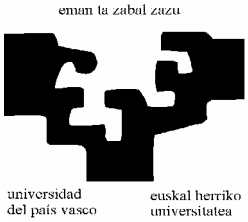
FACULTAD DE INFORMÁTICA
Lenguajes y Sistemas Informáticos



INFORMATIKA FAKULTATEA
Lengoaia eta Sistema Informatikoak

EVALUACIÓN EMPÍRICA DE LA COMPRENSIÓN DEL MODELADO DINÁMICO EN LOS LENGUAJES UML Y OML DE APLICACIONES SOFTWARE

por

Dña. Mari Carmen Otero Vidal

Memoria presentada para optar al Título de Doctora en Informática
realizada bajo la dirección del profesor
Dr. D. José Javier Dolado Cosín

Donostia, Febrero de 2003

Dedicado al cariñoso recuerdo de mi padre,
Manuel Otero Otero

AGRADECIMIENTOS

La culminación de esta tesis no habría sido posible sin el constante apoyo que me han demostrado las siguientes personas:

- Javier Dolado, un amigo espléndido y un director excelente, con quien me inicié, he aprendido y deseo sinceramente seguir colaborando en esta aventura de la investigación. Muchas gracias por todos tus ánimos y tus valiosos consejos en los momentos decisivos.

- Aitor, mi querido hijo, a quien a partir de ahora espero poder dedicarle más tiempo. Gracias por ser mi otro gran proyecto en la vida y así equilibrar la balanza entre familia y trabajo.

- Maruja, mi madre y confidente, quien permaneciendo en un segundo plano me ha proporcionado la tranquilidad necesaria para que yo pudiese centrarme en la tesis. Muchas gracias por tu paciencia, tu comprensión y tu ayuda en la crianza de Aitor.

- José Manuel, mi hermano, a quien admiro por la serenidad que siempre me ha transmitido. Gracias por confiar en mí.

- Mis compañeros del departamento de Lenguajes y Sistemas Informáticos y también de los proyectos REMIS y ARGO, con quienes he podido compartir mis dudas e inquietudes profesionales. Gracias por vuestras interesantes recomendaciones y comentarios.

AGRADECIMIENTOS INSTITUCIONALES

El trabajo de investigación de esta tesis se ha desarrollado con el apoyo de los proyectos:

- CICYT TIC98-1179-E,
- UPV-EHU 141.226-EA083/98,
- UPV-EHU 141.226-EA8099/2000.

Actualmente se encuentra financiado por el siguiente proyecto:

- CICYT TIC2001-1143-C03-01.

Índice

INTRODUCCIÓN 1

PARTE I: MODELADO EN INGENIERÍA DEL SOFTWARE

CAPÍTULO 1

LENGUAJES DE MODELADO EN LA INGENIERÍA DEL SOFTWARE 3

1.1. METODOLOGÍAS DE DESARROLLO DE SOFTWARE 3

1.2. LENGUAJES DE MODELADO 5

1.2.1. El lenguaje unificado de modelado (UML) 5

1.2.1.1. Vistas de UML 6

1.2.1.2. Diagramas de UML 7

1.2.1.2.1. Diagrama de clases 8

1.2.1.2.2. Diagrama de secuencia 8

1.2.1.2.3. Diagrama de colaboración 11

1.2.1.2.4. Diagrama de estado 13

1.2.2. El lenguaje de modelado de OPEN (OML) 15

1.2.2.1. Nodos comunes 17

1.2.2.2. Relaciones comunes 17

1.2.2.3. Diagramas de OML 18

1.2.2.3.1. Diagrama de colaboración de paquete 19

1.2.2.3.2. Diagrama de secuencia de caja blanca 20

1.2.2.3.3. Diagrama de colaboración interna 21

1.2.2.3.4. Diagrama de transición de estado 22

1.3. MÉTRICA Y LOS LENGUAJES DE MODELADO ORIENTADOS A OBJETOS 23

PARTE II: ESTADO DE LA CUESTIÓN

CAPÍTULO 2

EL PARADIGMA EXPERIMENTAL EN LA INGENIERÍA DEL SOFTWARE 25

2.1. MÉTODOS DE INVESTIGACIÓN EN LA INGENIERÍA DEL SOFTWARE 25

2.1.1. Métodos empíricos 26

2.1.2. Necesidad de experimentos 26

2.2. EXPERIMENTACIÓN EN LA INGENIERÍA DEL SOFTWARE 27

2.2.1. Variables 28

2.2.2. Tratamientos, unidades y sujetos 29

2.3. PROCESO EXPERIMENTAL 29

2.3.1. Definición del objetivo 31

2.3.1.1. Método GQM 31

2.3.2. Planificación 32

2.3.2.1. Selección de las variables 32

2.3.2.2. Planteamiento de las hipótesis 32

2.3.2.3. Sujetos experimentales 33

2.3.2.4. Objetos experimentales 35

2.3.2.5. Diseño experimental 35

2.3.2.5.1. Principios de diseño 36

2.3.2.5.2. Tipos de diseño 38

2.3.2.6. Validez experimental 38

2.3.3. Ejecución 40

2.3.4. Análisis de los datos 41

2.3.5. Interpretación 46

2.3.6. Presentación 48

CAPÍTULO 3

SITUACIÓN DE LA PRÁCTICA EXPERIMENTAL EN UML 49

3.1. INTRODUCCIÓN 49

3.2. ESTADO DE LA PRÁCTICA EMPÍRICA DE LOS ARTEFACTOS CONSTRUIDOS CON UML 50

3.3. MOTIVACIÓN Y CONTEXTO EXPERIMENTAL DE ESTA INVESTIGACIÓN 53

PARTE III: EVALUACIÓN DEL MODELADO DINÁMICO

CAPÍTULO 4

PLANIFICACIÓN EXPERIMENTAL PARA LA EVALUACIÓN DEL MODELADO DINÁMICO ... 55

4.1. INTRODUCCIÓN ... 55
4.2. OBJETIVO GENERAL DE LA INVESTIGACIÓN ... 55
4.3. PLANIFICACIÓN EXPERIMENTAL ... 57
4.4. PRINCIPIOS DE DISEÑO EXPERIMENTAL ... 58
4.5. TIPOS DE DISEÑO EXPERIMENTAL EMPLEADOS EN ESTA INVESTIGACIÓN ... 58
4.5.1. Diseño cross-over ... 59
4.5.1.1. Diseño cross-over de 2 tratamientos: CO-2 ... 60
4.5.1.1.1. Diagrama de CO-2 ... 60
4.5.1.1.2. Modelo Lineal Estadístico de CO-2 ... 61
4.5.1.2. Diseño cross-over de 3 tratamientos: CO-3 ... 62
4.5.1.2.1. Diagrama de CO-3 ... 62
4.5.2. Diseño factorial confundido ... 63
4.5.2.1. Diseño factorial confundido de cuadrado latino ... 64
4.5.2.1.1. Diagrama de LSCF-3^2 ... 64
4.5.2.1.2. Modelo Lineal Estadístico de LSCF-3^2 ... 66
4.5.2.2. Diseño factorial de parcelas divididas ... 67
4.5.2.2.1. Diagrama de SPF-3.3 ... 67
4.5.2.2.2. Modelo Lineal Estadístico de SPF-3.3 ... 68

CAPÍTULO 5

EVALUACIÓN INICIAL DEL MODELADO DINÁMICO EN UML ... 71

5.1. INTRODUCCIÓN ... 71
5.2. DEFINICIÓN DEL OBJETIVO DEL PRIMER EXPERIMENTO ... 71
5.2.1. Objetivo ... 71
5.3. PLANIFICACIÓN DEL PRIMER EXPERIMENTO ... 72
5.3.1. Variables experimentales ... 72
5.3.2. Hipótesis nulas ... 72
5.3.3. Sujetos experimentales ... 73
5.3.4. Material experimental ... 73
5.3.5. Diseño experimental ... 74
5.3.6. Validez experimental ... 76
5.3.6.1. Validez constructiva ... 76
5.3.6.2. Validez interna ... 76
5.3.6.3. Validez externa ... 77
5.4. EJECUCIÓN DEL PRIMER EXPERIMENTO ... 77

5.5. ANÁLISIS DE LOS DATOS 78
5.5.1. Primer análisis de varianza 78
5.5.1.1. Variable dependiente: TSEC 79
5.5.1.2. Variable dependiente: NRESP 80
5.5.2. Segundo análisis de varianza 80
5.5.2.1. Variable dependiente: TSEC 80
5.5.2.2. Variable dependiente: NRESP 82
5.6. INTERPRETACIÓN DE LOS RESULTADOS 83
5.7. PRESENTACIÓN 86

CAPÍTULO 6

RÉPLICA DEL MODELADO DINÁMICO EN UML 87

6.1. INTRODUCCIÓN 87
6.2. DEFINICIÓN DE LA RÉPLICA Y DEL TERCER EXPERIMENTO 88
6.2.1. Hipótesis de alto nivel 88
6.3. PLANIFICACIÓN DE LA RÉPLICA 88
6.3.1. Variables experimentales 89
6.3.2. Hipótesis nulas concretas derivadas de la Hipótesis nula 1 89
6.3.3. Sujetos experimentales 90
6.3.4. Material experimental 90
6.3.4.1. Documentos de diseño UML 91
6.3.5. Diseño experimental 93
6.3.6. Validez experimental 94
6.3.6.1. Validez interna 95
6.3.6.2. Validez de la conclusión 96
6.3.7. Estimación del tamaño de la muestra de la réplica 96
6.4. EJECUCIÓN DE LA RÉPLICA 99
6.5. ANÁLISIS DE LOS DATOS 100
6.5.1. Estrategia del análisis de varianza 100
6.5.2. Análisis de varianza completo 102
6.5.2.1. Variable dependiente: TSEC 102
6.5.2.2. Variable dependiente: NRESP 103
6.5.3. Análisis de varianza reducido 104
6.5.3.1. Variable dependiente: TSEC 104
6.5.3.2. Variable dependiente: NRESP 105
6.6. INTERPRETACIÓN DE LOS RESULTADOS 106
6.6.1. Interpretación del tiempo total 106
6.6.2. Interpretación de la puntuación total 108
6.7. PRESENTACIÓN 112

CAPÍTULO 7

EVALUACIÓN COMPLEMENTARIA DEL MODELADO DINÁMICO EN UML 113

7.1. INTRODUCCIÓN 113
7.2. DEFINICIÓN DEL TERCER EXPERIMENTO 113
7.2.1. Hipótesis nula 2 de alto nivel 113
7.3. PLANIFICACIÓN DEL TERCER EXPERIMENTO 114
7.3.1. Variables experimentales 114
7.3.2. Hipótesis nulas concretas derivadas de la Hipótesis nula 2 114
7.3.3. Sujetos experimentales 115
7.3.4. Material experimental 115
7.3.5. Diseño experimental 117
7.3.6. Validez experimental 118
7.3.6.1. Validez interna 118
7.4. EJECUCIÓN DEL TERCER EXPERIMENTO 119
7.5. ANÁLISIS DE LOS DATOS 120
7.5.1. Estrategia del análisis de varianza 120
7.5.2. Análisis de varianza para TSEC y NRESP 120
7.6. INTERPRETACIÓN DE LOS RESULTADOS 121
7.7. PRESENTACIÓN 124

CAPÍTULO 8

OML Y UML 125

8.1. INTRODUCCIÓN A OML 125
8.1.1. Diagramas de modelado en los lenguajes OML y UML 125
8.2. DEFINICIÓN DEL OBJETIVO DEL CUARTO EXPERIMENTO 127
8.2.1. Objetivo 127
8.3. PLANIFICACIÓN DEL CUARTO EXPERIMENTO 127
8.3.1. Hipótesis nula de alto nivel 127
8.3.2. Variables experimentales 128
8.3.3. Sujetos experimentales 128
8.3.4. Material experimental 129
8.3.5. Diseño experimental 129
8.3.6. Validez experimental 131
8.3.6.1. Validez interna 131
8.3.6.2. Validez externa 132
8.4. EJECUCIÓN DEL CUARTO EXPERIMENTO 132
8.5. ANÁLISIS DE LOS DATOS 133
8.5.1. Estrategia del análisis de varianza 133
8.5.2. Análisis de varianza para contrastar la hipótesis $H_{0\text{-NOTATION}}$ 135

8.5.3. Análisis de varianza para contrastar la hipótesis $H_{0\text{-STATE}}$ 136
8.5.4. Análisis de varianza para contrastar la hipótesis $H_{0\text{-SEQUENCE}}$ 137
8.6. INTERPRETACIÓN DE LOS RESULTADOS 138
8.7. PRESENTACIÓN 140

CAPÍTULO 9
RESUMEN DE LOS RESULTADOS 141
9.1. INTRODUCCIÓN 141
9.2. COMPENDIO DE LOS DATOS DE UML PARA EL TIEMPO TOTAL 142
9.3. COMPENDIO DE LOS DATOS DE UML PARA LA PUNTUACIÓN TOTAL 144
9.4. EFICACIA DE LA CAPACIDAD SEMÁNTICA DEL MODELADO DINÁMICO EN UML 147
9.5. EFICACIA DEL MODELADO DINÁMICO EN OML Y UML 149

PARTE IV: CONCLUSIONES

CAPÍTULO 10
CONCLUSIONES 151
10.1. INTRODUCCIÓN 151
10.2. APORTACIONES 152
10.3. CONCLUSIONES 153
10.4. FUTURAS LÍNEAS DE INVESTIGACIÓN 155

