asignatura en la Guía de estudio. La valoración de este apartado puede por tanto complementarse con la información incluida en la Guía de estudio)</i>						
	NF	1	2	3	4	Observaciones
Explica con claridad cómo abordar el estudio de sus contenidos					X	En el prefacio "Cómo utilizar este libro"

# MADI Protocolo de revisión de Libros de texto didácticos



Orienta sobre cómo superar las posibles dificultades que puede encontrar el estudiante en su proceso de aprendizaje	X					Recomendamos tratar en guía de estudio
Proporciona pautas de estudio para cada tema o bloque temático	X					Recomendamos tratar en guía de estudio
Prioriza los aspectos más relevantes del texto, de los temas o bloques temáticos				X		Recomendamos utilizar más recursos gráficos, color (si no se imprime), sangrías con viñetas...

## 4. ELEMENTOS FACILITADORES DEL APRENDIZAJE

(pueden ampliarse o combinarse con lo ofrecido en la Guía de estudio, por lo que pueden valorarse conjuntamente)

	NF	1	2	3	4	Observaciones
Ofrece una presentación atractiva y motivadora			X			
Incluye resúmenes					X	
Dispone de glosario(s)					X	Índice onomástico al final
Emplea mapas conceptuales					X	
Posee cronologías						No procede
Da ejemplos					X	
Utiliza esquemas					X	
Introduce elementos de diálogo con el estudiante que le ayuden y animen a reflexionar					X	Se advierte que el libro se entrega con las tareas "Gradiance", tutoriales de ayuda al alumno en inglés.
Cuenta con ilustraciones, fotografías e imágenes alusivas	X					
Ofrece cuadros y tablas					X	
Hace uso de grafías y atributos de texto y formato que dirijan la atención						Cuadros con definiciones
Ofrece bibliografía recomendada, así como comentarios y actualizaciones					X	
Procura enlaces de interés					X	
Propone actividades complementarias que motiven y amplíen sus propios contenidos					X	Se habla de actividades on line, Gradiance con la compra del libro
Otros (especifíquense)						

## 5. ELEMENTOS DE AUTOEVALUACIÓN

(pueden ampliarse o combinarse con lo ofrecido en la Guía de estudio, por lo que pueden valorarse conjuntamente)

	NF	1	2	3	4	Observaciones
En qué medida se incluyen procedimientos de autoevaluación					X	Preguntas de respuesta corta en cada apartado y on line (Gradiance)
Las autoevaluaciones son coherentes con los contenidos						Carecemos de criterio para evaluar
Las autoevaluaciones son coherentes con las actividades propuestas						Carecemos de criterio para evaluar
En qué medida las autoevaluaciones proporcionan realimentación						Carecemos de criterio para evaluar

## 6. REDACCIÓN

(1) No adecuado      (2) Adecuado (recom.)      (3) Bien      (4) Excelente

(añádase el comentario que corresponda en la columna **Observaciones**, si procede)

	1	2	3	4	Observaciones
Claridad de la redacción				X	
Extensión adecuada de los párrafos				X	
Exposición ordenada de ideas		X			A veces, interrupciones del discurso sin marcas de puntuación. Por ej. pág. 506, segundo párrafo: el párrafo referido al capítulo 10 es un inciso que debería estar entre paréntesis o guiones.
Formulación correcta de citas y referencias bibliográficas					No figuran
Ortografía y signos de puntuación (acentuación, puntuación, siglas, símbolos, abreviaturas, mayúsculas, cursiva, etc.)				X	Algunos errores ortográficos, sobre todo de puntuación.

La valoración final del material de una asignatura contemplará la valoración del libro de texto y de su Guía de estudio. De esta manera, se garantizará que el conjunto reúne todos los elementos de orientación necesarios.

En ese sentido, los revisores del libro de texto deben referir qué elementos pueden no estar suficientemente desarrollados en el texto, para que se preste especial atención a los mismos en la Guía de estudio.

VALORACIÓN GENERAL DEL LIBRO DE TEXTO					
	1	2	3	4	Observaciones
Presentación del libro		X			-
Estructura y objetivos				X	-
Orientaciones para el aprendizaje				X	Autosuficiente GdE*
Elementos facilitadores del aprendizaje			X		Recomendamos complementar con guía. GdE*
Elementos de autoevaluación				X	GdE*
Redacción				X	-
<i>Motivador y atractivo</i>		X			
<i>Favorece la autonomía del estudiante</i>			X		Recomendamos complementar con guía.
<i>Favorece el estudio activo</i>			X		Recomendamos complementar con guía.
<i>Adecuación a la metodología de la enseñanza a distancia</i>			X		Recomendamos complementar con guía.

**\*GdE:** Con estas siglas se recuerda que los apartados señalados guardan una estrecha relación con la información ofrecida en la Guía de estudio, por lo que su valoración puede complementarse con la de su correspondiente Guía de estudio.

En el caso de algunos textos elaborados por autores externos a la UNED, o en el caso de textos que, por las características de la asignatura, no respondan al estilo de manual, será necesario hacer constar explícitamente que la guía deberá recoger determinados aspectos no presentes, o proporcionar mayor información.

**VALORACIÓN GLOBAL**

<b>Excelente</b>	
<b>Adecuado y sin modificaciones</b>	
<b>Adecuado. Requiere complementar con la Guía de Estudio</b>	
<b>Con propuestas de mejora</b>	<b>X</b>
<b>Con modificaciones necesarias</b>	

**Comentario final**

*(añádase tantas líneas como sean necesarias)*

El material entregado es un PDF de más de mil páginas, que solo se ha podido ver en pantalla.

En general las herramientas didácticas del libro son suficientes para valorar positivamente su adecuación a los requerimientos del EEES. En su prefacio se incluyen pautas y orientaciones de estudio, contiene ejercicios de autoevaluación en cada uno de sus apartados, e invita al estudiante a entrar en el tutorial que se incluye con la compra del libro, facilitando el trabajo autónomo.

No obstante, advertimos la dificultad de manejar el manual por su volumen, sin hipervínculos o separación para cada una de las dos asignaturas de Procesadores.

También recomendamos, independientemente de otros recursos electrónicos, aportar en la guía de estudio los objetivos de cada tema, la solución de los ejercicios, o incorporar otras fórmulas para la autoevaluación, aspectos sumamente importantes para el estudiante a distancia.

**Madrid, 25 de mayo de 2016**

**VºBº Comisión y fecha**

**PROTOCOLO ORIENTATIVO DE VALORACIÓN  
POR PARTE DEL DEPARTAMENTO  
DEL TEXTO PROPUESTO  
COMO MATERIAL DIDÁCTICO BÁSICO**



<b>IDENTIFICACIÓN DEL LIBRO DE TEXTO</b>										
<b>TÍTULO DE GRADO</b> Grado en Ingeniería Informática										
<b>FACULTAD / ESCUELA</b> ETSI Informática										
<b>DEPARTAMENTO</b> Lenguajes y Sistemas Informáticos										
<b>MATERIA</b> Procesadores del Lenguaje										
<b>ASIGNATURA</b> Procesadores del Lenguaje (I y II)								Código 71013130, 71013118		
<b>Curso en que se imparte:</b>				<b>Carácter</b>			<b>Duración</b>			<b>ECTS</b> 6+6
1º	2º	3º	4º	FB	OBX	OPT	ANUAL	SEMESTRAL		
		X						1er.X	2do.X	
<b>EQUIPO DOCENTE</b> P. Lenguaje I: Anselmo Peñas y Álvaro Rodrigo; P. Lenguaje II: Anselmo Peñas, Laura Plaza y Álvaro Rodrigo										
<b>TÍTULO DEL TEXTO</b> Compiladores: principios, técnicas y herramientas										
<b>AUTOR/ES DEL TEXTO</b> Sethi, Ravi ; Lam, Monica S. ; Aho, Alfred V. ; Ullman, Jeffrey							<b>AÑO DE EDICIÓN</b>			
<b>EDITORIAL DEL TEXTO</b> Pearson Addison-Wesley										

<b>REFERENTES</b>	
<i>(Añádase el comentario que corresponda en la columna valoración y recomendaciones)</i>	
	<i>Valoración y, en su caso, recomendaciones</i>
Ajuste al programa (evitando lagunas formativas y solapamientos con otras asignaturas)	El texto se ajusta completamente al programa y sirve de libro de texto para las asignaturas Procesadores del Lenguaje I y Procesadores del Lenguaje II, que están conectadas.
Calidad técnica de los contenidos	Es un compendio exhaustivo y autocontenido de todos los aspectos de la asignatura, con una gran calidad en las explicaciones.
Adecuación al nivel de los estudios de Grado	El texto es adecuado para el nivel de estudios de grado en el que se utiliza (tercer curso)
Ajuste en la extensión a los ECTS de la asignatura	El libro está ajustado a los créditos del global de las asignaturas Procesadores del Lenguaje I y Procesadores del Lenguaje II
Otros aspectos a considerar (p.e. estado de desarrollo del texto presentado y valorado; necesidad de elementos complementarios como la información de la Guía de Estudios, etc.)	El anterior texto está descatalogado, lo que hace necesario buscar otro. El texto que se propone se utiliza como referencia en otras universidades, por lo que se considera muy adecuado.

Madrid, a 3 de Junio de 2016

Fdo.

Director/a del Departamento.

## PROTOCOLO DE VALORACIÓN DE LA ADECUACIÓN DE DE LOS LIBROS DE TEXTO DIDÁCTICOS

### IDENTIFICACIÓN DEL LIBRO DE TEXTO

<b>TÍTULO DE GRADO</b>											<b>INGENIERÍA INFORMÁTICA</b>							
<b>FACULTAD / ESCUELA</b>											<b>INFORMÁTICA</b>							
<b>MATERIA</b>											<b>LENGUAJES DE PROGRAMACIÓN</b>							
<b>DEPARTAMENTO</b>											<b>LENGUAJES Y SISTEMAS INFORMÁTICOS</b>							
<b>ASIGNATURA</b>											<b>PROCESADORES DEL LENGUAJE II</b>				<b>Código</b>		<b>71013118</b>	
<b>Curso en que se imparte:</b>				<b>Carácter</b>			<b>Duración</b>				<b>ECTS</b>							
<b>1º</b>	<b>2º</b>	<b>3º</b>	<b>4º</b>	<b>FB</b>	<b>OB</b>	<b>OPT</b>	<b>ANUAL</b>		<b>SEMESTRAL</b>		<b>6</b>							
		<b>X</b>			<b>X</b>				<b>1er.</b>	<b>2do.X</b>								
<b>EQUIPO DOCENTE</b>																		
Anselmo Peñas y Álvaro Rodrigo																		
<b>AUTOR/ES</b>																		
Alfred V. Aho, Monica S. Lam, Ravi Sethi y Jeffrey D. Ullman																		
<b>EDITORIAL</b>																		
Pearson-Addison Wesley (Título: Compiladores. Principios, técnicas y herramientas)																		

### REFERENTES

<i>(Añádase el comentario que corresponda en la columna valoración y recomendaciones)</i>	
	<i>Valoración y, en su caso, recomendaciones</i>
Ajuste en la extensión a los ECTS de la asignatura	El libro propuesto contiene más temas de los exigidos por el temario de la asignatura (según la memoria ANECA de verificación del plan de estudios). Sin embargo, restringiendo el estudio a los temas que conforman el temario, la extensión se ajustaría al número de créditos de la asignatura.
Adecuación al nivel de conocimientos previos de los alumnos	El estudio de esta asignatura requiere conocimientos de varias asignaturas: "Fundamentos de Programación", "Programación Orientada a Objetos", "Estrategias de Programación y Estructuras de Datos", "Programación y Estructuras de Datos Avanzadas", "Autómatas, Gramáticas y Lenguajes" y "Procesadores del Lenguaje I"
Adecuación a los contenidos de la asignatura, sus características y su lugar en el plan de estudios	El libro presenta adecuadamente los contenidos de la asignatura, contiene ejemplos en cada uno de sus apartados y ofrece preguntas de respuesta corta en cada apartado para facilitar el estudio. Como el libro incluye más contenidos de los necesarios para estudiar la asignatura, el equipo docente deberá utilizar la Guía de estudio (Parte II) para orientar adecuadamente al alumno.
Integración adecuada con el resto de los contenidos de la materia y de la titulación (evitando solapamientos y lagunas)	No se han detectado lagunas ni solapamientos con asignaturas de la materia ni con otras de la titulación.
Contenidos actualizados (si la asignatura lo requiere)	La fecha de la publicación es de 2008, El libro es un referente en la materia y permite al estudiante asentar las bases para el diseño de compiladores.

Adecuación del Texto como material de enseñanza-aprendizaje a distancia <sup>1</sup>	Valoración Global del IUED: Con propuestas de mejora. Para más detalles, ver informe de IUED.
Adecuación del Texto a la normativa de la UNED sobre edición de Textos Básicos.	Se recomienda un único texto, que está editado por la editorial PEARSON-ADDISON WESLEY.

### VALORACIÓN, Y EN SU CASO, RECOMENDACIONES FINALES

El libro es correcto para un primer curso de diseño de compiladores. Presenta al alumno la resolución de problemas universalmente encontrados en el diseño de un lenguaje traductor, con independencia de la máquina fuente u objetivo. Este libro se utilizará tanto en la asignatura de Procesadores del Lenguaje I como en la Procesadores del Lenguaje II por lo que es necesario dejar claro al alumno qué debe estudiar en cada una de ellas y su correspondencia con los contenidos publicados en la guía docente. Además, es importante que en la guía II de la asignatura el equipo docente indique los objetivos de cada capítulo.

En cuanto a los contenidos de los capítulos correspondientes, el libro constituye un material ADECUADO para cubrir completamente la asignatura.

El equipo docente deberá tener en cuenta las recomendaciones y comentarios realizados en el informe emitido por el IUED.

Madrid, a 16 de junio de 2016

**La Comisión Coordinadora del Título, o en su  
representación el/la Coordinador/a del Título**

**Fdo.**

**Vº Bº El/La Decano/a  
Director/a**

<sup>1</sup> Para hacer esta valoración la Comisión Coordinadora del Título dispondrá del informe técnico del IUED.

## PROTOCOLO DE VALORACIÓN DE LA ADECUACIÓN DE DE LOS LIBROS DE TEXTO DIDÁCTICOS

IDENTIFICACIÓN DEL LIBRO DE TEXTO										
TÍTULO DE GRADO										
INGENIERÍA INFORMÁTICA										
FACULTAD / ESCUELA										
INFORMÁTICA										
MATERIA										
LENGUAJES DE PROGRAMACIÓN										
DEPARTAMENTO										
LENGUAJES Y SISTEMAS INFORMÁTICOS										
ASIGNATURA										
PROCESADORES DEL LENGUAJE I								Código		71013130
Curso en que se imparte:				Carácter			Duración			ECTS
1º	2º	3º	4º	FB	OB	OPT	ANUAL		SEMESTRAL	
		X			X			1er. X	2do.	6
EQUIPO DOCENTE										
Anselmo Peñas y Álvaro Rodrigo										
AUTOR/ES										
Alfred V. Aho, Monica S. Lam, Ravi Sethi y Jeffrey D. Ullman										
EDITORIAL										
Pearson-Addison Wesley (Título: Compiladores. Principios, técnicas y herramientas)										

REFERENTES	
<i>(Añádase el comentario que corresponda en la columna valoración y recomendaciones)</i>	
	<i>Valoración y, en su caso, recomendaciones</i>
Ajuste en la extensión a los ECTS de la asignatura	El libro propuesto contiene más temas de los exigidos por el temario de la asignatura (según la memoria ANECA de verificación del plan de estudios). Sin embargo, restringiendo el estudio a los temas que conforman el temario, la extensión se ajustaría al número de créditos de la asignatura.
Adecuación al nivel de conocimientos previos de los alumnos	El estudio de esta asignatura requiere conocimientos de varias asignaturas: "Fundamentos de Programación", "Programación Orientada a Objetos", "Estrategias de Programación y Estructuras de Datos", "Programación y Estructuras de Datos Avanzadas" y "Autómatas, Gramáticas y Lenguajes".
Adecuación a los contenidos de la asignatura, sus características y su lugar en el plan de estudios	El libro presenta adecuadamente los contenidos de la asignatura, contiene ejemplos en cada uno de sus apartados y ofrece preguntas de respuesta corta en cada apartado para facilitar el estudio. Como el libro incluye más contenidos de los necesarios para estudiar la asignatura, el equipo docente deberá utilizar la Guía de estudio (Parte II) para orientar adecuadamente al alumno.
Integración adecuada con el resto de los contenidos de la materia y de la titulación (evitando solapamientos y lagunas)	No se han detectado lagunas ni solapamientos con asignaturas de la materia ni con otras de la titulación.
Contenidos actualizados (si la asignatura lo requiere)	La fecha de la publicación es de 2008, El libro es un referente en la materia y permite al estudiante asentar las bases para el diseño de compiladores.