BIBLIOGRAFÍA

REFERENCIAS BIBLIOGRÁFICAS 157

ACRÓNIMOS

GLOSARIO DE ACRÓNIMOS 169

APÉNDICES

APÉNDICE A
MATERIAL PARA LA REALIZACIÓN DEL PRIMER EXPERIMENTO A-1
A.1. INSTRUCCIONES PARA EL PRIMER EXPERIMENTO A-1
A.2. TELÉFONO MÓVIL: DIAGRAMAS UML Y CUESTIONARIO A-2
A.3. SISTEMA DE BIBLIOTECA: DIAGRAMAS UML Y CUESTIONARIO A-10
A.4. DICTÁFONO: DIAGRAMAS UML Y CUESTIONARIO A-18

APÉNDICE B

MATERIAL PARA LA REALIZACIÓN DEL SEGUNDO (RÉPLICA) Y TERCER EXPERIMENTO B-1

B.1. INSTRUMENTOS PARA LA CLASIFICACIÓN Y EL ENTRENAMIENTO DE LOS SUJETOS B-1

B.1.1. Cuestionario sobre UML B-1

B.1.2. Máquina de Café: Diagramas UML y Cuestionario B-5

B.2. APLICACIÓN DE UN SISTEMA DE SEGUROS: DIAGRAMAS UML Y CUESTIONARIO B-11

B.3. APLICACIÓN DE UN PARKING: DIAGRAMAS UML Y CUESTIONARIO B-31

APÉNDICE C

MATERIAL PARA LA REALIZACIÓN DEL CUARTO EXPERIMENTO C-1

C.1. INSTRUCCIONES PARA EL CUARTO EXPERIMENTO C-1

C.2. APLICACIÓN DE UNA MÁQUINA EXPENDEDORA DE ARTÍCULOS C-2

C.2.1. Diagramas UML C-2

C.2.2. Diagramas OML C-14

C.2.3. Cuestionario desde la perspectiva del responsable de mantenimiento C-24

C.2.4. Cuestionario desde la perspectiva del cliente C-29

APÉNDICE D

RESULTADOS CORRESPONDIENTES AL PRIMER EXPERIMENTO D-1

D.1. FICHERO DE DATOS D-1

D.2. SINTAXIS PARA LOS DATOS EN SPSS D-3

D.3. SCRIPT PARA LOS DATOS EN JMP D-4

D.4. PLANTILLAS PARA LOS DATOS EN NCSS D-4

APÉNDICE E

RESULTADOS CORRESPONDIENTES A LA RÉPLICA E-1

E.1. GRUPOS EMPAREJADOS: PROCEDIMIENTO Y RESULTADOS E-1

E.2. FICHERO DE DATOS CODIFICADOS PARA EL MODELO ANOVA COMPLETO E-3

E.3. SINTAXIS PARA LOS DATOS CON EFECTOS DE CARRYOVER EN SPSS E-6

E.4. FICHERO DE DATOS CODIFICADOS PARA EL MODELO ANOVA REDUCIDO E-6

E.5. SINTAXIS PARA LOS DATOS SIN EFECTOS DE CARRYOVER EN SPSS E-8

E.6. SCRIPT PARA LOS DATOS SIN EFECTOS DE CARRYOVER EN JMP E-9

E.7. PLANTILLA PARA LOS DATOS SIN EFECTOS DE CARRYOVER EN NCSS E-9

APÉNDICE F

RESULTADOS CORRESPONDIENTES AL TERCER EXPERIMENTO F-1

F.1. FICHERO DE DATOS F-1

F.2. SINTAXIS PARA LOS DATOS EN SPSS F-2

F.3. SCRIPT PARA LOS DATOS EN JMP F-2

F.4. PLANTILLA PARA LOS DATOS EN NCSS F-3

APÉNDICE G

RESULTADOS CORRESPONDIENTES AL CUARTO EXPERIMENTO G-1

G.1. FICHERO DE DATOS G-1

G.2. SINTAXIS PARA LOS DATOS EN SPSS G-3

G.3. PLANTILLA PARA LOS DATOS EN NCSS G-5

Lista de figuras

Figura 1.1. Elementos clave en ingeniería del software 3
Figura 1.2. Enfoques y elementos implicados en el desarrollo de UML 6
Figura 1.3. Diagrama de clases del termostato digital 8
Figura 1.4. Diagrama de secuencia para Pedir Artículos 9
Figura 1.5. Ejemplo de un diagrama de secuencia para Controlar la Temperatura 11
Figura 1.6. Diagrama de colaboración para Pedir Artículos 12
Figura 1.7. Ejemplo de un diagrama de colaboración para Controlar la Temperatura 12
Figura 1.8. Diagrama de estado para Petición de Artículos 14
Figura 1.9. Ejemplo de un diagrama de estado para la instancia de la clase Habitación 15
Figura 1.10. Métodos implicados en la configuración de OPEN 16
Figura 1.11. Componentes de OML 17
Figura1.12. Metamodelo para CIRT 18
Figura 1.13. Metamodelo para las relaciones 19
Figura 1.14. Diagrama de paquete del termostato digital 20
Figura 1.15. Diagrama de colaboración de paquete del termostato digital 21
Figura 1.16. Ejemplo de un diagrama de secuencia de caja blanca para Controlar la Temperatura ... 22
Figura 1.17. Ejemplo de un diagrama de colaboración interna para la instancia de la clase Habitación 23
Figura 1.18. Ejemplo de un diagrama de transición de estado para la instancia de la clase Habitación 24
Figura 2.1. Variables de un experimento controlado 28
Figura 2.2. Proceso experimental 30
Figura 2.3. Regiones correspondientes a un error de tipo I (α) y a un error de tipo II (β) 43
Figura 2.4. Diagrama de dispersión de las puntuaciones y tiempos del primer experimento 46
Figura 2.5. Diagrama de caja con respecto al tiempo 47
Figura 4.1. Proceso de planificación experimental para la evaluación del modelado dinámico 57
Figura 5.1. Gráfico de perfil del tiempo total (en segundos) 83

Figura 5.2. Gráfico de perfil de la puntuación total (5 puntos máximo) 84
Figura 6.1. Relación entre los factores de la réplica 94
Figura 6.2. Gráfico de potencia versus tamaño muestral por grupo (n) para TSEC 97
Figura 6.3. Diagramas de barras de la variable TSEC: a) experimento original b) réplica 99
Figura 6.4. Diagramas de barras de la variable NRESP: a) experimento original b) réplica 99
Figura 7.1. Gráficos de perfil correspondientes al tiempo (TSEC) y a la puntuación (NRESP) 122
Figura 8.1. Diseño factorial cross-over 2x2 130
Figura 8.2. Gráficos de caja para cada tipo de diagrama UML y OML con respecto al tiempo 138
Figura 8.3. Gráficos de caja para cada tipo de diagrama con respecto a la puntuación 139
Figura 9.1. Gráfico de líneas para el tipo de diagrama y el dominio de aplicación en cada experimento con respecto al tiempo total 143
Figura 9.2. Gráfico de líneas para el tipo de diagrama y el dominio de aplicación en cada experimento con respecto a la puntuación total 145
Figura 9.3. Porcentaje de acierto para aplicaciones de gestión 146
Figura 9.4. Gráfico de líneas para el tipo de diagrama y el dominio de aplicación con respecto a la eficacia 148
Figura 9.5. Gráfico de barras de la eficacia en la comprensión de OML y UML 150

Lista de tablas

Tabla 1.1. Vistas, diagramas y conceptos de UML 7
Tabla 1.2. Diagramas de OML 19
Tabla 2.1. Factores para seleccionar un método empírico 26
Tabla 2.2. Ejemplos de los parámetros de la plantilla GQM 32
Tabla 2.3. Diferentes tipos de decisión para las situaciones de μ = 50 y μ = 55,5 42
Tabla 3.1. Artefactos de UML utilizados como material en las referencias citadas 52
Tabla 4.1. Formulación del objetivo general de este trabajo de investigación 56
Tabla 4.2. Diseño cross-over de 2 tratamientos o CO-2 60
Tabla 4.3. Diseño cross-over de 3 tratamientos o CO-3 62
Tabla 4.4. Diseño factorial confundido de cuadrado latino o LSCF-3^2 64
Tabla 4.5. Diseño factorial de parcelas divididas o SPF-3.3 67
Tabla 5.1. Hipótesis nulas del primer experimento 73
Tabla 5.2. Diseño factorial confundido cuadrado latino 3^2 75
Tabla 5.3. Primer ANOVA para la variable TSEC (Tiempo total) 79
Tabla 5.4. Primer ANOVA para la variable NRESP (Puntuación total) 80
Tabla 5.5. Segundo ANOVA para la variable TSEC (Tiempo total) 81
Tabla 5.6. Efectos simples de TSEC para el factor APPLICATION 81
Tabla 5.7. Segundo ANOVA para la variable NRESP (Puntuación total) 82
Tabla 5.8. Efectos simples de NRESP para el factor APPLICATION 82
Tabla 5.9. Resumen estadístico del primer experimento con respecto al tiempo total (mm:ss) 83
Tabla 5.10. Resumen estadístico del primer experimento con respecto a la puntuación total 84
Tabla 6.1. Restricción impuesta al segundo y tercer experimento 88
Tabla 6.2. Hipótesis nulas concretas del segundo experimento (réplica) 90
Tabla 6.3. Métricas correspondientes a cada escenario de nuestros diseños 92
Tabla 6.4. Diseño factorial split-plot 36.3 93
Tabla 6.5 Tabla de potencia versus n para NRESP (α = 0,1 y SD = 1) 98

Tabla 6.6. Resultados del contraste sobre el supuesto de esfericidad 99
Tabla 6.7. Resultados SPSS del ANOVA completo para la variable TSEC según [Mill92] 99
Tabla 6.8. Resultados SPSS del ANOVA completo para la variable NRESP según [Ratk93] 99
Tabla 6.9. Resultados del modelo ANOVA reducido con respecto a TSEC utilizando JMP 99
Tabla 6.10. Resultados del modelo ANOVA reducido con respecto a NRESP utilizando NCSS 99
Tabla 6.11. Resumen estadístico del tiempo total (mm:ss) para ambos experimentos 99
Tabla 6.12. Resumen estadístico de la puntuación total (5 puntos) para ambos experimentos 99
Tabla 7.1. Hipótesis nulas concretas del tercer experimento 114
Tabla 7.2. Métricas de tamaño correspondientes al diseño de PARKING 116
Tabla 7.3. Métricas de tamaño correspondientes al diseño de SEGUROS 116
Tabla 7.4. Diseño factorial split-plot 3.2 117
Tabla 7.5. Resultados SPSS para las variables TSEC y NRESP 121
Tabla 8.1. Comparación de tipos principales de diagramas (UML y COMN) 126
Tabla 8.2. Equivalencia entre los modelos dinámicos de UML y OML a estudiar 126
Tabla 8.3. Equivalencias en la terminología de tres modelos para el diseño crossover 2x2 134
Tabla 8.4. Hipótesis nulas concretas del cuarto experimento 135
Tabla 8.5. Análisis de varianza para las variables TSEC y NRESP 136
Tabla 8.6. Comparaciones múltiples para contrastar la hipótesis $H_{0\text{-STATE}}$ 137
Tabla 8.7. Comparaciones múltiples para contrastar la hipótesis $H_{0\text{-SEQUENCE}}$ 137
Tabla 8.8. Estadísticos descriptivos del tiempo (media - desviación típica) de verificación 139
Tabla 9.1. Síntesis de la eficacia en la comprensión para cada diagrama y dominio de aplicación ... 149

Introducción

Estructura de la tesis

Esta tesis está organizada en cuatro partes, además de las referencias bibliográficas, de los acrónimos y de los apéndices. Cada una de estas partes, a su vez, se divide en uno o varios capítulos:

PARTE I: MODELADO EN INGENIERÍA DEL SOFTWARE

En el Capítulo 1 se ofrece una versión reducida de dos lenguajes estándares de modelado orientado a objetos (UML y OML), haciendo especial hincapié en los tipos de diagramas objeto de estudio y utilizados como material experimental en esta investigación.

PARTE II: ESTADO DE LA CUESTIÓN

En el Capítulo 2 se trata en primer lugar la situación actual de la experimentación en el contexto de la ingeniería del software. Asimismo se describen todos los elementos y los pasos necesarios para abordar de forma rigurosa una investigación empírica en esta disciplina.

En el Capítulo 3 se concreta cuál es el estado de la práctica experimental en UML, así como la motivación de nuestra investigación.

PARTE III: EVALUACIÓN DEL MODELADO DINÁMICO

En el Capítulo 4 se plantea primero el objetivo general de esta tesis. A continuación, se especifica todo el proceso empírico planificado para llevarlo a cabo, explicando tanto los principios como los tipos de diseño experimental utilizados en esta investigación.

En los siguientes Capítulos 5, 6, 7 y 8 se describen respectivamente desde el primer hasta el cuarto experimento que engloban esta tesis.

En el Capítulo 9 se trata de sintetizar todos los resultados obtenidos en los estudios empíricos de los capítulos anteriores.

PARTE IV: CONCLUSIONES

Por último, en el Capítulo 10 se presentan las conclusiones y sugerencias para futuros trabajos.

BIBLIOGRAFÍA

ACRÓNIMOS

APÉNDICES

En el Apéndice A se especifica el material experimental correspondiente al primer y segundo (réplica del primero) estudio empírico, y en los Apéndices B y C se describe el material del tercer y cuarto experimento, respectivamente.