Adecuación del Texto como material de enseñanza-aprendizaje a distancia <sup>1</sup>	Valoración Global del IUED: Con propuestas de mejora. Para más detalles, ver informe de IUED.
Adecuación del Texto a la normativa de la UNED sobre edición de Textos Básicos.	Se recomienda un único texto, que está editado por la editorial PEARSON-ADDISON WESLEY.

### VALORACIÓN, Y EN SU CASO, RECOMENDACIONES FINALES

El libro es correcto para un primer curso de diseño de compiladores. Presenta al alumno la resolución de problemas universalmente encontrados en el diseño de un lenguaje traductor, con independencia de la máquina fuente u objetivo. Este libro se utilizará tanto en la asignatura de Procesadores del Lenguaje I como en la Procesadores del Lenguaje II por lo que es necesario dejar claro al alumno qué debe estudiar en cada una de ellas y su correspondencia con los contenidos publicados en la guía docente. Además, es importante que en la guía II de la asignatura el equipo docente indique los objetivos de cada capítulo.

En cuanto a los contenidos de los capítulos correspondientes, el libro constituye un material ADECUADO para cubrir completamente la asignatura.

El equipo docente deberá tener en cuenta las recomendaciones y comentarios realizados en el informe emitido por el IUED.

Madrid, a 16 de junio de 2016

**La Comisión Coordinadora del Título, o en su  
representación el/la Coordinador/a del Título**

**Fdo.**

**Vº Bº El/La Decano/a  
Director/a**

<sup>1</sup> Para hacer esta valoración la Comisión Coordinadora del Título dispondrá del informe técnico del IUED.



GRADO

# GUÍA DE ESTUDIO DE LA ASIGNATURA PROCESADORES DE LENGUAJES I

2ª PARTE | PLAN DE TRABAJO Y ORIENTACIONES PARA SU DESARROLLO



2016-2017

Anselmo Peñas Padilla  
Álvaro Rodrigo Yuste  
GRADO EN INFORMÁTICA

## 1.- PLAN DE TRABAJO

El estudio de los conceptos teóricos debe plantearse con el objetivo de ser aplicados, es decir, en todo momento el alumno debe ir integrando los conceptos que va adquiriendo a la comprensión de cómo se desarrolla un compilador y las distintas alternativas que se le presentan en su diseño y desarrollo. Por esta razón, la práctica es una herramienta fundamental que acompaña al alumno a lo largo de todo el curso.

El temario ha sido planteado de tal forma que el alumno pueda introducirse en los contenidos de la asignatura de una manera gradual, adquiriendo los conocimientos necesarios para comprender la secuencia de traducción de un lenguaje. La asignatura de Procesadores de Lenguajes I se centra en los pasos de análisis léxico y sintáctico del compilador. La asignatura consta de 6 créditos lo que equivale a unas 150 horas de trabajo del alumno. De esas 150 horas, deberían dedicarse:

- 100 horas al estudio teórico-práctico, incluyendo conceptos teóricos, resolución de problemas y ejercicios y adquisición de los conceptos teórico-prácticos necesarios para desarrollar la práctica
- 50 horas dedicadas a la programación de la práctica.

Considerando 15 semanas en un semestre, el alumno debería dedicar a la asignatura unas 10 horas a la semana.

SEMANA	CONTENIDO	HORAS DE TRABAJO TEÓRICO-PRÁCTICO	HORAS DE TRABAJO PRÁCTICO
1	1. Introducción		0
	1.1. Procesadores de Lenguaje (introducción al tema 1 y apartado 1.1 del libro)	1	
	1.2. Estructura de un compilador (apartado 1.2 del libro)	0.5	
1, 2	2. Análisis léxico		10
	2.1. Proceso de análisis léxico (apartado 3.1 del libro)	1	
	2.2. Expresiones Regulares (apartado 3.3 del libro)	1.5	
	2.3. Autómatas finitos (apartados 3.6, 3.7, 3.9.5 y 3.9.6 del libro)	3	

	2.4. Uso de Lex/jLex para la generación automática de analizadores léxicos (apartado 3.5 del libro y manual de jLex. Esto va orientado sobre todo a la práctica. La información del libro sirve como apoyo, ya que la sintaxis a usar debe ser la de JLex)	3	
3, 4, 5, 6	3. Análisis sintáctico		10
	3.1. Proceso de análisis sintáctico (apartado 4.1 del libro)	1	
	3.2. Gramáticas libres de contexto (apartados 4.2, 4.3 y 4.8 (excepto apartado 4.8.3) del libro)	19	
	3.3. Uso de Yacc/Cup para la generación automática de analizadores sintácticos (apartado 4.9 del libro y manual de Cup. Esto va orientado sobre todo a la práctica. La información del libro sirve como apoyo, ya que la sintaxis a usar debe ser la de Cup)	5	
6, 7, 8, 9,10	4. Análisis sintáctico descendente		15
	4.1. Análisis sintáctico descendente recursivo (apartado 4.4.1 del libro)	5	
	4.2. Conjuntos Primero y Siguiente (apartado 4.4.2 del libro)	1	
	4.3. Análisis sintáctico LL(1) (4.4.3 del libro)	14	
	4.4. Recuperación de errores en analizadores sintácticos descendentes (apartado 4.4.5 del libro)	5	

## PROCESADORES DE LENGUAJES I

10, 11, 12,13, 14, 15	5. Análisis sintáctico ascendente		15
	5.1. Introducción (apartado 4.5 del libro)	1	
	5.2. Análisis sintáctico LR(0) y SLR(1) (apartado 4.6 del libro)	17	
	5.3. Análisis sintáctico LALR(1) y LR(1) (apartado 4.7 del libro)	17	
	5.4. Recuperación de errores en analizadores sintácticos ascendentes (apartado 4.8.3 del libro)	5	

El estudio de la teoría está relacionado directamente con su implementación práctica. Por esta razón, para un mayor aprovechamiento de los conocimientos impartidos, se considera conveniente solapar la realización de la práctica con el estudio de la teoría.

### **2.- ORIENTACIONES PARA EL ESTUDIO DE LOS CONTENIDOS**

El temario teórico se corresponde con el libro de texto de la asignatura. En el entorno virtual los alumnos pueden encontrar tanto cuestiones y problemas resueltos como sus soluciones, con las que podrá realizar una autoevaluación de sus conocimientos.

Asimismo encontrará 3 documentos importantes:

1. Normas de la asignatura
2. Enunciado de la práctica
3. Directrices de Implementación

También encontrará la plataforma de desarrollo de la práctica que el alumno utilizará de acuerdo con el documento de Directrices de Implementación.

### 3.- ORIENTACIONES PARA LA REALIZACIÓN DE LA PRÁCTICA

La construcción de un compilador es una compleja labor que requiere atender a numerosas cuestiones técnicas como la implementación de estructuras de datos orientadas a objetos, la organización de éstas en paquetes, el uso de diversos patrones de diseño, etc.

Para el desarrollo de la práctica de procesadores de lenguajes se proporciona al alumno un marco de trabajo de referencia. Este marco de trabajo se proporciona con el siguiente fin:

- Reducir el tiempo de puesta en marcha del entorno necesario para el desarrollo de la práctica.
- Acelerar la adquisición de buenas prácticas en el desarrollo de un compilador
- Centrar el trabajo del alumno en el desarrollo de aquellos aspectos que inciden directamente con los contenidos propios de la asignatura.

En este documento se presenta el marco de trabajo software que se proporciona al alumno y se describe detalladamente el trabajo que el alumno debe realizar para superar satisfactoriamente las entregas de Febrero y de Junio. La corrección de la práctica se apoya en la organización software que prescribe este marco de trabajo de trabajo y, por tanto, es normativo que el alumno respete completamente dicha organización. Cualquier violación a este respecto podría suponer un suspenso en el proceso de corrección.

Para la comprensión de este documento, se presupone que el alumno ha estudiado previamente la teoría de la asignatura y que por tanto está capacitado para entender de forma correcta las explicaciones y terminología ofrecida.

Las normas de entrega y división del trabajo deben consultarse en el enunciado de la práctica.