En los Apéndices D, E, F y G se presentan los datos recopilados pertenecientes a cada uno de los cuatro experimentos, así como la sintaxis de acuerdo a los tres paquetes estadísticos utilizados para el análisis de dichos datos: JMP v.4, NCSS 2000 y SPSS v.10.

Para finalizar, tanto el material experimental como los datos recogidos se encuentran disponibles para su replicación en páginas web, cuyas direcciones se indican en los capítulos que describen cada uno de estos experimentos controlados.

PARTE I

MODELADO EN INGENIERÍA DEL SOFTWARE

Capítulo 1

Lenguajes de modelado en la ingeniería del software

1.1. Metodologías de desarrollo de software

La ingeniería del software surgió como disciplina a partir de la "Crisis del Software" de los años 70. Esta disciplina estudia los principios y las metodologías para el desarrollo y mantenimiento de proyectos software, y se concibe como una tecnología multicapas tal como muestra la Figura 1.1 [Ghez91]. Un método es un enfoque técnico que se puede utilizar en toda o parte de una metodología. Una metodología es un conjunto de procedimientos, técnicas y herramientas que ayuda a los desarrolladores a construir nuevo software. Tanto los métodos como las metodologías se basan en técnicas que suelen incorporar notaciones gráficas y/o textuales pertenecientes a un lenguaje determinado. Las herramientas suministran un soporte automático para las técnicas.

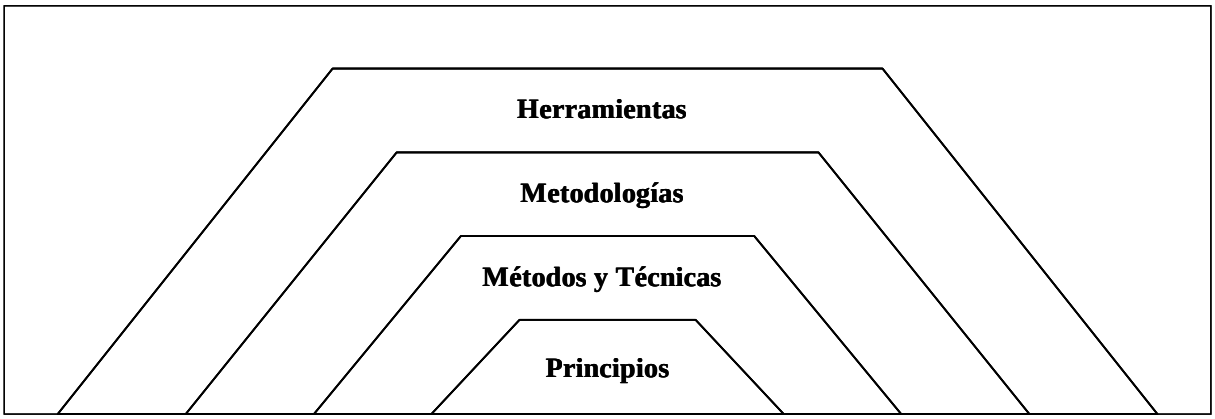


Figura 1.1. Elementos clave en ingeniería del software

En la actualidad en la ingeniería del software existe una gran variedad de enfoques y metodologías para el desarrollo de software. Algunos enfoques de proceso para el desarrollo

de software son: desarrollo en cascada, desarrollo con prototipado rápido, desarrollo en espiral, desarrollo incremental, etc. Las metodologías se apoyan en algún enfoque o combinación de ellos. Tanto a nivel comercial como en el ámbito académico y de investigación existe un gran número de metodologías, que se pueden agrupar en: metodologías estructuradas y metodologías orientadas a objetos.

Las metodologías estructuradas aparecieron a finales de los 60 con la programación estructurada y se extendieron con el diseño estructurado y, posteriormente, con el análisis estructurado a finales de los 70. Algunos ejemplos de metodologías estructuradas en el ámbito académico y comercial son: Gane & Sarson, Ward & Mellor, Yourdon & DeMarco, etc.

La historia de las metodologías orientadas a objetos va ligada a la evolución de los lenguajes de programación orientada a objetos. Algunos lenguajes representativos son: SIMULA a finales de los 60, Smalltak-80 a finales de los 70, la primera versión de C++ de Bjarne Stroustrup en 1981 y actualmente Java. Algunos ejemplos de metodologías Orientadas a Objetos (OO) son: Coad & Yourdon, Martín & Odell, Shaler & Mellor, OOAD (*Object-Oriented Analysis and Design*) de Booch [Booc94], OOSE (*Object-Oriented Software Engineering*, también denominado *Objectory*) de Jacobson [Jaco92] y OMT (*Object Modeling Technique*) de Rumbaugh [Rumb91].

Dentro del desarrollo orientado a objetos, a principios de los 90 existían demasiados lenguajes y enfoques diferentes de modelado. Esta realidad llegó a ser tan desfavorable que [Slon94] resumió la situación de la siguiente forma: “Las metodologías orientadas a objetos son un fracaso. Existen más de 150 competidores sin un líder claro. Cada metodología ensalza su propia teoría, su propia terminología y sus propias técnicas de diagramación”. A partir de entonces, los esfuerzos no se encaminaron hacia la dispersión sino hacia la unificación de métodos y metodologías.

En general, las principales actividades que incluye el ciclo de vida del software son: planificación del proyecto, gestión del proyecto, análisis, diseño, codificación, pruebas, documentación y mantenimiento. Independientemente de que se trate de una metodología estructurada u orientada a objetos, el modelado en la fase de diseño es la piedra angular para el desarrollo de software, pues su valor está muy ligado al producto obtenido. Si los modelos mejoran la calidad o reducen el coste del producto final, entonces los modelos tienen valor [Fowl99]. Los modelos están escritos sobre la base de un lenguaje de modelado. Dado que el trabajo de esta tesis se basa en los lenguajes de modelado orientados a objetos, en los siguientes apartados vamos a abordar los dos tipos de lenguajes estándares de modelado de

objetos aprobados por el consorcio OMG (*Object Management Group*): UML y OML. Sus acrónimos y sus características se detallan en los apartados 1.2.1. y 1.2.2. respectivamente.

1.2. Lenguajes de modelado

Los lenguajes de modelado se utilizan para visualizar, construir y documentar artefactos de un sistema software. Estos artefactos constituyen los modelos o diagramas que se utilizan para representar una aplicación informática. Los modelos tienen dos características importantes: información semántica (*semántica*) y presentación visual (*notación*). Precisamente estos dos aspectos son importantes a la hora de estudiar la comprensión de los diagramas que se abordan en esta investigación.

1.2.1. El lenguaje unificado de modelado (UML)

Como la "guerra de métodos" de modelado de objetos no hacía progresar la tecnología de objetos, Rumbaugh, Booch y Jacobson, decidieron unificar sus trabajos en un lenguaje de modelado único: UML (*Unified Modeling Language*). En 1997 la definición de la versión 1.0 de UML se estandarizó como lenguaje de modelado de objetos dentro del consorcio OMG.

El desarrollo de UML no sólo aglutina las tres estructuras de modelado orientado a objetos, Booch, OMT y OOSE, sino que también integra elementos de otras. Los casos de uso de OOSE, incorporados a UML, se basan en los diagramas de flujo de datos y de contexto tradicionales [Hitz98]. Los diagramas de clases en UML se sustentan en el lenguaje de modelado existente dentro de OMT, que a su vez deriva de un lenguaje para modelado de datos semánticos [Blah98]. Los diagramas de secuencia se fundamentan en el trabajo de Booch, que a su vez se basa en SDL (*Specification and Description Language*), un lenguaje muy utilizado en la industria de las telecomunicaciones desde hace más de 20 años. Los diagramas de colaboración también se basan en el trabajo de Booch pero está inspirado en los mecanismos disponibles en OORAM [Reen95]. El modelado de los diagramas de estado tiene su origen en el trabajo de [Hare98], que también se utiliza en OMT. Los diagramas de actividad provienen de varias técnicas, como diagramas de eventos de Odell y redes de Petri. Los diagramas de despliegue y de componentes se basan en el trabajo de Booch. A nivel general en la Figura 1.2 se nombran los enfoques, cuyos elementos han tenido alguna influencia y se han integrado en la definición de UML.

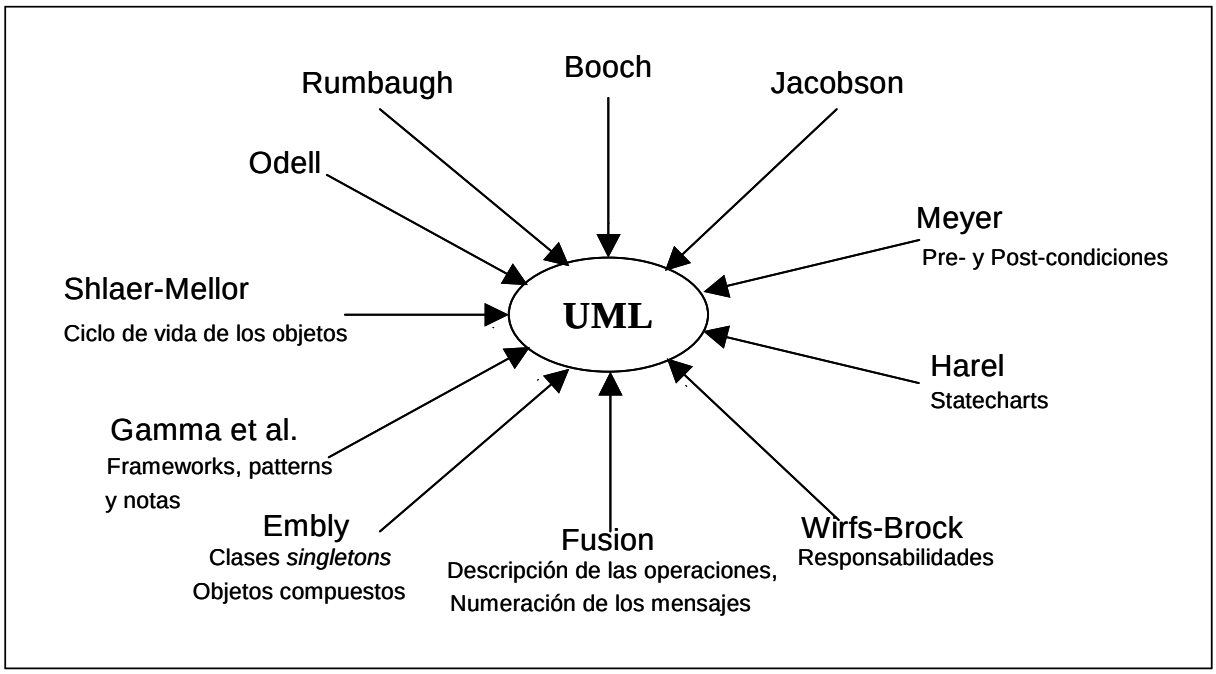


Figura 1.2. Enfoques y elementos implicados en el desarrollo de UML

UML es un lenguaje de modelado de objetos y no un método ni una metodología de objetos. El lenguaje UML se pensó para servir de lenguaje de modelado de objetos independientemente del método implementado. Por lo tanto, UML se puede aplicar a los enfoques orientados a objetos de Booch, OMT y Objectory. Aún así, los creadores de UML, una vez fundada su propia empresa Rational, crearon posteriormente su particular metodología denominada Proceso Unificado (*RUP: Rational Unified Process*) [Jaco00].

1.2.1.1. Vistas de UML

Los conceptos y las construcciones en UML se agrupan en tres áreas: clasificación estructural, comportamiento dinámico y gestión del modelo. Cada una de estas áreas se divide en varias vistas. Una vista es simplemente un subconjunto de UML que modela construcciones que representan un aspecto de un sistema software [OMG01]. La Tabla 1.1. recoge las vistas de UML y los diagramas que las muestran, junto con los principales conceptos de cada diagrama [Rumb00, p. 22].

Asimismo UML posee varias construcciones para proporcionar una capacidad de extensión del lenguaje. Los conceptos que incluyen estas construcciones son: restricciones, estereotipos y valores etiquetados, que se pueden aplicar a los elementos de todas las vistas. Para expresar restricciones UML incorpora la definición de un lenguaje de restricción, denominado OCL (*Object Constraint Language*).

Áreas	Vistas	Diagramas	Conceptos
Estructural	Vista estática	Diagrama de clases	Clase, asociación, generalización, dependencia, interfaz
	Vista de casos de uso	Diagrama de casos de uso	Caso de uso, actor, asociación, extensión, inclusión, generalización
	Vista de implementación	Diagrama de componentes	Componente, interfaz, dependencia
		Diagrama de despliegue	Nodo, componente, dependencia
Dinámica	Vista de máquina de estado	Diagrama de estado	Estado, evento, transición, acción
	Vista de actividad	Diagrama de actividad	Estado, actividad, transición de terminación
	Vista de interacción	Diagrama de secuencia	Interacción, objeto, mensaje, activación
		Diagrama de colaboración	Colaboración, interacción, rol de colaboración, mensaje
Gestión del modelo	Vista de gestión del modelo	Diagrama de clases	Paquete, subsistema

Tabla 1.1. Vistas, diagramas y conceptos de UML

1.2.1.2. Diagramas de UML

Los diagramas deben describir el conocimiento de una aplicación informática en una notación que sea fácil de comunicar. En [OMG01] un diagrama se define como una representación gráfica en dos dimensiones de una colección de elementos de modelado, a menudo dibujada como un grafo con vértices conectados por arcos. Independientemente de que se trate de un elemento de modelado o de un conjunto de ellos, todos estos conceptos se explican de forma detallada en el libro de [Rumb00, capítulo 13], con respecto a los dos aspectos de cualquier lenguaje: semántica y notación.

UML proporciona los diagramas de la Tabla 1.1 para modelar un sistema software, que están organizados en función de las vistas arquitectónicas de dicho sistema. En los siguientes apartados se incluyen sólo aquellos diagramas objeto de estudio en esta investigación: el *diagrama de clases* (apartado 1.2.1.2.1) con respecto al área estructural, y el *diagrama de secuencia* (apartado 1.2.1.2.2), el *diagrama de colaboración* (apartado 1.2.1.2.3) y el *diagrama de estado* (apartado 1.2.1.2.4) con respecto al modelado del comportamiento dinámico, utilizando como ejemplo el de un termostato digital.

1.2.1.2.1. Diagrama de clases

Un diagrama de clases se utiliza para modelar la vista estática de un sistema. Normalmente contiene clases, interfaces y relaciones entre ellas: de asociación, de dependencia y/o de generalización. Además, las clases se agrupan en paquetes y en un diagrama de clases se pueden mostrar las relaciones entre clases pertenecientes a diferentes paquetes. En la Figura 1.3 se muestra el diagrama de clases para el modelado estático de un termostato digital, donde las clases pertenecen a tres paquetes diferentes (Entrada, Control y Salida).

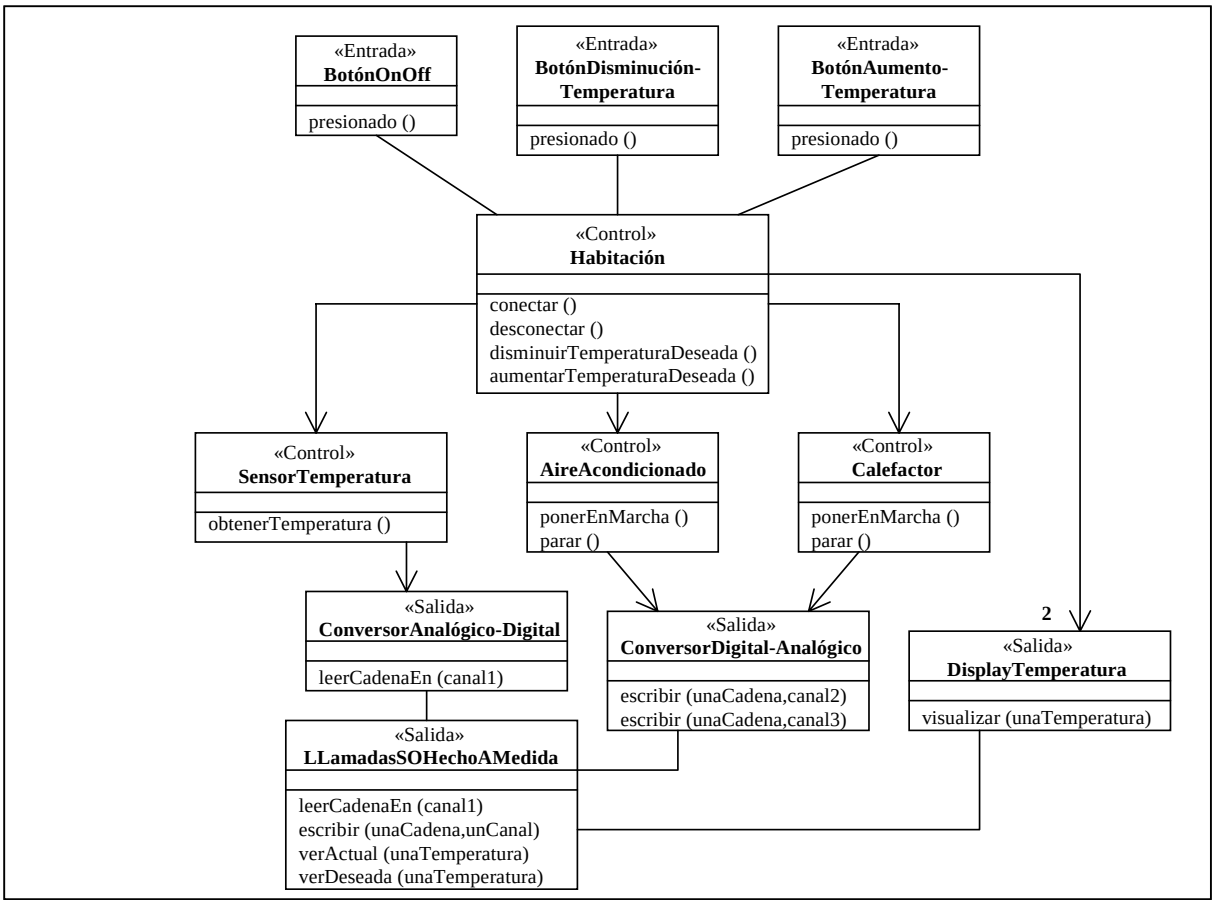


Figura 1.3. Diagrama de clases del termostato digital

Aunque un diagrama de clases es una presentación gráfica de elementos estáticos, también contiene elementos de comportamiento, cuya dinámica se representa en otros diagramas, como diagramas de secuencia, diagramas de colaboración o diagramas de estado.

1.2.1.2.2. Diagrama de secuencia

Un diagrama de secuencia muestra las interacciones entre objetos organizadas visualmente a través del tiempo [Rumb00, p. 216]. Esta descripción sirve para dar detalle a los casos de uso, indicando los mensajes que se intercambian entre los objetos. Es decir, proporciona la

interacción entre los objetos, que se sucede en el tiempo, para un escenario específico durante la ejecución del sistema.

Gráficamente un diagrama de secuencia se representa como un grafo bidimensional: el eje horizontal muestra el conjunto de objetos y el eje vertical muestra el conjunto de mensajes ordenados en el tiempo. Un mensaje se muestra como una flecha horizontal continua desde la línea de vida de un objeto a la del otro. En el diagrama estas flechas se organizan en orden cronológico hacia abajo, por lo que no es necesario numerar los mensajes. Los elementos de modelado utilizados en la construcción de un diagrama de secuencia se muestran en la Figura 1.4, correspondiente al escenario para solicitar pedidos de artículos [Fowl00, p. 104].

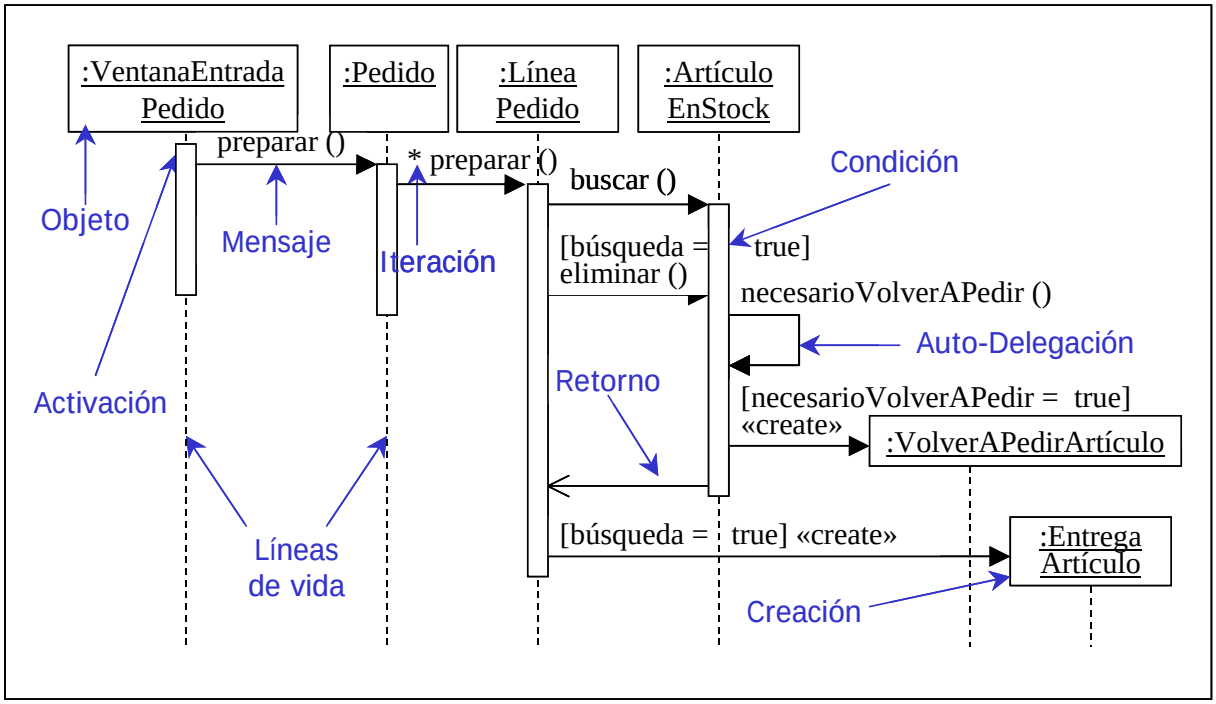


Figura 1.4. Diagrama de secuencia para Pedir Artículos

En esta investigación no sólo se estudiaron los diagramas UML de la vista dinámica, sino también los correspondientes diagramas OML (*OPEN Modeling Language*) para especificar el comportamiento dinámico. Por ello, en la Figura 1.5 se muestra el diagrama de secuencia en lenguaje UML del caso de uso para controlar la temperatura del termostato digital, cuya traducción al diagrama de secuencia de caja blanca en lenguaje OML se puede ver en la Figura 1.16.

En el diagrama de secuencia de la Figura 1.5, los mensajes representados mediante flechas con puntas de flecha con relleno indican llamadas a procedimientos o flujos de control anidado. La secuencia anidada debe terminar antes de que se reanude la secuencia de nivel

más externo. Por ejemplo, hasta que no finalice la secuencia anidada correspondiente al mensaje obtenerTemperatura (), no se puede iniciar la ejecución del siguiente mensaje en la secuencia visualizar (temperaturaActual). Asimismo en este diagrama de secuencia se muestran cuatro flechas que parten de un mismo punto, que representan cuatro bifurcaciones en el flujo de control. Cada una de estas ramificaciones está etiquetada con una condición encerrada entre "[" y "]". Dos de estas condiciones, también denominadas condiciones de guarda, son: [mucho frío] y [mucho calor]. Por otra parte, hay varias flechas con punta de flecha sin relleno, que indican retorno de las llamadas a varios de los procedimientos que aparecen en el diagrama. Estas flechas de retorno se suelen suprimir con el fin de ganar claridad en el diagrama [Fowl00, p. 106].

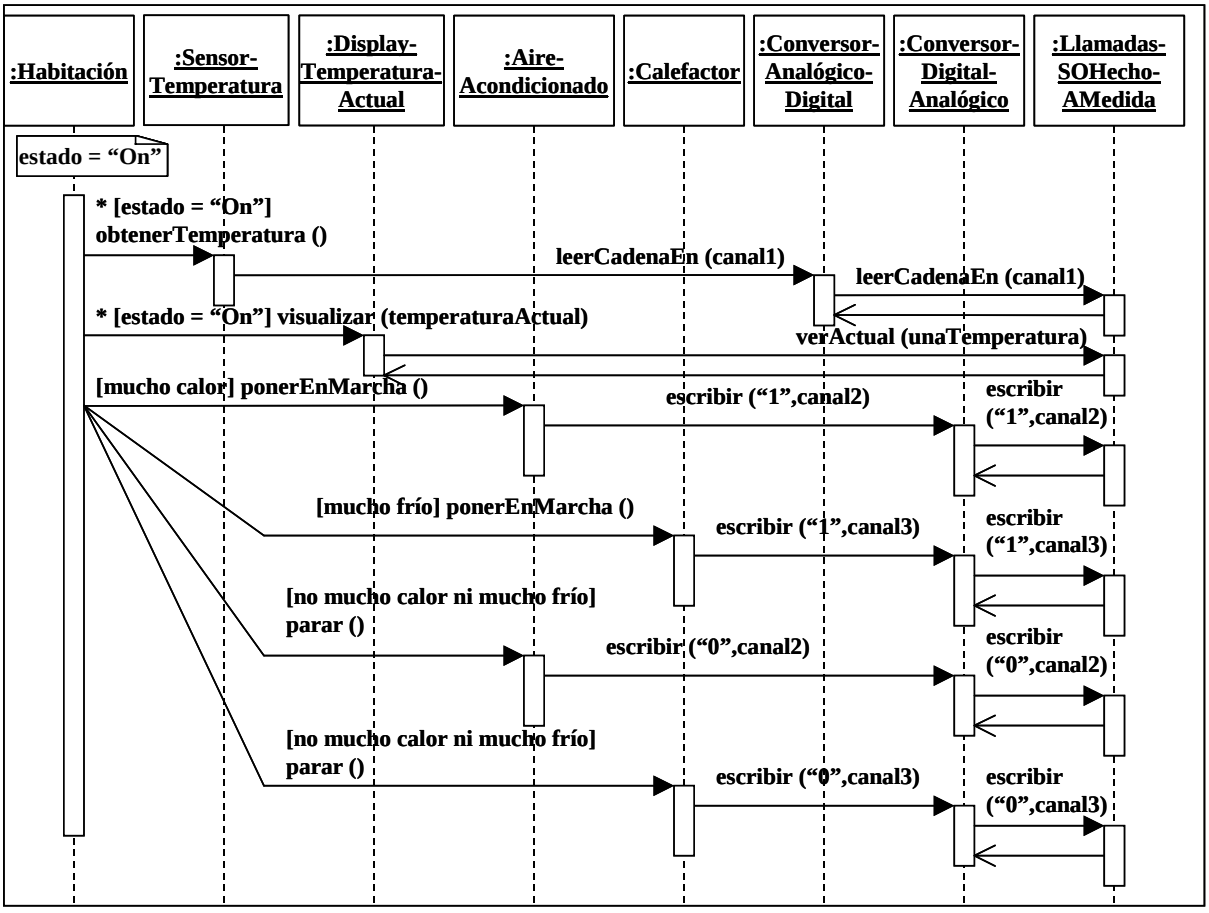


Figura 1.5. Ejemplo de un diagrama de secuencia para Controlar la Temperatura

1.2.1.2.3. Diagrama de colaboración

Un diagrama de colaboración muestra las interacciones entre objetos organizadas en torno a los objetos que efectúan operaciones [Rumb00, p. 211]. Esta descripción se centra en la

organización estructural de los objetos que envían y reciben mensajes. Es decir, se parece a un diagrama de objetos que muestra los objetos y los enlaces que existen entre ellos y que son necesarios para implementar un escenario.