En caso de correcciones sobre el marco de trabajo se anunciarán en el Tablón de Anuncios de la asignatura. *Es necesario por tanto que el estudiante visite con asiduidad los foros de la asignatura.*

#### MARCO DE TRABAJO Y DESARROLLO DE LA PRÁCTICA

El desarrollo de la práctica se realiza sobre un marco de trabajo tecnológico y por tanto constituye un recurso necesario para el desarrollo de la misma. El marco de trabajo se entrega como un fichero comprimido que proporciona dos elementos fundamentales:

- **Una estructura de directorios para que el alumno organice su código.** Estos directorios contiene ficheros fuente en java con código de andamiaje que el alumno deberá completar y / o modificar para implementar las distintas funcionalidades del compilador. Estas clases están organizadas jerárquicamente en paquetes siendo **compiler** el paquete raíz. Por tanto, cualquier clase dentro del paquete compiler (o alguno de sus paquetes hijos) es una clase abierta que el alumno puede y debe modificar. Consulte la API en el marco de trabajo y una descripción más detallada en la sección 2.1 para obtener más información.
- **Una librería en java que proporcionan clases de soporte para el desarrollo del compilador.** Todas las clases abiertas del paquete compiler se apoyan en clases definidas dentro de la librería compiler-api.jar. Esta librería contiene, básicamente, clases de ayuda, clases abstractas e interfaces que utilizan, extienden o implementan la mayoría de las clases del paquete compiler (y

sus paquetes hijos). Se trata de una familia de clases cerradas y compiladas por lo que el alumno no dispone de su código fuente para alterarlas ya que NO debe ni puede modificarlas. Todas las clases de soporte, dentro de la librería, están organizadas jerárquicamente en paquetes siendo **es.uned.lsi.compiler** el nombre del paquete padre. Por tanto toda clase dentro de éste paquete (o alguno de sus paquetes hijos) es una clase de soporte proporcionada por el equipo docente que no debe modificarse. Consulte la API en el marco de trabajo o el resto del documento para obtener una información más detallada.

Tanto la estructura de directorios como las implementaciones y clases de referencia están en inglés. Se presupone, por tanto, que el alumno está familiarizado con la terminología inglesa a la hora de programar. No es necesario que el alumno complete la práctica en inglés, pero no ha de traducir el código proporcionado.

Además, debe quedar claro que la práctica se desarrolla en Java y dentro del paradigma de orientación a objetos. Existe una amplia literatura sobre este lenguaje de programación como referencias podemos destacar <http://java.sun.com/> y <http://www.javahispano.org/>. En esta última se puede encontrar documentación en castellano.

### LA ESTRUCTURA DE DIRECTORIOS

A continuación se procede a explicar la estructura de directorios del marco de trabajo, deteniéndonos en profundidad cada uno de los paquetes donde debe almacenarse el código.

- **src.** En este directorio deben almacenarse todo el código fuente del compilador. Las clases están organizadas en diferentes paquetes. El paquete principal se llama 'compiler' y dentro existen otros paquetes: uno por cada una de las fases conceptuales que constituyen el compilador. En este directorio deben almacenarse todo el código fuente del compilador organizado en paquetes tal y como describiremos a continuación. El contenido de este directorio contiene algunas clases de soporte de las que el alumno debe partir para el desarrollo de la práctica.
  - **/compiler/lexical.** Este directorio debe almacenar las clases del paquete 'compiler.lexical' que implementan el scanner de la práctica y todas sus clases asociadas (incluido scanner.java). En este paquete se incluye la clase Token.java que el alumno debe utilizar y puede completar y/o extender para realizar el analizador léxico tal y como se describirá más adelante.
  - **/compiler/syntax.** Este directorio debe almacenar todas las clases del paquete 'compiler.syntax' que implementan el parser y todas sus clases asociadas (incluidas parser.java y sym.java).
  - **/compiler/syntax/nonTerminal.** En este directorio se almacenarán las clases que representan los símbolos no terminales necesarios para llevar a cabo la traducción dirigida por la sintaxis en cup como comentaremos más adelante. Se incluyen las clases Axiom y NonTerminal. La primera representa al axioma de la gramática. La segunda representa la clase abstracta de la que deben extender las clases de no terminales creadas por el estudiante.

- **/compiler/semantic.** Contendrá las clases necesarias para el análisis semántico. Está compuesto de varios paquetes organizativos.
  - **/compiler/semantic/symbol.** En este directorio se encuentran las clases necesarias para utilizar la tabla de símbolos, incluyendo los diferentes símbolos que puede almacenar.
  - **/compiler/semantic/type.** En este directorio se almacenan las clases necesarias para utilizar el sistema de tipos del lenguaje.
  - **/compiler/intermediate.** En este directorio estarán las clases necesarias para la generación del código intermedio mediante cuartetos (también llamados cuádruplas) junto con los tipos de operandos que admiten.
  - **/compiler/code.** En este directorio se situarán las clases necesarias para la generación del código final.
  - **/compiler/test.** En este directorio se encuentran tres clases llamadas `LexicalTestCase.java`, `SyntaxTestCase.java` y `FinalTestCase.java` que sirven para probar el analizador léxico, sintáctico y la entrega final respectivamente. Más adelante se detallará cómo ejecutar y probar la práctica.
- **classes.** En este directorio se almacenarán las clases (archivos de extensión `.class`) que se generan mediante el proceso de compilación del código fuente del compilador. El alumno no debe modificar este directorio, ya que la herramienta Ant, que presentaremos más adelante, se encarga de automatizar las tareas de compilación. Es importante que los ficheros `.class` se sitúen en este directorio y no en otros creados automáticamente por algunos IDEs, como por ejemplo el directorio `bin`.
  - **lib.** Este directorio contiene las librerías necesarias para la realización de la práctica (archivos con extensión `.jar`). Aquí se incluyen las librerías `jflex.jar` y `cup.jar` necesarias para llevar a cabo el proceso de generación del scanner y el parser respectivamente, y `compiler-api.jar` que contiene las clases e interfaces y clases necesarias para el desarrollo del compilador. El contenido de esta última librería será discutida en detalle más adelante. Todas las librerías bajo el directorio `/lib` NO deben eliminarse bajo ningún concepto. Aunque no es necesario, si el alumno considera oportuno hacer uso de otras librerías de apoyo debe incluirlas en este directorio e indicarlo en la memoria.
  - **doc.** Este directorio contiene diferentes tipos de documentos necesarios para la realización y documentación de la práctica organizados de acuerdo a una serie de subdirectorios que se explican a continuación:
    - **/api.** En este directorio se almacena una completa descripción de las clases (el API) que constituyen el marco de trabajo incluido en la librería `'compiler-api.jar'`. El formato de la documentación responde al estándar `'javadoc'` definido por Sun. Para visualizarlo en forma de página Web ha de accederse a `index.html`.
    - **/config.** En este directorio se proporcionan el fichero de configuración de la aplicación: `'build.xml'` que incluye las descripción de las tareas Ant para realizar la práctica (véase más adelante una descripción en profundidad de este documento). Nótese no obstante, que el

alumno no requiere comprender el contenido de este fichero sino solamente saber utilizarlo con el uso de la herramienta Ant.

- **/memoria.** Aquí debe incluirse la memoria de la práctica según las normas explicadas en el documento del enunciado de la práctica.
- **/specs.** Este directorio incluye los ficheros para generar el analizador léxico o scanner (scanner.flex) y el analizador sintáctico o parser (parser.cup) que deben ser completados por el alumno. Para un correcto funcionamiento NO se debe cambiar su ubicación a otro directorio.
- **/test.** Contiene los ficheros de prueba para probar el compilador, desde la tarea ant correspondiente (ver más adelante). Cualquier fichero fuente de prueba que desee utilizar el alumno deberá almacenarse en este directorio.

### LA LIBRERÍA COMPILER - API

La librería de referencia compiler-api.jar contiene una familia de clases organizadas en diferentes paquetes que deben ser utilizados por el alumno para implementar el compilador. En el directorio /doc/api puede encontrar una completa descripción técnica del contenido de este fichero jar. No obstante, a continuación describimos el uso, utilidad y organización en paquetes de cada una de las clases incluidas:

- Paquete **es.uned.lsi.compiler.lexical.** Este paquete contiene las clases que el alumno debe utilizar para desarrollar el analizador léxico de su compilador. En concreto podemos destacar las siguientes clases:
  - La clase **LexicalError.** Esta clase es una excepción de java (hereda de java.lang.Exception) que encapsula toda la información relativa a un error ocurrido durante el proceso de análisis léxico. Esencialmente incorpora 3 atributos miembro con sus correspondientes métodos de acceso get/set: line, el número de línea donde se encontró el error; column, el número de columna donde se encontró el error y lexema, el lexema que produjo dicho error. Esta clase debe construirse y emitirse como un mensaje de error léxico por la salida estándar mediante el uso de la clase LexicalErrorManager tal y como se ilustrará más adelante.
  - La clase **LexicalErrorManager.** Esta clase es un gestor de errores que permite generar automáticamente y emitir los mensajes de error léxicos que encuentra el scanner durante el análisis de un fichero de código fuente mediante el uso de la función lexicalError. por ejemplo la llegada de un carácter inválido o la ausencia de concordancia del lexema a la entrada con ningún patrón léxico definido en la especificación JFlex. Si el error es irreparable debe utilizarse el método lexicalFatalError, que provoca la terminación del proceso de análisis léxico. Además esta clase puede utilizarse para realizar trazas de información y depurado que informen del avance en el proceso de análisis léxico. En concreto se proporcionan 3 tipos de métodos: lexicalInfo, de uso opcional para emitir mensajes de información por la salida estándar; lexicalDebug de uso opcional para emitir

mensajes de depuración por la salida estándar y `lexicalError`, de uso obligatorio para emitir mensajes de error por la salida estándar.

- Paquete **es.uned.lsi.compiler.syntax**. Este paquete contiene las clases que el alumno debe utilizar para desarrollar el analizador sintáctico de su compilador. A continuación se describe su clase principal:
  - La Clase **SyntaxErrorManager**. Esta clase persigue un objetivo similar al de la clase `LexicalErrorManager` con la diferencia de que su contexto de uso es, en este caso la especificación gramatical de Cup. Mas adelante se describirá el uso que puede hacerse de la misma del fichero `parser.cup`. De momento destacamos la existencia de los siguientes tipos de métodos: `syntaxInfo`, de uso opcional para emitir mensajes de información por la salida estándar; `syntaxDebug`, de uso opcional, para emitir mensajes de depuración por la salida estándar que asistan en las tareas de depuración; `syntaxError`, de uso obligatorio, para emitir mensajes de error recuperables por la salida estándar y `syntaxFatalError`, de uso obligatorio para emitir mensajes de error irre recuperables por la salida estándar
- Paquete **es.uned.lsi.compiler.syntax.nonTerminal**. Este paquete contiene la interfaz `NonTerminalIF` de la que extiende la clase abstracta `NonTerminal`, explicada anteriormente dentro del paquete `compiler.syntax.nonTerminal`.
- Paquete **es.uned.lsi.compiler.semantic** Contiene la clase `SemanticErrorManager` (ver apartado de análisis semántico) usada para indicar los errores semánticos junto con métodos para ayudar a realizar trazas personalizadas (`semanticDebug` y `semanticInfo`). Su uso es similar al `LexicalErrorManager` o `SyntaxErrorManager`. Además en este paquete residen las clases `Scope` y `ScopeManager` necesarias para dar soporte a la gestión de ámbitos de visibilidad.
- Paquete **es.uned.lsi.compiler.semantic.symbol** Contiene la colección de clases necesarias para dar soporte a los símbolos que aparecen durante la compilación de un programa. En concreto nos referimos al interfaz `SymbolIF` y a la clase base que lo extiende `SymbolBase` y de la cual derivan todas las clases abiertas que el alumno deberá modificar en el paquete `compiler.semantic.symbol` para realizar su compilador tal y como se describirá más adelante. También se ofrece una *implementación completa de la tabla de símbolos denominada `SymbolTable`*.
- Paquete **es.uned.lsi.compiler.semantic.type** Contiene las clases necesarias para dar soporte al sistema de tipos del compilador. En concreto allí se puede encontrar el interfaz `TypeIF` y la clase abstracta que implementa dicho interfaz `TypeBase`. También hay definido en este paquete el interfaz `TypeTableIF` que es implementado por la clase `TypeTable`, también proporcionada. Por lo tanto, al igual que la tabla de símbolos, no es necesario implementar la tabla de tipos.
- Paquete **es.uned.lsi.compiler.intermediate**. Dentro de este paquete aparecen todos los artefactos necesarios para la generación de código intermedio. En este sentido se incluyen las interfaces e implementaciones para gestionar cuartetos, temporales, etiquetas y variables. También se incluye la clase `IntermediateCodeBuilder` necesaria para construir de forma sencilla las colecciones de cuarteos.

- Paquete **es.uned.lsi.compiler.code** En el paquete code residen los artefactos necesarios para generar código final a partir de código intermedio. La clase FinalCodeFactory organiza la generación de código final y debe usarse para crear el código final a partir de la lista de cuádruplas. Para la conversión de cuádruplas a código final se define el entorno de ejecución, a partir del interfaz ExecutionEnvironmentIF y que el alumno debe completar en la clase ExecutionEnvironmentEns2001, situada en el paquete compiler.code. También se ofrecen las interfaces MemoryDescriptorIF y RegisterDescriptorIF implementadas, igualmente, en el paquete compiler.code cuyo uso, opcional, asiste en las tareas de gestión de registros máquina y memoria

### INSTALACIÓN

La instalación del marco de trabajo es un proceso sencillo que puede resumirse en la secuencia de pasos que exponemos a continuación:

- Entre al entorno de WebCT y acceda al curso titulado PROCESADORES DEL LENGUAJE
- Acceda a PRACTICA > HERRAMIENTAS
- Descargue el fichero Arquitectura.zip
- Cree una carpeta para albergar el desarrollo del proyecto, por ejemplo C:/PDL.
- Descomprima el fichero Arquitectura.zip bajo ese directorio

Si desea utilizar algún entorno de desarrollo integrado puede hacerlo. En ese caso debería crear un nuevo proyecto y ubicar allí la estructura de directorios del fichero Arquitectura.zip asegurándose de indicar que el directorio /scr contiene los fuentes del proyecto y que la versión compilada de los mismos deberá redirigirse al directorio /classes.

### DESARROLLO

Las tareas de generación, compilación, ejecución y prueba de los analizadores léxico y semántico que son necesarias para realizar el compilador se han automatizado mediante la herramienta Ant. Esta es una herramienta que permite automatizar diferentes tipos de tareas descritas en un script expresado como un documento en XML (normalmente llamado build.xml). Su uso resulta similar a los mecanismos de procesamiento por lotes proporcionados por algunos sistemas operativos, que usan ficheros '.bat' o '.sh' para describir un script de actuación o a las herramientas Makefile utilizadas por algunos compiladores para automatizar los procesos de compilación y enlazado de acuerdo a ficheros '.mak'.

El primer paso para utilizar la herramienta ANT es la instalación de la misma. Se puede encontrar toda la información acerca de ella en la siguiente dirección Web:

<http://ant.apache.org>

El documento build.xml proporcionado en el directorio doc/config contiene la descripción de todas las tareas necesarias para realizar el compilador. No obstante, para que éstas funcionen es necesario importar las JFlex.jar y Cup.jar que se incluyen en el directorio /lib. Por ello, se recomienda copiar ambos archivos

dentro del directorio 'lib' de Ant. Por ejemplo, si tenemos Ant instalado en C:/ant, debemos copiar los dos archivos en el directorio C:/ant/lib

Para invocar las tareas de Ant desde la línea de comandos debe abrirse una consola de sistema y situarse sobre el directorio que contenga el fichero con la especificación de las tareas (en nuestro caso build.xml) y escribir:

```
ant nombreTarea -Dpropiedad=valor
```

Donde 'nombreTarea' es el nombre de una tarea definida en el fichero de tareas XML, 'propiedad' el nombre de una propiedad definida en ese fichero y 'valor' el valor que tendrá esa propiedad, siendo este parámetro opcional. Además los ficheros de descripción de tareas de Ant permiten especificar una tarea por omisión. En ese caso no haría falta poner el nombreTarea tampoco<sup>1</sup>.