Un diagrama de colaboración se representa como un grafo formado por un conjunto de vértices, los objetos, y de arcos, los enlaces que conectan dichos vértices. Como en un diagrama de colaboración no se muestra el tiempo, es necesario etiquetar con números la ejecución de la secuencia de los mensajes. Los números de secuencia y otros conceptos de modelado empleados para dibujar un diagrama de colaboración se muestran en la Figura 1.6, correspondiente al escenario para solicitar pedidos de artículos [Fowl00, p. 111].

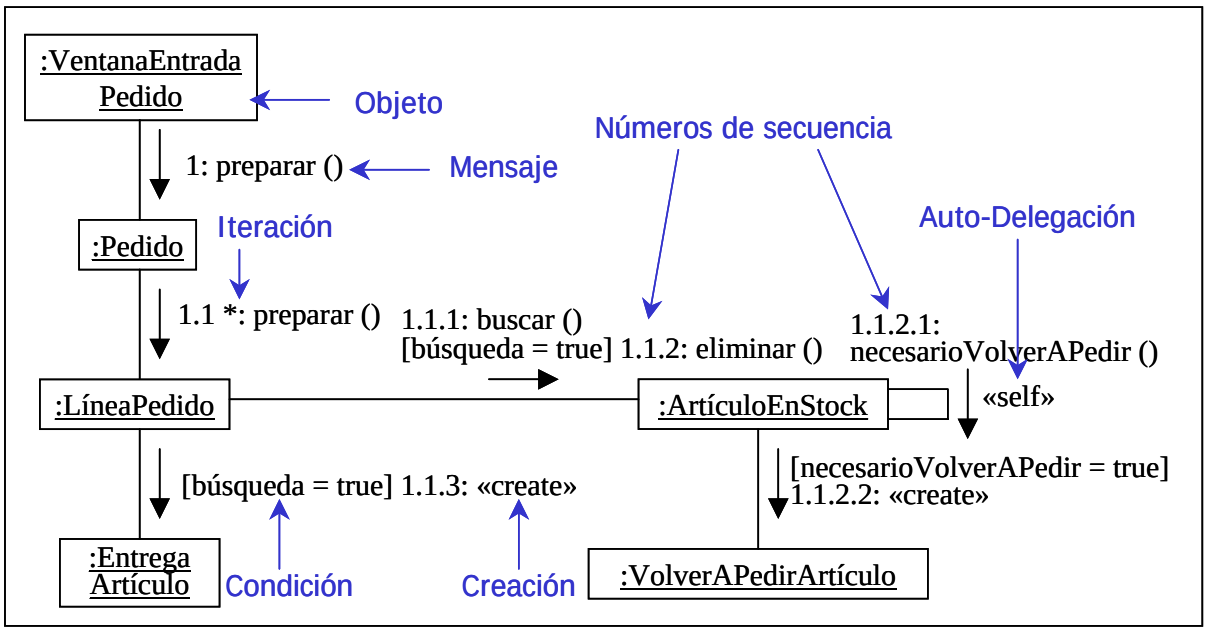


Figura 1.6. Diagrama de colaboración para Pedir Artículos

Un diagrama de secuencia y de colaboración sirven para representar la misma información, pero la especifican de maneras diferentes. Así pues, el comportamiento dinámico del mismo escenario ControlarTemperatura se especifica bien mediante un diagrama de secuencia en la Figura 1.5, o bien mediante un diagrama de colaboración en la Figura 1.7. El diagrama de secuencia de la Figura 1.5 muestra la secuencia explícita de los mensajes, pero no los enlaces entre los objetos. El diagrama de colaboración de la Figura 1.7 ilustra qué objetos están relacionados y qué mensajes envían y/o reciben. En otras palabras, un diagrama de secuencia ofrece una dimensión en el tiempo, pero no en el espacio. En cambio, el diagrama de colaboración presenta una visión espacial de los objetos y de sus enlaces, pero no incorpora el tiempo como una dimensión más.

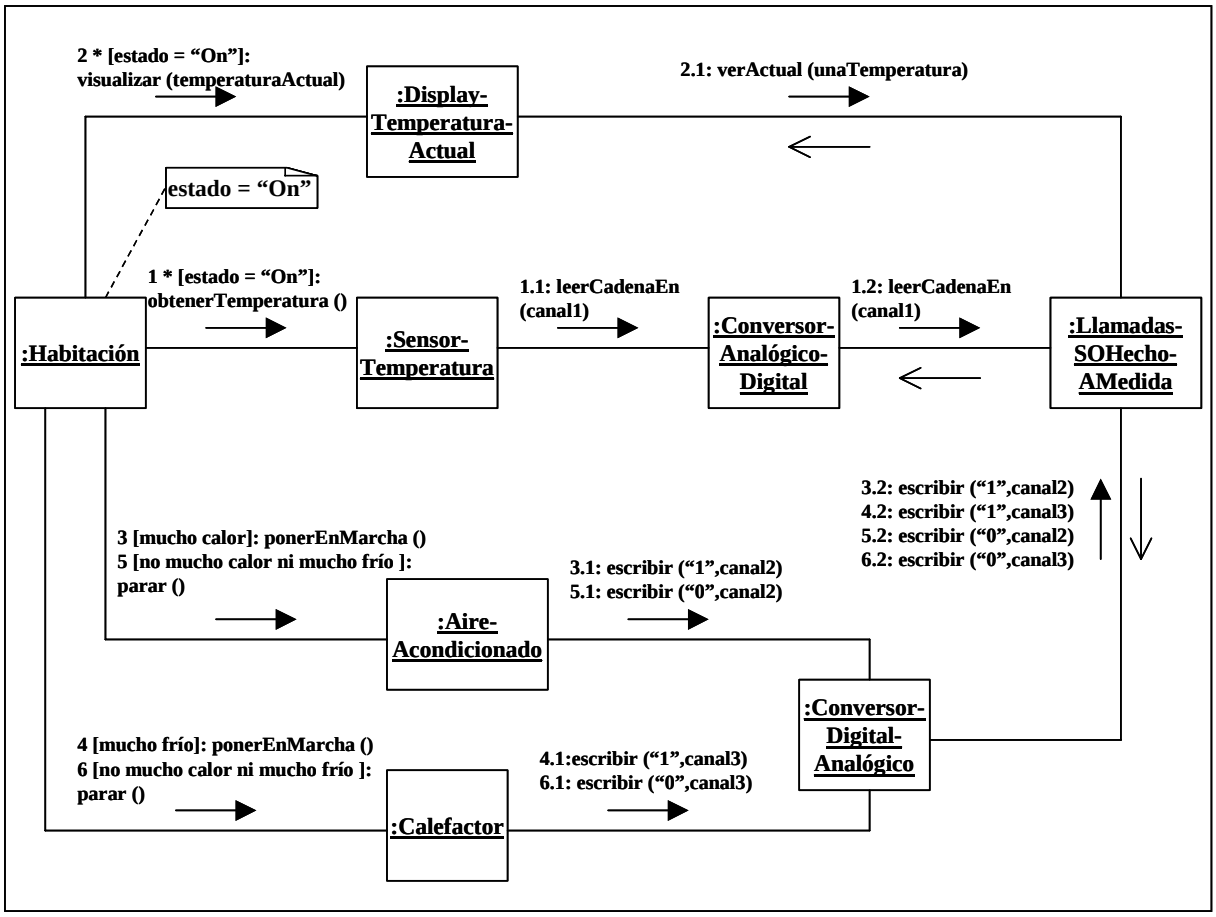


Figura 1.7. Ejemplo de un diagrama de colaboración para Controlar la Temperatura

En el diagrama de colaboración de la Figura 1.7, los enlaces se representan mediante líneas que conectan los objetos y los mensajes por medio de flechas continuas con puntas con relleno o sin relleno, dependiendo del flujo de control. En este caso los mensajes van etiquetados con números de secuencia. La numeración puede ser simple (1, 2, 3, ...) o anidada (1, 1.1, 1.2, 1.1.1, 1.1.2, ...). Este último tipo de numeración es la elegida para nuestros diagramas de colaboración. Por ejemplo, el mensaje 3.2 sigue al mensaje 3.1 dentro del mensaje 3. Opcionalmente seguido del número de secuencia puede haber una cláusula de iteración (precedida de un "*") o una cláusula de condición, para indicar que se ejecutan cero o más mensajes. Por ejemplo, en este diagrama de colaboración se muestran dos mensajes iterativos, 1* [estado = "On"]: obtenerTemperatura () y 2* [estado = "On"]: visualizar (temperaturaActual). Mientras el termostato detecte que el aparato está conectado, primero se lee y luego se visualiza repetidamente la temperatura real de la habitación. También aparecen cuatro mensajes alternativos, cuya ejecución depende de que la condición sea verdad. Por ejemplo, si se cumpla la condición del mensaje 3, [mucho calor], se ejecuta el mensaje

ponerEnMarcha (), lo que implica que el aire acondicionado del aparato se ponga en funcionamiento.

1.2.1.2.4. Diagrama de estado

Un diagrama de estado muestra el conjunto de estados por los que pasa un objeto o una interacción en respuesta a los distintos eventos a lo largo de su vida, junto con las acciones de respuesta [Rumb00, p. 345]. Asimismo ilustra qué eventos son los que pueden cambiar el estado de los objetos de la clase. Un diagrama de estado se puede asociar a un clasificador, tal como una clase o un caso de uso, así como a una colaboración o a un método.

Un diagrama de estado es un grafo de estados y transiciones, que a su vez incluye eventos, acciones y actividades. La Figura 1.8 muestra el diagrama de estado de una petición de artículos [Fowl00, p. 122]. Este diagrama indica los diferentes estados en los que se puede encontrar una petición, comenzando en el punto de inicio.

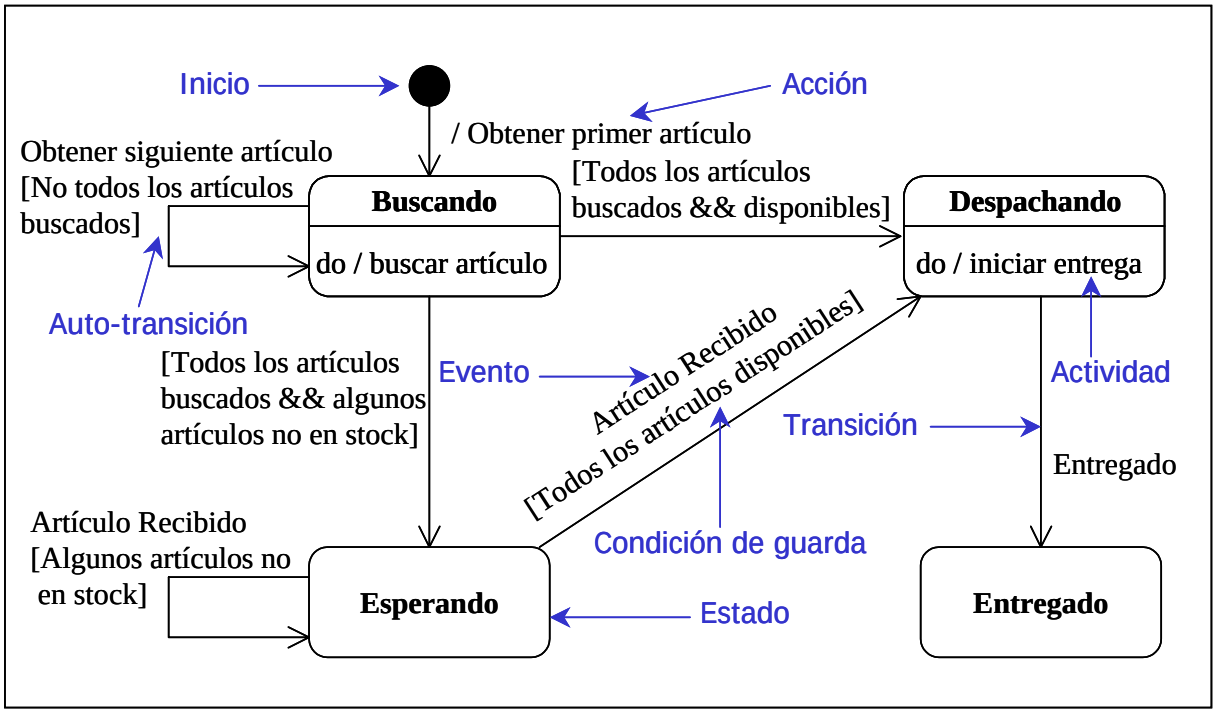


Figura 1.8. Diagrama de estado para Petición de Artículos

Un evento puede causar la transición de un estado a otro de un objeto. La sintaxis de una transición consta de tres partes opcionales: *Evento [Condición de guarda] / Acción*. Una acción es una operación atómica que no se puede interrumpir por un evento y que se ejecuta hasta su finalización. En cambio, una actividad contrasta con una acción, pues mientras un objeto está en un estado realizando una actividad, ésta puede ser interrumpida por un evento.