Para realizar el compilador en el directorio /doc/build se proporciona el fichero build.xml que se incluye y que contiene la definición de las siguientes tareas, que están a disposición del alumno:

- clear. Borra los archivos generados por procesos anteriores de compilación.
- jflex. Lee el archivo 'doc/specs/scanner.flex' y genera el código fuente del analizador léxico asociado en el archivo 'src/compiler/lexical/Scanner.java'.
- cup. Lee el archivo 'doc/specs/parser.cup' y genera los archivos de código fuente asociados 'src/compiler/syntax/sym.java' y 'src/compiler/syntax/parser.java' que corresponden a la implementación del analizador sintáctico.
- build. Llama por orden a las tres tareas anteriores y compila todos los archivos fuente, dejando las clases generadas en el directorio 'classes'. Queda así generado el compilador.
- flexTest. Llama al método main de la clase 'LexicalTestCase' pasándola como parámetro el nombre de un fichero situado en el directorio /doc/test. Para indicar qué fichero queremos compilar se ha de modificar en el fichero build.xml la siguiente línea:

```
<property name="test-file-name" value="testA.ha" />
```

Cambiando 'value' con el nombre del fichero que queramos. Por ejemplo el grupo B debe poner 'testA.ha'. Los casos de prueba que desarrolle el alumno deberán situarse en el directorio /doc/test y llamarlos tal y como se ha indicado. Otra opción es pasar este valor como parámetro en la llamada a Ant. Por ejemplo:

```
Ant flexTest -Dtest-file-name=testB.ha
```

- cupTest. Llama al método main de la clase SyntaxTestCase de forma similar a lo descrito en la tarea 'flexTest'. En este caso se probará el analizador léxico y el sintáctico.

---

<sup>1</sup> Existen también entornos de desarrollo integrados (IDEs) que disponen de la herramienta Ant incorporada lo que simplifica su uso. Un ejemplo de tales entornos es Eclipse ([www.eclipse.org](http://www.eclipse.org))

- `finalTest`. Llama al método `main` de la clase `FinalTestCase` de forma similar a `flexTest` y `cupTest`. Esta clase prueba el funcionamiento del compilador generando los ficheros con el código final.

Estas tres últimas tareas de test invocan previamente a las tareas `clear` y `build`, por lo que no es necesario llamarlas previamente.

### ENTREGA

El alumno solamente deberá prestar atención a las clases de soporte dentro de los paquetes `es.uned.lsi.compiler.lexical` y `es.uned.lsi.compiler.syntax` de la librería `compiler-api.jar` y (en todo caso) modificar las clases dentro de los paquetes `compiler.lexical` y `compiler.syntax`.

Sin embargo, la práctica se centra en definir las especificaciones para los analizadores léxico y sintáctico (ficheros `JFlex` y `Cup` respectivamente) con lo que no es preciso grandes modificaciones de las clases proporcionadas.

### Análisis léxico

El analizador léxico debe implementarse dentro del paquete `'compiler.lexical'`. Para ello, ha de utilizarse la clase proporcionada `'Token.java'` como medio de comunicación con el analizador sintáctico. Por tanto, el objeto que ha de devolverse al pedir un `TOKEN` (mediante el uso de la función `next_token` desde el parser) ha de ser de ese tipo (como se ve en el código de `scanner.flex` y en el listado 1). Esta clase puede ser completada con más atributos y métodos si el alumno lo considera necesario pero NO ha de eliminarse nada de código proporcionado (consulte la documentación en `doc/api`).

El alumno ha de modificar `scanner.flex` para crear el autómata apropiado que reconozca los `TOKENS` del lenguaje pedidos.

Dentro de `scanner.flex` se crea un nuevo `TOKEN` con la sentencia:

```
Token token = new Token (CONSTANTE_NUMERICA);
```

Donde `CONSTANTE_NUMERICA` es un valor numérico arbitrario que identifica de manera unívoca el tipo de `Token` emitido. Adicionalmente, es necesario incorporar dentro del objeto `Token` información acerca del lexema del `TOKEN`, así como el número de línea y columna donde está ubicado dentro del código fuente<sup>2</sup>. Estas tres informaciones las provee `JFlex` a través del método `yyText ()` y de las variables `yyline` e `yycolumn` respectivamente. Por tanto, tal como puede verse en el fichero proporcionado la secuencia de instrucciones para emitir un `Token` de tipo `SUMA` sería similar a lo siguiente:

#### Listado 1. Acción para emitir un token SUMA desde JFlex.

```
Token token = new Token (SUMA);  
  
token.setLine (yyline + 1);  
  
token.setColumn (yycolumn + 1);  
  
token.setLexema (yytext ());
```

<sup>2</sup> Esta información será posteriormente explotada por el analizador sintáctico descrito con `Cup`.

```
return token;
```

En aquellos casos en los que el alumno identifique que el analizador léxico debe emitir un error léxico las acciones a realizar son dos. En primer lugar construir un objeto de la clase `LexicalError` para encapsular toda la información relativa al error. Esta clase encapsula esencialmente el lexema a la entrada que causó el error y el número de línea y columna dentro del fichero fuente de entrada donde se detectó el mismo (consulte la documentación en doc/api). En segundo lugar, debe utilizarse la clase `LexicalErrorManager` para emitir el mensaje de error por la salida estándar (consulte la documentación en doc/api)

## Listado 2. Acción para emitir un mensaje de error léxico desde JFlex

```
LexicalError error = new LexicalError ();

error.setLine (yyline + 1);

error.setColumn (yycolumn + 1);

error.setLexema (yytext ());

lexicalErrorManager.lexicalError (error);
```

La clase `'LexicalErrorManager'` aparte de ser usada para mostrar los errores léxicos puede ser usada como *mecanismo de traza* dentro de la especificación de JFlex. Para ello se ofrecen los métodos `'lexicalDebug'` y `'lexicalInfo'`, que permiten emitir un mensaje, pasado como parámetro por la salida estándar. Las llamadas a `lexicalDebug` se utilizan para emitir informaciones que ayudan a depurar el código del analizador léxico. Las llamadas a `lexicalInfo` se utilizan para emitir mensajes informativos acerca del estado de progreso del analizador léxico. En todo caso se deja al alumno la forma de utilizar dichos métodos. Aunque no debe olvidar que al producirse un error es **obligatoria** la llamada a `'lexicalErrorManager.lexicalError'` tal y como se ha explicado.

## Análisis sintáctico

El analizador sintáctico se implementa dentro del paquete `compiler.syntax` y las clases generadas por la especificación de Cup se incluyen automáticamente dentro de dicho paquete. El trabajo de esta segunda fase consiste en completar el fichero de especificación `parser.cup` para obtener las clases `sym.java` y `parser.java`.

Dentro de la especificación incluida en el marco de trabajo de referencia existen una serie de declaraciones que NO deben modificarse bajo ningún concepto (aunque sí pueden ampliarse). Entre ellas destaca:

- El bloque *parser code*. Dentro de este bloque se incluyen las declaraciones de las funciones manejadoras de los errores sintácticos. Estas funciones son llamadas de forma automática por el parser al producirse un error recuperable o irrecuperable en el análisis sintáctico y utilizan llamadas al método `'syntaxError'` o `'syntaxFatalError'` de la clase `'SyntaxErrorManager'`. De forma similar a como se explicó en el análisis léxico, esta clase se utiliza como gestor de errores pero también puede ser utilizada como mecanismo de traza, disponiendo de los métodos para `syntaxDebug` y `syntaxInfo` homólogos a los descritos en la sección anterior.

- Bloque *action code*. Dentro de este bloque se incluye la declaración de las variables que deseen accederse desde las acciones semánticas de la gramática. Aunque estén declarados varios gestores, para la primera entrega sólo es necesario el gestor de errores `SyntaxErrorManager`.

Por último es necesario describir cuáles son las modificaciones que hay que hacer en la especificación de JFlex para que se integre con la herramienta Cup. La estructura del documento de especificación JFlex (`scanner.flex`) ya está preparada para posibilitar la integración. No obstante, las acciones asociadas a cada regla patrón acción en JFlex tienen que cumplir las siguientes restricciones:

- Cada acción debe emitir un objeto `Token` tal y como se explicó en el listado 1
- El `Token` emitido debe corresponderse con alguno de los terminales definidos en Cup.

En efecto, cuando se describe la especificación gramatical de Cup, se debe indicar el conjunto de elementos terminales que la constituyen. Para ello existe, dentro del fichero de especificación de Cup una sección específica de Cup. El siguiente listado es un ejemplo de esta sección que se corresponde con las declaraciones proporcionadas en la estructura de referencia. El alumno deberá reescribir estas declaraciones para que se ajusten a la especificación de su compilador.

### Listado 3. Bloque de declaración de terminales en Cup

```
// Declaración de terminales (Ejemplo)

terminal Token PLUS;

terminal Token MINUS;
```

En este ejemplo se ha declarado en la gramática se utilizarán dos elementos terminales llamados `PLUS` y `MINUS` para representar al token '+' y '-' respectivamente e indicar que son objetos de tipo `Token` (ya que es la clase que manejarán JFlex y Cup). Estas declaraciones sirven para indicar a Cup que debe generar una clase que contenga una constante simbólica de tipo entero para representar cada elemento terminal. En concreto, en el ejemplo anterior la clase que se generaría contendría la declaración de dos constantes, llamadas `PLUS` y `MINUS`.