Cuando un estado tiene asociada una actividad, se identifica mediante una etiqueta con la siguiente sintaxis: *do / actividad*. Así pues, una acción se asocia a una transición de estado y una actividad a un estado. Por ejemplo, en la Figura 1.8 para pasar del estado de Inicio al estado Buscando se ejecuta la acción */ Obtener primer artículo* y permaneciendo en el estado Buscando se ejecuta la actividad *do / buscar artículo*.

Dado que en el capítulo 8 de esta tesis se examinan los diagramas para documentar el comportamiento de las instancias de una clase tanto en UML como en OML, en la Figura 1.9 se muestra el diagrama de estado en notación UML para el objeto de la clase Habitación del termostato digital. En él aparecen tres condiciones de guarda, que son excluyentes entre sí: [muchoCalor], [muchoFrio] y [Otras Situaciones]. Los dos diagramas en lenguaje OML que se corresponden con este diagrama de estado se muestran en las Figuras 1.17 y 1.18.

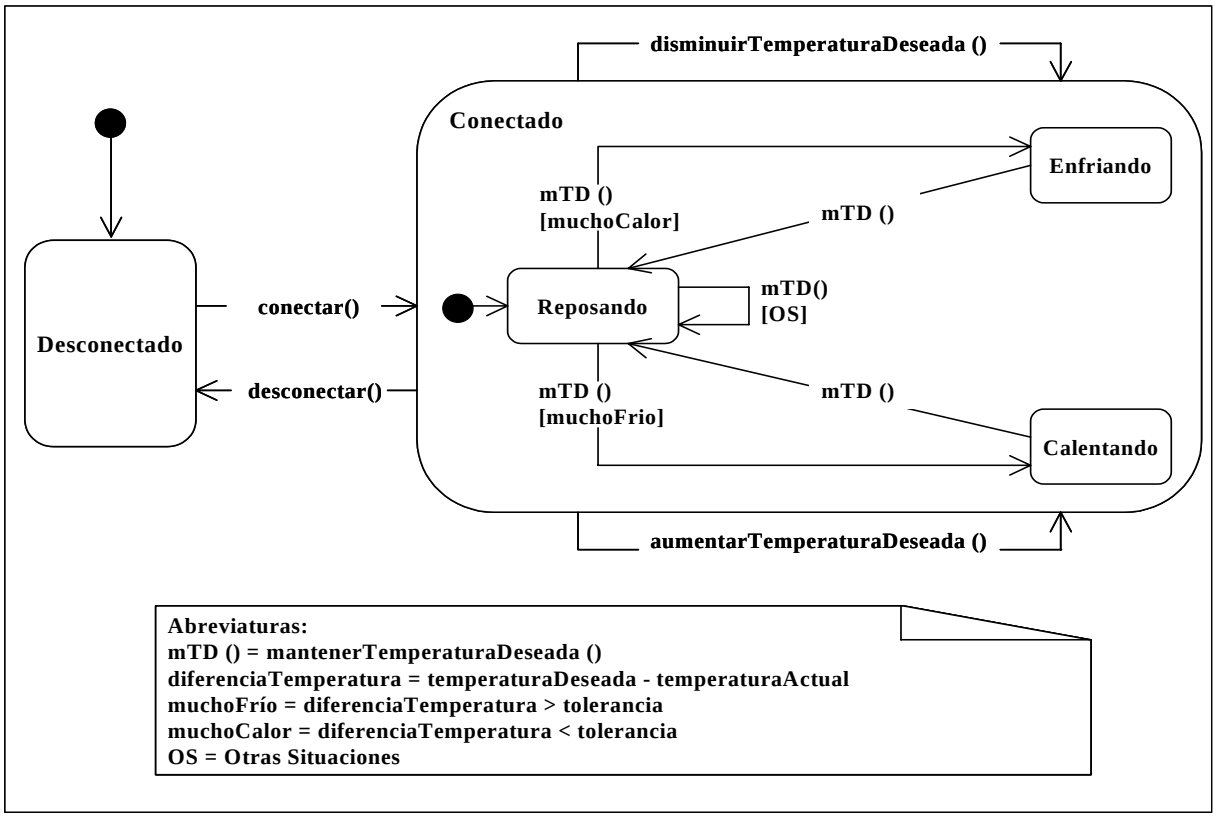


Figura 1.9. Ejemplo de un diagrama de estado para la instancia de la clase Habitación

1.2.2. El lenguaje de modelado de OPEN (OML)

OPEN es un acrónimo de *Object-oriented Process, Environment and Notation*. En un principio se creó mediante la fusión de tres métodos orientados a objetos: MOSES [Hend94], SOMA [Grah95] y el método de Firesmith [Fire93]. En el año 1997 la metodología Synthesis se incluyó en OPEN. Posteriormente mejoró por el estado de la cuestión correspondiente a los conceptos derivados de BON (*Business Object Notation*) [Wald95], Mainstream Objects, Martin & Odell, OBA (*Object Behavior Analysis*) [Rubi92], RDD (*Responsibility-Driven Design*) [Wirf90], ROOM (*Real-time Object-Oriented Modeling*) [Seli94], Syntropy [Cook94] y UML v 1.1. Al igual que sucedió con la definición de UML, que integró elementos de diferentes métodos, OPEN también agrupa conceptos de diversos enfoques orientados a objetos que se muestran en la Figura 1.10.

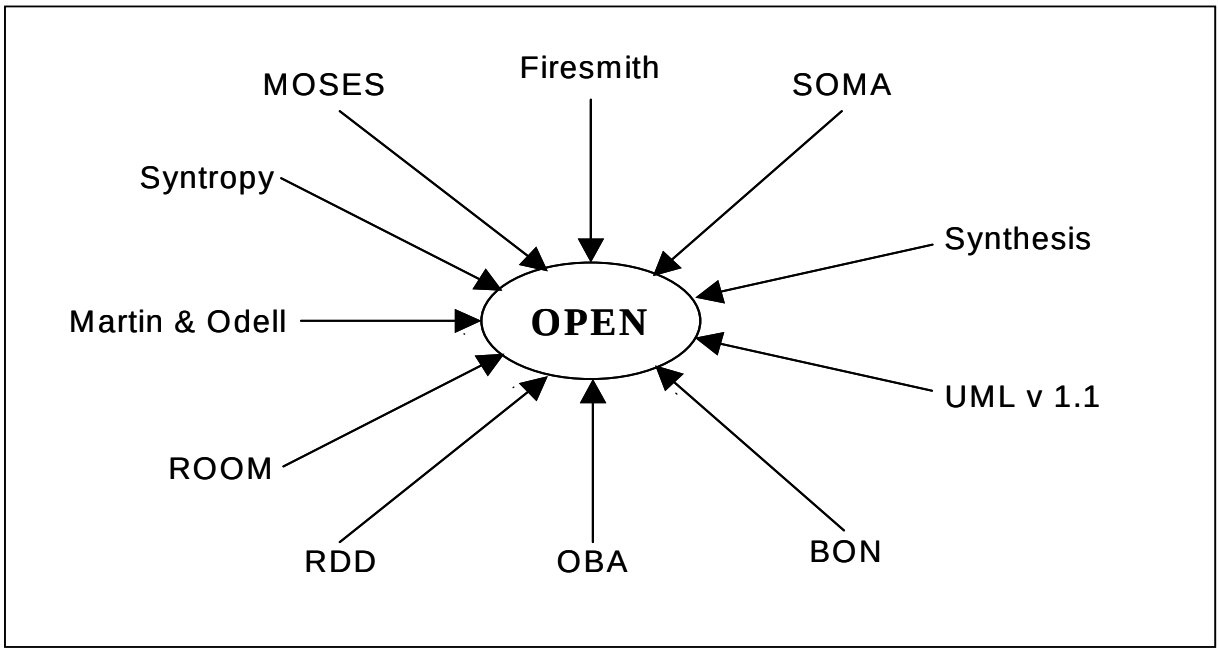


Figura 1.10. Métodos implicados en la configuración de OPEN

OPEN es una metodología OO de ciclo de vida completo, aprobada por el consorcio OPEN, una organización internacional formada por más de treinta miembros entre investigadores, desarrolladores de software y vendedores de herramientas CASE. Esta metodología incluye un proceso completo de desarrollo de software, que contiene [Hend98]:

a) Una estructura de proceso de desarrollo consistente en actividades que se descomponen en tareas que se ejecutan mediante el uso de técnicas para generar entregables.

b) Un lenguaje de modelado denominado OML (*OPEN Modeling Language*) que consta de dos componentes (ver Figura 1.11):

- El metamodelo derivado del proyecto COMMA, que define la terminología, sintaxis y semántica de los elementos de modelado.
- La notación COMN (*Common Object Modeling Notation*), que define los diagramas OML (apartado 1.2.2.3) y sus iconos.

c) Estándares, procedimientos y guías.

d) Métricas.

Durante el año 1995 la propuesta por parte del grupo OMG era lograr una notación y un metamodelo estándar con el fin de soportar la interoperabilidad de las herramientas CASE frontales, que abarcan las fases de análisis y diseño. Esta propuesta llevó a que algunos miembros del consorcio OPEN invirtieran sus esfuerzos en esa dirección. Entonces se creó el lenguaje de modelado OPEN, denominado OML, que aúna tanto al metamodelo como a la notación. Paralelamente la empresa Rational concibió una combinación similar, conocida como UML. Más adelante, OML proporcionará una extensión a la notación y al metamodelado de UML, de acuerdo con las normas del consorcio OMG, siendo OML aprobado como lenguaje estándar de modelado en 1998. Las notaciones que incorporan ambos lenguajes se pueden utilizar conjuntamente con la metodología OPEN [Hend99b].

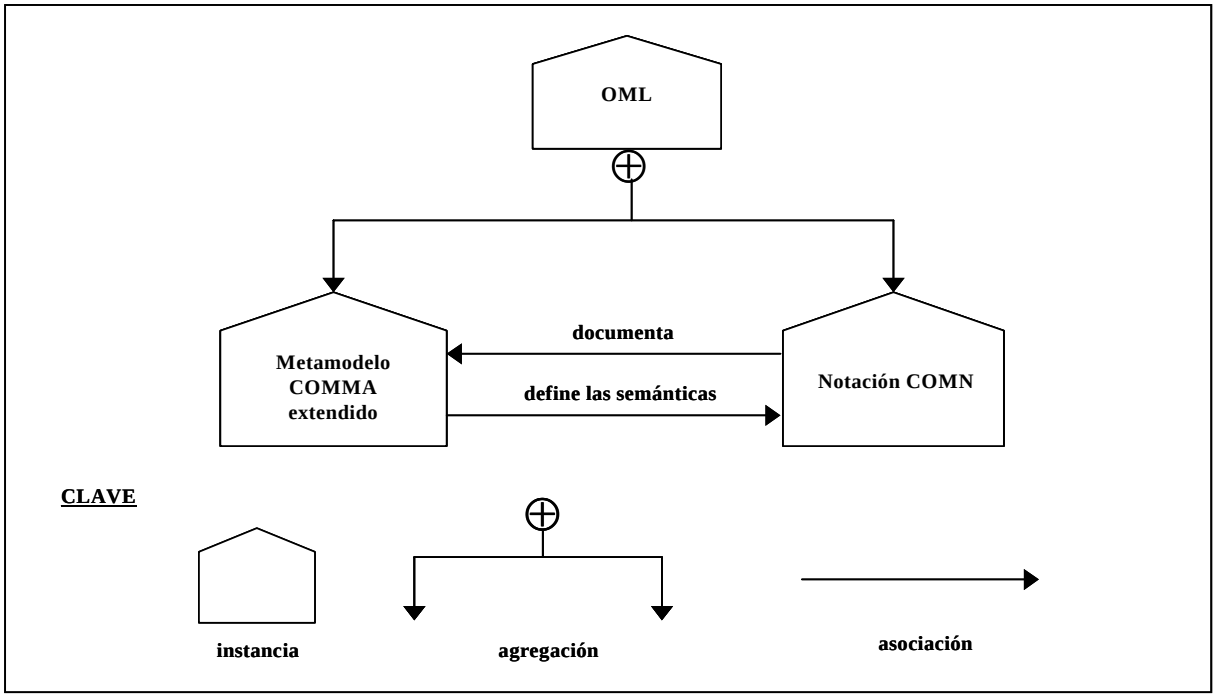


Figura 1.11. Componentes de OML

Mientras que una notación se utiliza para modelar aplicaciones y dominios de empresa con una gran cantidad de complejidad inherente, una sola vista o un único tipo de diagrama no es adecuado para capturar todos los aspectos del modelo resultante. El lenguaje OML al igual que el lenguaje UML ofrece un conjunto de tipos de diagramas que proporcionan vistas ortogonales del modelo subyacente. Gráficamente, los diagramas son grafos formados por vértices (*nodos comunes*) y por arcos (*relaciones comunes*) que conectan dichos vértices.