De acuerdo a esta idea, el último paso de integración que tenemos que realizar consiste en cambiar la acción para emitir `TOKEN`, mostrada en el listado 1, por la que se ilustra en el listado 4.

### Listado 4. Acción para emitir un token SUMA desde JFlex hacia Cup

```
Token token = new Token (sym.PLUS);

token.setLine (yyline + 1);

token.setColumn (yycolumn + 1);

token.setLexema (yytext ());

return token;
```

La única diferencia con el listado 1 es que ahora el identificador único de `TOKEN` es el valor de una de las constantes definidas dentro de la clase `Cup`.

## REALIZACIÓN DE PRUEBAS

Para terminar la descripción de este marco de trabajo vamos a discutir el contenido de las clases `LexicalTestCase` y `SyntaxTestCase` que son utilizadas para probar sendas las fases del compilador para esta primera entrega.

### Pruebas del analizador léxico

Para evaluar el analizador léxico generado a través de la especificación de JFlex deben seguirse los siguientes pasos:

- Escribir un fichero de código fuente del lenguaje especificado en la práctica (correcto o incorrecto) y depositarlo dentro del directorio `doc/test`
- Escribir la especificación completa del analizador léxico completando la estructura proporcionada dentro del fichero `doc/specs/scanner.flex` de acuerdo a las directrices impuestas a lo largo de este documento
- Si la especificación de JFlex referencia las constantes de la clase `sym` generadas por el analizador sintáctico ejecutar la tarea Ant 'Cup' para generar el código fuente de las clases que implementan el autómata del parser (`sym.java` y `parser.java`)
- Ejecutar la tarea de Ant 'jflex' para obtener en el paquete `compiler.lexical` la implementación del autómata `scanner.java` de acuerdo a la especificación en JFlex
- Compilar todo el proyecto para obtener en `/classes` el código binario del programa. Estos tres últimos puntos pueden realizarse de forma automática invocando a la tarea de Ant 'build'
- Ejecutar la tarea Ant 'flexTest' para ejecutar el programa `LexicalTestCase` sobre uno de los ficheros de código fuente contenidos dentro de `doc/test`. Recuerde que el nombre del fichero que se evalúa por defecto es `TestA` pero puede cambiarse editando el fichero `build.xml`. Esta tarea llama previamente a la tarea `build`, por lo que *englobaría los puntos anteriores*.

En esencia el código del programa de prueba `LexicalTestCase` va solicitando al `scanner` uno por uno los `TOKENS` del fichero de entrada y los va imprimiendo por la salida estándar haciendo uso de las facilidades de traza de la clase `LexicalErrorManager`. Aunque no es necesario comprender su funcionamiento para ejecutar las pruebas a continuación discutimos el funcionamiento de la clase `LexicalTestCase` que se ilustra en el listado 5.

#### Listado 5. Código de prueba de la clase `LexicalTestCase`

```
while (anObject instanceof Token)
{
    Token aToken = (Token) anObject;
    if (aToken.sym == sym.EOF) break;
    lexicalErrorManager.lexicalInfo (aToken);
}
```

```

        anObject = aScanner.next_token ();
    }

    lexicalErrorManager.lexicalInfo ("End of file.");

```

Como puede apreciarse mediante un bucle while se van solicitando uno por uno todos los tokens del fichero de entrada invocando al método miembro next\_token (). En cada iteración se imprime el Token por la salida estándar invocando al método LexicalInfo de la clase LexicalErrorManager y se solicita un nuevo Token hasta que se encuentre el final del fichero (identificada por la constante simbólica sym.EOF de la clase sym). El resultado de la ejecución de este programa sobre un código fuente debería ser el listado de todos los token encontrados por el scanner en el fichero.

### Pruebas del analizador sintáctico

Para evaluar el analizador sintáctico generado a través de la especificación de Cup deben seguirse los siguientes pasos:

- Escribir un fichero de código fuente en el lenguaje especificado en la práctica (correcto o incorrecto) y depositarlo dentro del directorio doc/test
- Escribir la especificación completa del analizador léxico completando la estructura proporcionada dentro del fichero doc/specs/scanner.flex de acuerdo a las directrices impuestas a lo largo de este documento
- Escribir la especificación completa del analizador sintáctico completando la estructura proporcionada dentro del fichero doc/specs/parser.cup de acuerdo a las directrices impuestas a lo largo de este documento
- Ejecutar la tarea de Ant 'jflex' para obtener en el paquete compiler.lexical la implementación del autómata scanner.java de acuerdo a la especificación en Jflex
- Ejecutar la tarea de Ant 'cup' para obtener en el paquete compiler.syntax la implementación del analizador sintáctico (sym.java y parser.java de acuerdo a la especificación en Cup)
- Compilar todo el proyecto para obtener en /classes el código binario del programa. Estos tres últimos puntos pueden realizarse de forma automática invocando a la tarea de Ant 'build'
- Ejecutar la tarea Ant 'cupTest' para ejecutar el programa SyntaxTestCase sobre uno de los ficheros de código fuente contenidos dentro de doc/test. Recuerde que el nombre del fichero que se evalúa por defecto es TestA pero puede cambiarse editando el fichero build.xml (consulte la sección 3.3.1). *Esta tarea llama previamente a la tarea build, por lo que englobaría los puntos anteriores.*

Como puede verse en el listado 6, el código del programa de prueba construye un stream de entrada conectado al fichero de código fuente cuyo nombre se pasa como parámetro en la variable 'fileName'. A partir de él construye el Scanner y el parser y arranca este último. Una vez arrancado el parser, éste irá solicitando al scanner los TOKENS de acuerdo a las reglas gramaticales para intentar construir un árbol de

análisis sintáctico. El resultado de esta ejecución será la emisión por la salida estándar de un mensaje 'Starting parsing...', seguido opcionalmente de una serie de mensajes de error y seguido del mensaje 'Parsing process finished'. El listado 6 ilustra el código de la clase `SyntaxTestCase`. Puede verse que la creación de los objetos `Scanner` y `parser` se hace mediante introspección, así como la invocación del método `parser.parse()`. No es necesario que el alumno entienda ni modifique esta estructura.

#### Listado 6. Código de prueba de la clase `SyntaxTestCase`

```
Constructor scannerConstructor =
scannerClass.getConstructor(Reader.class);

aScanner = (ScannerIF) scannerConstructor.newInstance(aFileReader);

// reflect parser

Class parserClass = Class.forName ("compiler.syntax.parser");

Constructor parserConstructor =
parserClass.getConstructor(java_cup.runtime.Scanner.class);

// reflect call parser.parse()

Method parseMethod = parserClass.getMethod("parse");

Object aParser = parserConstructor.newInstance(aScanner);

parseMethod.invoke(aParser);
```



**Adecuación de la segunda parte de las guías de estudio** de las asignaturas de Grado de la UNED a la metodología a distancia.

**PROTOCOLO DE REVISIÓN DE GUÍAS DE ESTUDIO DE EEES. Parte 2**

IDENTIFICACIÓN DE LA GUÍA DE ESTUDIO										
Título de Grado: Grado en Ingeniería Informática										
Facultad / Escuela: E.T.S de Ingeniería Informática										
Departamento:										
Asignatura: Procesadores del Lenguaje I								Código: 71013130		
Curso				Carácter			Duración			ECTS:
1º	2º	3º X	4º	FB	Ob X	Op	Anual	Sem 1º X	Sem 2º	6
Equipo Docente: Anselmo Peñas Padilla; Álvaro Rodrigo Yuste										

ESCALA DE VALORACIÓN						
(NF) <u>No figura</u> (1) <u>Insuficiente</u> (2) <u>Suficiente</u> (3) <u>Bien</u> (4) <u>Muy bien</u>						
(añádase el comentario que corresponda en la columna <b>Observaciones</b> , si procede)						
PLAN DE TRABAJO						
	NF	1	2	3	4	Observaciones
Se presenta una descripción ordenada y temporalmente secuenciada de las actividades que tendrá que realizar el estudiante a lo largo del curso.			X			
Se proporciona un cronograma claro para el seguimiento del curso.			X			
Se plantea un plan razonable, ajustado a los ECTS de la asignatura.					X	Excelente
ORIENTACIONES PARA EL ESTUDIO DE LOS TEMAS /BLOQUES TEMÁTICOS						
	NF	1	2	3	4	Observaciones
Los resultados de aprendizaje de los distintos temas/bloques temáticos están claramente definidos.			X			De forma general en GdE1 <sup>1</sup>
Se introduce de forma clara y ordenada la secuencia de contenidos que serán objeto de estudio en cada tema/bloque temático.	X					
Contextualiza los temas/bloques temáticos en el conjunto de la asignatura; proporciona información sobre la contribución específica al perfil profesional y al desarrollo de las competencias del título.			X			En GdE1
Refiere contenidos previos a repasar, cuando son necesarios.					X	En GdE1 y texto base

<sup>1</sup> Con relación a los objetivos y los resultados de aprendizaje, ver documento redactado por ANECA: Guía de apoyo para la redacción, puesta en práctica y evaluación de los resultados del aprendizaje. Versión 1.0. (pp. 20-21). <http://www.aneca.es/Sala-de-prensa/Noticias/2013/ANECA-presenta-la-Guia-para-la-redaccion-y-evaluacion-de-los-resultados-del-aprendizaje>



Proporciona orientaciones específicas para el estudio de los temas o bloques temáticos, cuando se estima necesario.			X			En <<Prefacio>> del texto base
Prioriza los aspectos más relevantes de cada tema/bloque temático.				X		En texto base
Señala los contenidos de mayor nivel de dificultad.	X					
Explica aquellos contenidos de mayor dificultad, facilitando la comprensión y estudio del texto, cuando es necesario.	X					
Proporciona elementos facilitadores del aprendizaje como elementos gráficos de apoyo (textos, mapas, cuadros, figuras, etc.), especialmente cuando el texto no los incorpora.	-	-	-	-	-	No procede
Proporciona pruebas de autoevaluación de asimilación de contenidos teóricos o refiere cómo utilizar estas pruebas si están disponibles en el curso virtual, especialmente cuando el texto no los incorpora.					X	Remite al Curso Virtual
Las pruebas de autoevaluación disponibles en la guía, o en el curso virtual, disponen de solucionario.					X	En el Curso Virtual
Proporciona información clara acerca de cómo utilizar la bibliografía básica.				X		En <<Prefacio>>
Proporciona información clara acerca de cómo utilizar la bibliografía complementaria.				X		En GdE1 bibliografía complementaria comentada.
Proporciona información clara acerca de cómo utilizar el curso virtual para el seguimiento de la asignatura y la resolución de dudas y dificultades.	X					
Sugiere actividades complementarias que ayudan a consolidar los conocimientos.	X					
<b>ORIENTACIONES SOBRE LA REALIZACIÓN DE LAS ACTIVIDADES</b>						
	NF	1	2	3	4	Observaciones
Las actividades de aprendizaje son coherentes con los resultados de aprendizaje planteados en la primera parte de la guía de estudio.				X		
Las actividades planteadas son variadas, en función de los resultados que los estudiantes tengan que alcanzar.				X		Práctica y Prueba Presencial
Se informa al estudiante sobre los objetivos y la utilidad de la realización de cada actividad para la formación del estudiante.	X					
Los requisitos de las actividades son razonables en cuanto a la estimación del tiempo y nivel requeridos.					X	
Las actividades se describen de forma clara en cuanto a procedimientos y secuencia de realización.		X				Únicamente se contempla como actividad la Parte Práctica. A falta de detallar información referente a la Prueba Presencial.
Se ofrecen indicadores y criterios claros sobre los procedimientos de evaluación y la corrección de cada una de las actividades.				X		A falta de especificar criterios de evaluación <sup>2</sup> . Los criterios de calificación vienen expuestos en la GdE1
Se proporcionan todos los recursos necesarios para la realización de las actividades.				X		
Se informa sobre el papel del tutor en las actividades en que interviene, informando al estudiante sobre los procedimientos de seguimiento de las actividades.	X					
Hay actividades dirigidas a preparar mejor las pruebas presenciales.	X					No se hace mención expresa
Se proporcionan procedimientos de autoevaluación cuando el tipo de prueba lo permite.					X	Remiten al Curso Virtual

<sup>2</sup> Con *criterios de evaluación*, a diferencia de los de calificación, hacemos referencia a aquellos criterios, indicios o indicadores que orientan cualitativamente la calificación final y les hará obtener una calificación numérica o apto-no apto.



El procedimiento de evaluación de las actividades permite valorar la adquisición de los resultados de aprendizaje.				X	
<b>REDACCIÓN</b>					
(1) <u>No adecuado</u> (2) <u>Adecuado (recom.)</u> (3) <u>Bien</u> (4) <u>Excelente</u>					
(añádase el comentario que corresponda en la columna <i>Observaciones</i> , si procede)					
	1	2	3	4	Observaciones
Claridad de la redacción.				X	
Extensión adecuada de los párrafos.				X	
Exposición ordenada de ideas.				X	
Ordenación clara de los apartados.				X	
Ortografía y signos de puntuación (acentuación, puntuación, siglas, símbolos, abreviaturas, mayúsculas, cursiva, etc.).				X	

<b>COMPLEMENTO DEL TEXTO (cuando se señala en la revisión del texto)</b>						
	NF	1	2	3	4	Observaciones
La guía complementa adecuadamente los apartados correspondientes del texto, que era necesario complementar según la revisión efectuada.		X				Se necesitan reforzar las orientaciones para el aprendizaje.
Incluye un glosario adecuado de términos, o informa de que se encuentra disponible en el curso virtual.	X					

<b>VALORACIÓN GENERAL DE LA GUÍA. Parte 2</b>					
	1	2	3	4	Observaciones
Plan de trabajo			X		Recomendado incluir un cuadro-cronograma donde se especifique la entrega de la actividad práctica y la Prueba Presencial. Igualmente es importante contemplar la semana de vacaciones en la planificación del estudio del contenido teórico.
Orientaciones para el estudio de los temas		X			Recomendado ampliar este apartado e incluir en cada tema o bloque temático orientaciones para el aprendizaje (además de los ejercicios de autoevaluación): - <u>Introducción.</u> - <u>Resultados de aprendizaje</u> - <u>Contextualización:</u> respondiendo a las preguntas: <i>¿Qué papel hace este tema en el conjunto de la asignatura? ¿Qué aportará este tema al estudiante como futuro profesional?</i> -¿Dónde encontrarán el <u>material requerido</u> para el estudio de cada tema? - <u>Orientaciones para su estudio</u> (entre otras: contenidos previos a repasar, contenidos fundamentales, contenidos de mayor dificultad, estrategias recomendadas). - <u>Actividades complementarias.</u>



Orientaciones para la realización de las actividades		X		<p>Recomendado en este epígrafe incluir tanto para la Parte Práctica como la Prueba Presencial, los siguientes apartados:</p> <ul style="list-style-type: none"> <li>• objetivos,</li> <li>• orientaciones para realizarla,</li> <li>• tiempo estimado de realización (o especificarlo en el cronograma),</li> <li>• plazo de entrega,</li> <li>• y especialmente recomendado: criterios de evaluación que se van a seguir para evaluar cada actividad, especialmente en las actividades que puntúen para nota final.</li> </ul>
Glosario	X			Recomendado incluirlo, remitir al Curso Virtual o justificar su ausencia.

### VALORACIÓN GLOBAL

Excelente	
Adecuada y sin modificaciones	
Con propuestas de mejora	X
Con modificaciones necesarias	

## **Comentario final**

*(Añádase tantas líneas como sean necesarias)*

El objetivo de la Guía de Estudio Parte 2 es presentarles el plan de trabajo de la asignatura y las orientaciones para su desarrollo.

En el apartado de planificación o plan de trabajo encontrarán todas las actividades previstas durante el curso (trabajos, lecturas, prácticas, pruebas presenciales, entre otras) y su temporalización durante el semestre, ajustándolo explícitamente a los créditos ECTS.

En las orientaciones encontrarán información referente al estudio de los contenidos y también orientaciones referentes al desarrollo de las actividades, incluyendo la Prueba Presencial como una actividad más de aprendizaje.

Con respecto a las orientaciones para las actividades es importante proporcionar información referente a la Prueba Presencial (PP). Igualmente es importante proporcionar los criterios de evaluación<sup>3</sup> tanto de la Parte Práctica como de la PP antes de la realización de los mismos, para que los estudiantes puedan anticipar qué se espera de ellos.

Es por ello que se recomienda la mejora de los apartados de esta guía y atender a las observaciones especificadas en la valoración general.

**IUED, a 15 de junio del 2016**

**VºBº Comisión y fecha**

---

<sup>3</sup> Hacemos referencia aquí lo explicado en nota al pie de página 2.