1.2.2.1. Nodos comunes

Los nodos comunes que se utilizan en los diagramas de OML incluyen objetos (tanto internos como externos), clases, tipos, *roles*, paquetes y notas [Fire98a]. En el metamodelo de la Figura 1.12 se muestran algunos de estos nodos (*CIRT: Class, Instance, Role and Type*) y sus conexiones.

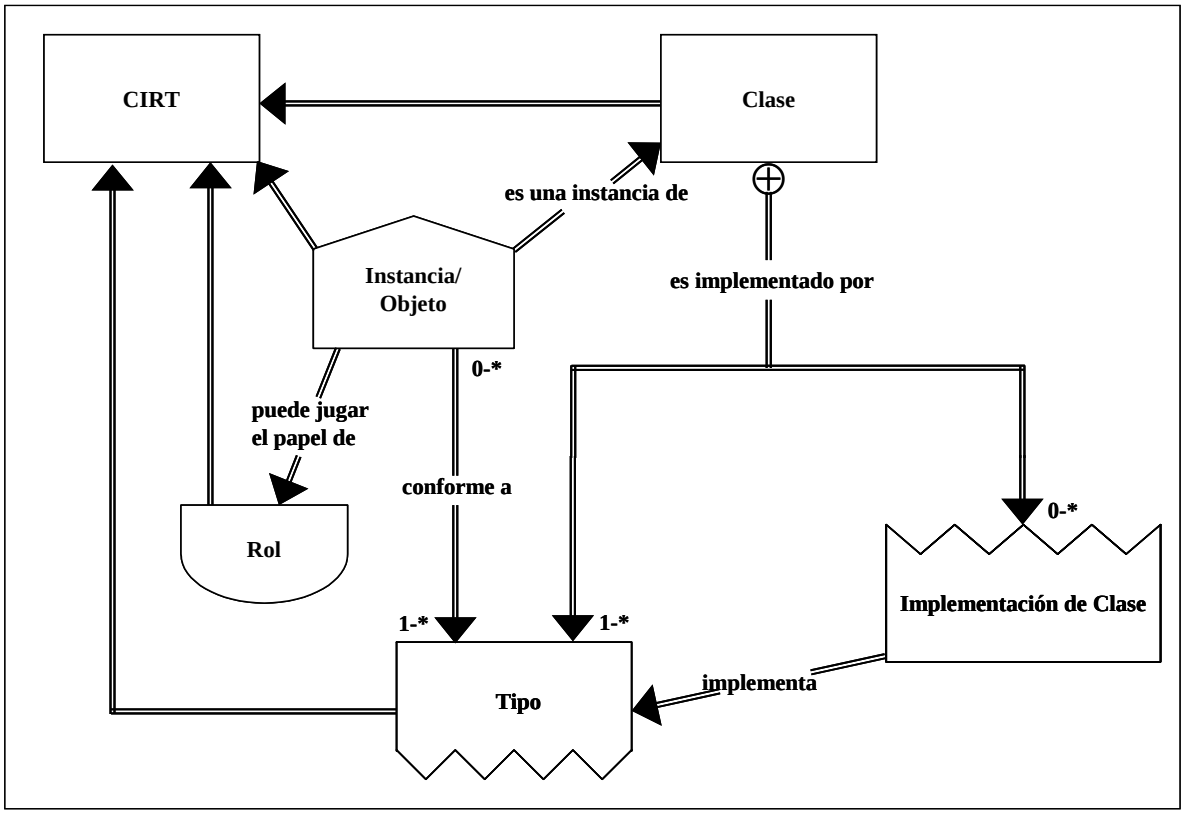


Figura 1.12. Metamodelo para CIRT

1.2.2.2. Relaciones comunes

En OML todas las relaciones son binarias y dependientes en una única dirección. Todos los arcos de las relaciones COMN se dibujan desde el cliente hasta el servidor y su dirección se especifica mediante una cabeza de flecha al final del servidor [Fire98a]. El lenguaje OML reconoce los tipos de relaciones que se muestran en la Figura 1.13.

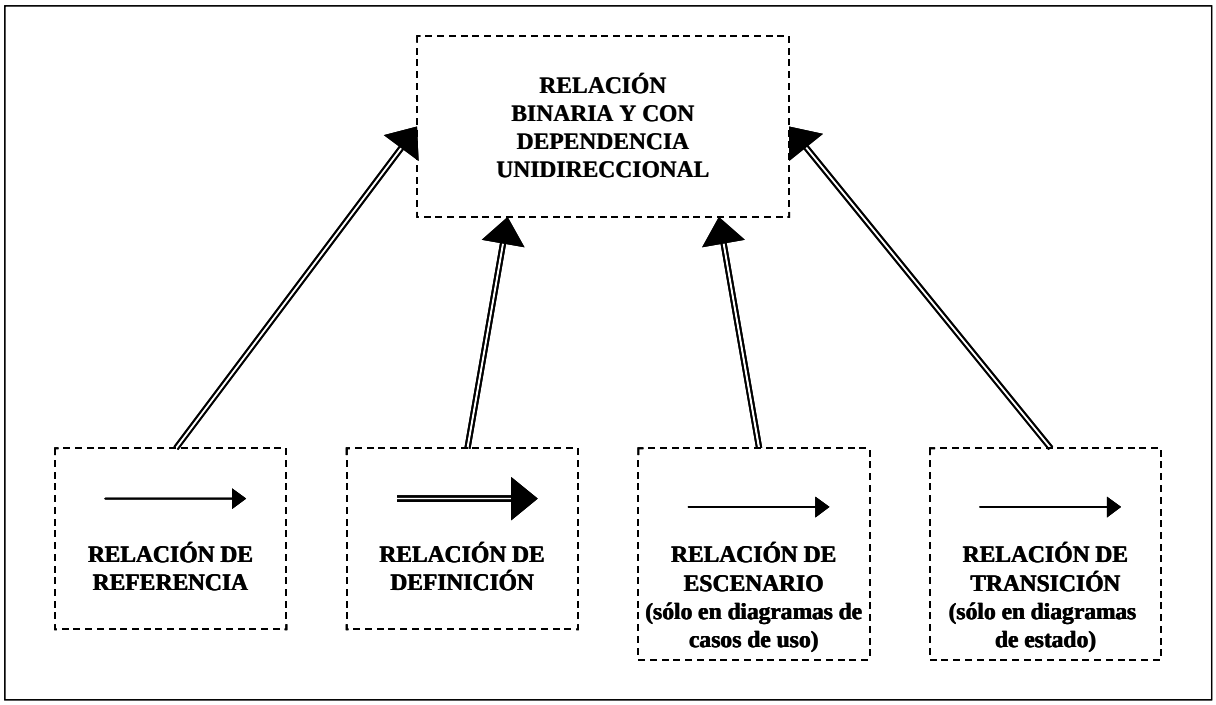


Figura 1.13. Metamodelo para las relaciones

1.2.2.3. Diagramas de OML

A nivel general OML incluye cuatro tipos principales de diagramas, que a su vez agrupan a varios diagramas, tal como muestra la Tabla 1.2. Unos diagramas documentan la arquitectura estática, mientras que otros documentan el comportamiento dinámico.

Diagramas principales	Diagramas específicos
Redes semánticas	• Diagrama de contexto o Diagrama de contexto del sistema o Diagrama de contexto del software • Diagrama de configuración • Diagrama de paquete • Diagrama de herencia • Diagrama de capas • Diagrama de despliegue
Diagramas de clases escenario	
Diagramas de interacción	• Diagrama de secuencia o Diagrama de secuencia de caja negra o Diagrama de secuencia de caja blanca • Diagrama de colaboración o Diagrama de colaboración de paquete o Diagrama de colaboración interna
Diagramas de transición de estado	

Tabla 1.2. Diagramas de OML

Para ilustrar los diferentes tipos de diagramas de OML que fueron objeto de estudio en esta investigación se utiliza como ejemplo el de un termostato digital con un funcionamiento sencillo [Fire98b]. Estos diagramas más relevantes son: el *diagrama de colaboración de paquete* (apartado 1.2.2.3.1), el *diagrama de secuencia de caja blanca* (apartado 1.2.2.3.2), el *diagrama de transición de estado* (apartado 1.2.2.3.3.) y el *diagrama de colaboración interna* (apartado 1.2.2.3.4).

1.2.2.3.1. Diagrama de colaboración de paquete

Un diagrama de paquete es una red semántica especializada en documentar la parte estática de uno o varios paquetes en términos de sus clases, tipos, *roles* y objetos así como de las relaciones significativas semánticamente entre ellos [Fire98b, p. 26]. Sin embargo, esta documentación no especifica qué mensajes son los que realmente se pueden enviar y/o recibir entre dichos nodos comunes CIRT, tal como muestra la Figura 1.14.

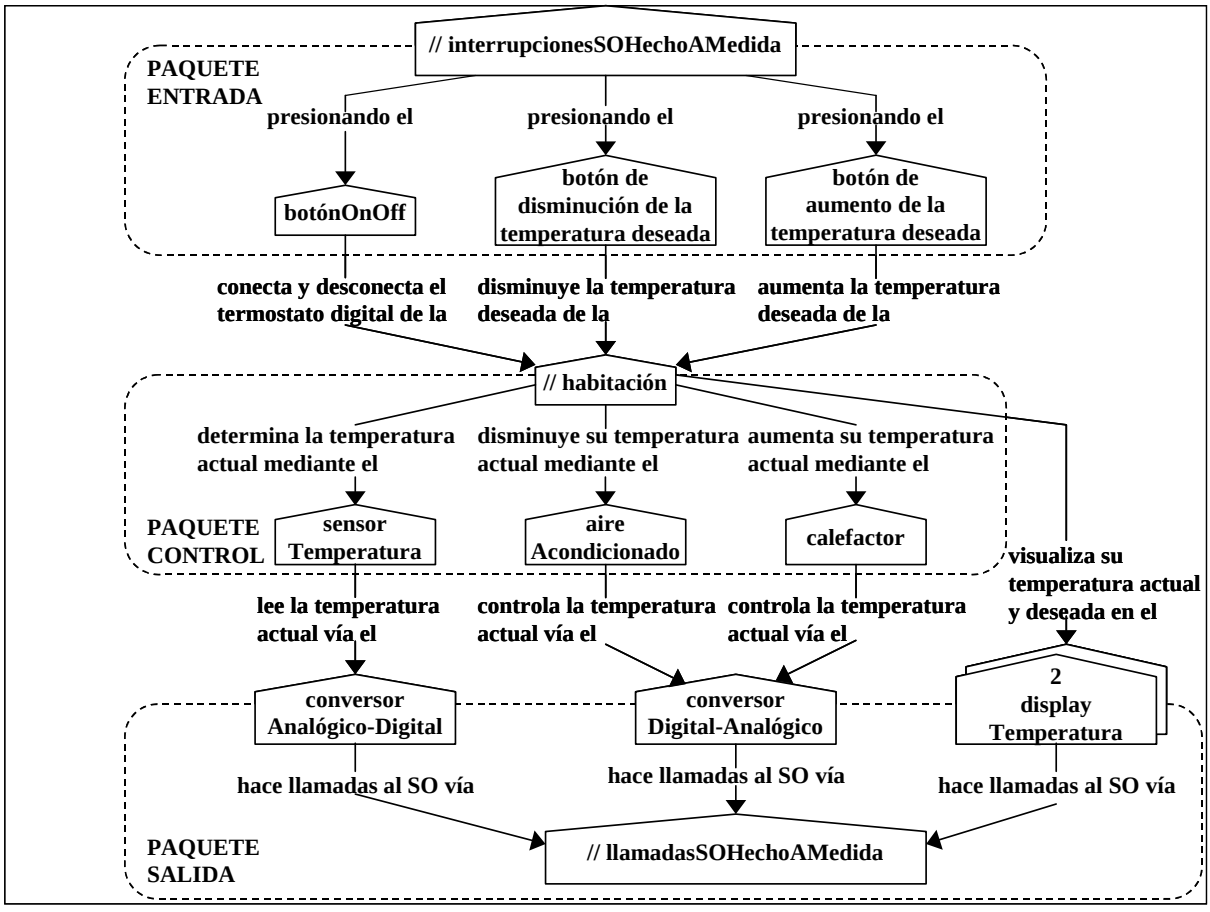


Figura 1.14. Diagrama de paquete del termostato digital

Aunque el diagrama de paquete OML de la Figura 1.14 es el equivalente al diagrama de clases UML de la Figura 1.3, este tipo de diagrama no se utilizó en esta investigación ya que el nivel de detalle no era el mismo en ambos modelos. Precisamente ese nivel de detalle en los modelos OML sólo se consigue recurriendo al diagrama de colaboración de paquetes, tal como se indica en [Fire98b, p. 380].

Un diagrama de colaboración de paquete es un documento de colaboración especializado que documenta la colaboración contenida dentro de uno o varios paquetes de clases de caja negra y de objetos en términos de los mensajes que pueden enviar y de las excepciones que pueden mostrar [Fire98b, p. 29]. La Figura 1.15 especifica el diagrama de colaboración de paquete completo del termostato digital, que incluye los objetos en los tres paquetes y los mensajes entre ellos.

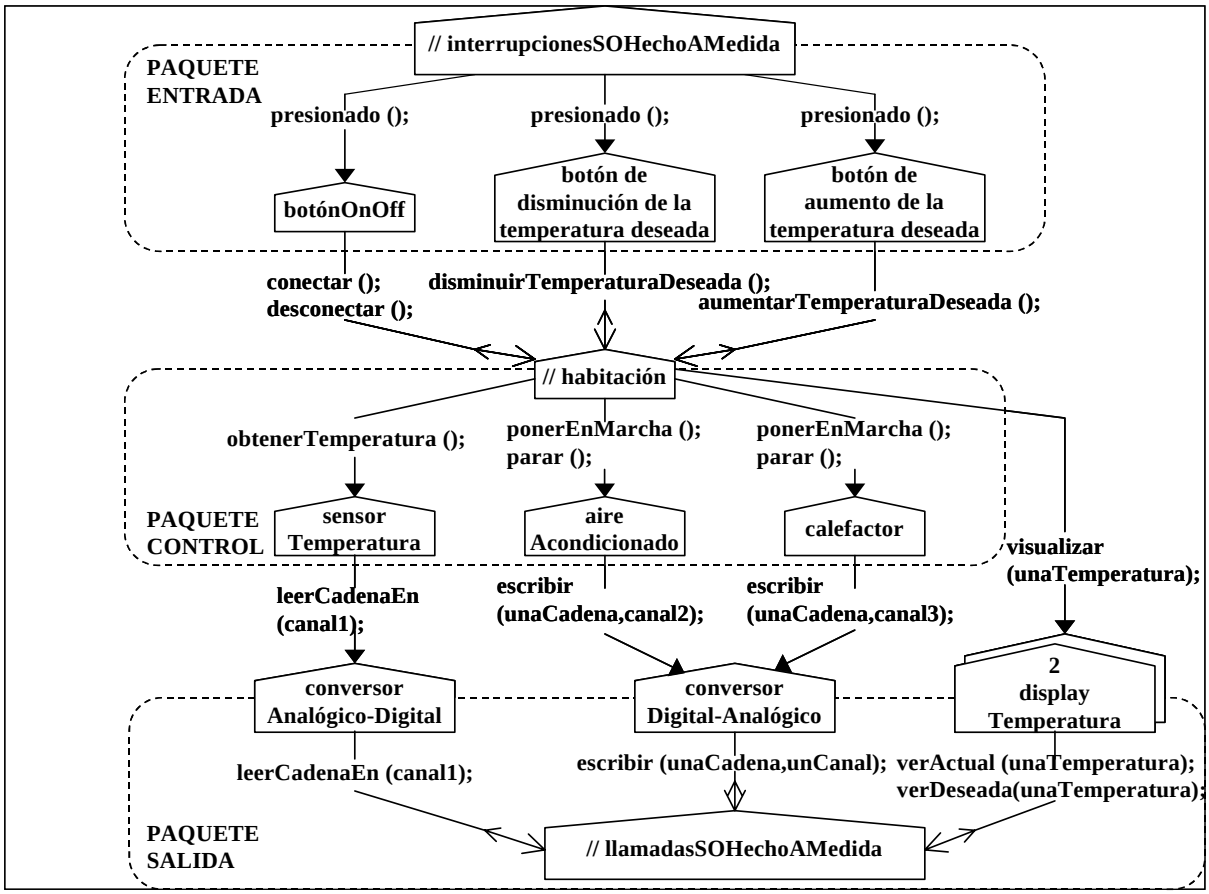


Figura 1.15. Diagrama de colaboración de paquete del termostato digital

1.2.2.3.2. Diagrama de secuencia de caja blanca

Un diagrama de secuencia de caja blanca es un diagrama de secuencia que documenta la secuencia de interacciones entre los objetos externos y las clases implicadas en un único patrón de diseño o parte de un caso de uso [Fire98b, p. 32]. Un diagrama de secuencia de caja

blanca trata la aplicación como una caja blanca, mostrando las interacciones entre sus objetos internos. Se utiliza para especificar el comportamiento dinámico y documentar un caso de prueba de integración. La Figura 1.16 documenta el diagrama de secuencia de caja blanca para controlar la temperatura del termostato digital, mostrando los objetos, las líneas del tiempo, los diferentes tipos de mensajes y las excepciones.

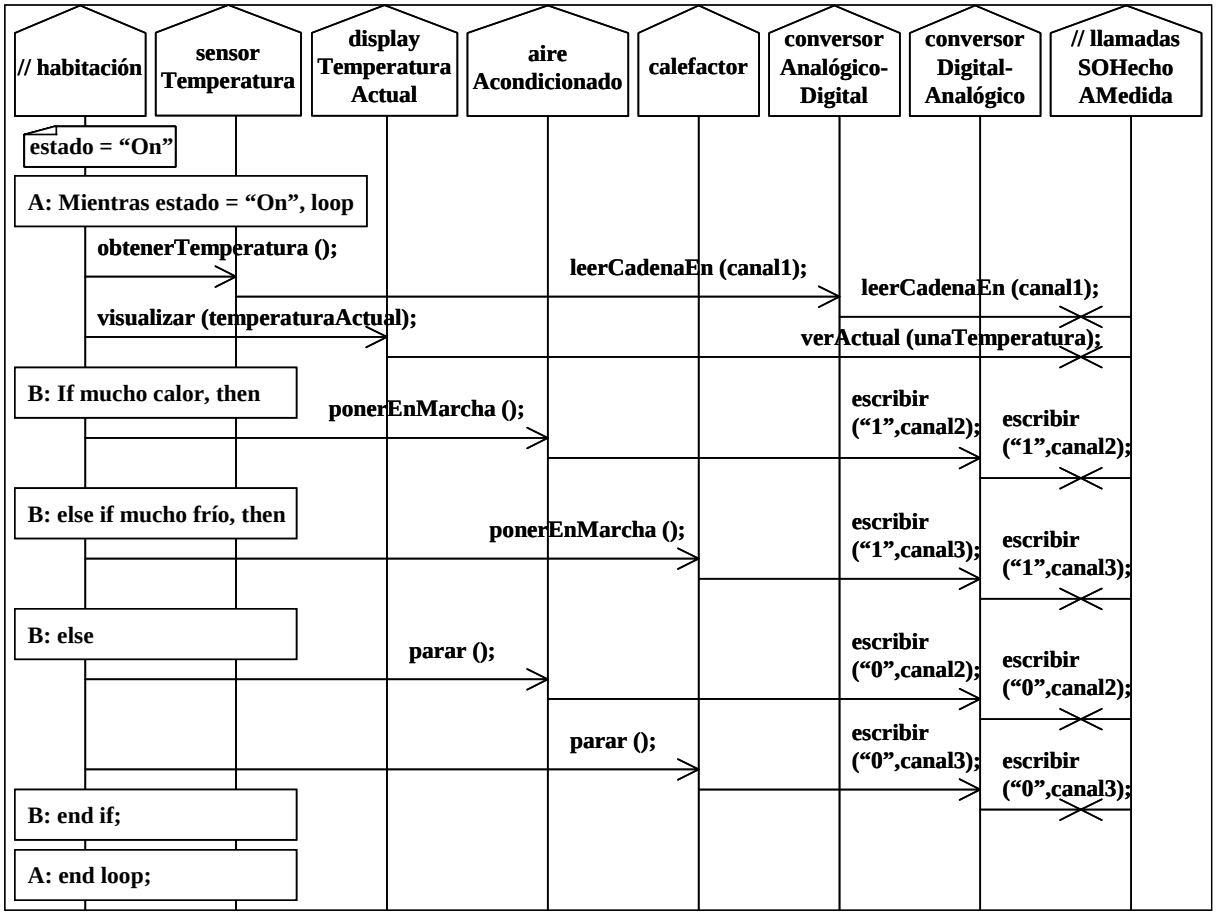


Figura 1.16. Ejemplo de un diagrama de secuencia de caja blanca para Controlar la Temperatura

1.2.2.3.3. Diagrama de colaboración interna

Un diagrama de colaboración interna es un diagrama que documenta la colaboración de las operaciones contenidas dentro de una clase u objeto individual, en términos del flujo de control (por ejemplo, el envío de un mensaje, la activación de una excepción) y del flujo del objeto (es decir, la lectura y la escritura de propiedades dentro de una clase o un objeto) [Fire98b, p. 31]. El concepto de flujo del objeto es análogo al de flujo de datos en un diagrama de flujo de datos tradicional. Este flujo se representa mediante un arco de flecha discontinua que conecta una operación y un atributo. La Figura 1.17 especifica el diagrama de

colaboración interna para las instancias de la clase Habitación, incluyendo las operaciones, las propiedades y las interacciones y los flujos del objeto entre ellas.

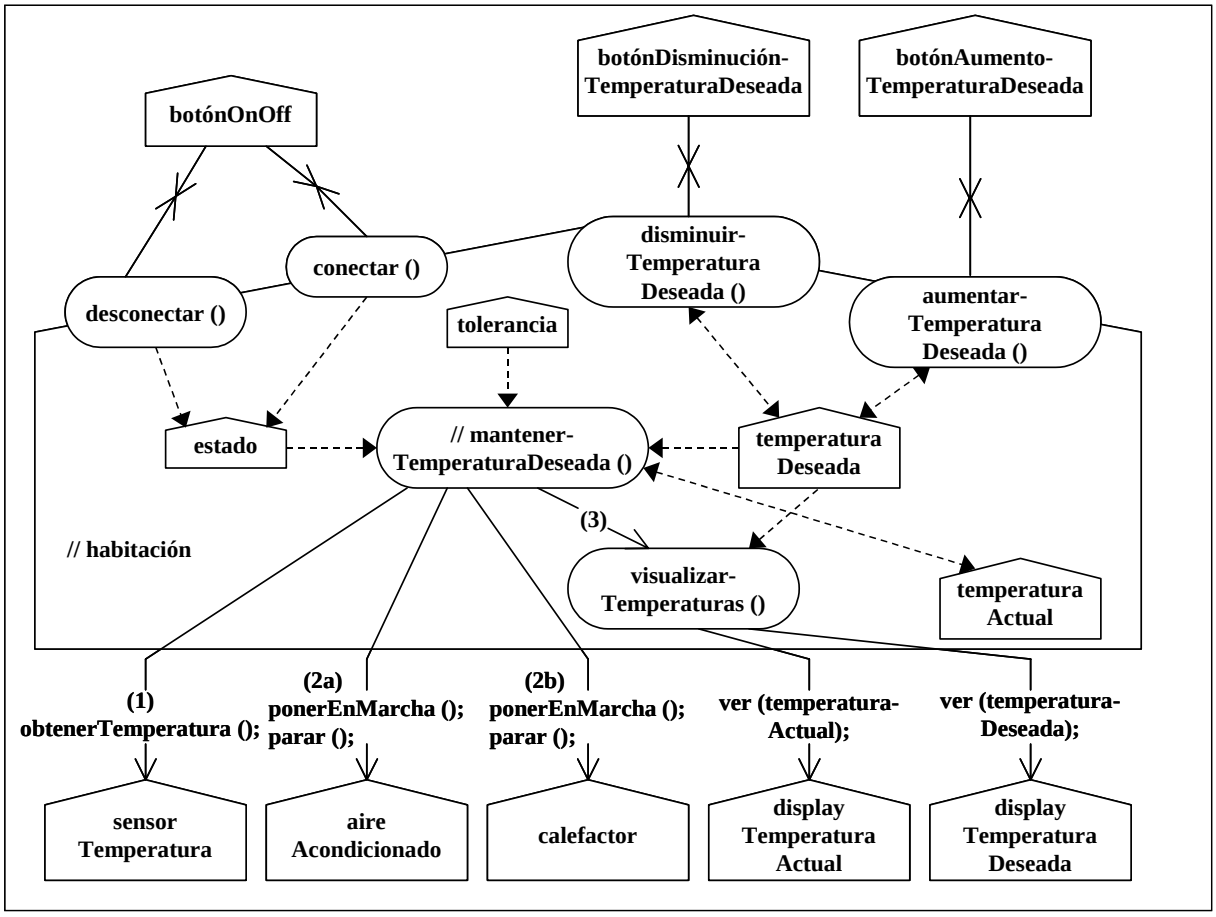


Figura 1.17. Ejemplo de un diagrama de colaboración interna para la instancia de la clase Habitación

1.2.2.3.4. Diagrama de transición de estado

Un diagrama de transición de estado es un diagrama consistente en un grafo dirigido de estados (nodos) conectado por medio de transiciones (arcos dirigidos), que documenta el comportamiento común (es decir, si una operación se ejecuta o no, o qué camino se toma de una rama lógica) de las instancias de una clase [Fire98b, p. 33]. Un diagrama de transición de estado describe la historia de la vida de los objetos de una clase dada, en términos de cómo responden a las interacciones con otros objetos.

La Figura 1.18 especifica el diagrama de transición de estado del objeto de la Clase Habitación en el ejemplo del termostato digital. Las operaciones desconectar(), disminuirTemperaturaDeseada() y aumentarTemperaturaDeseada() disparan transiciones de grupo y la operación mantenerTemperaturaDeseada() forma parte de las condiciones de guarda: [muchoCalor], [muchoFrio] y [Otras Situaciones].

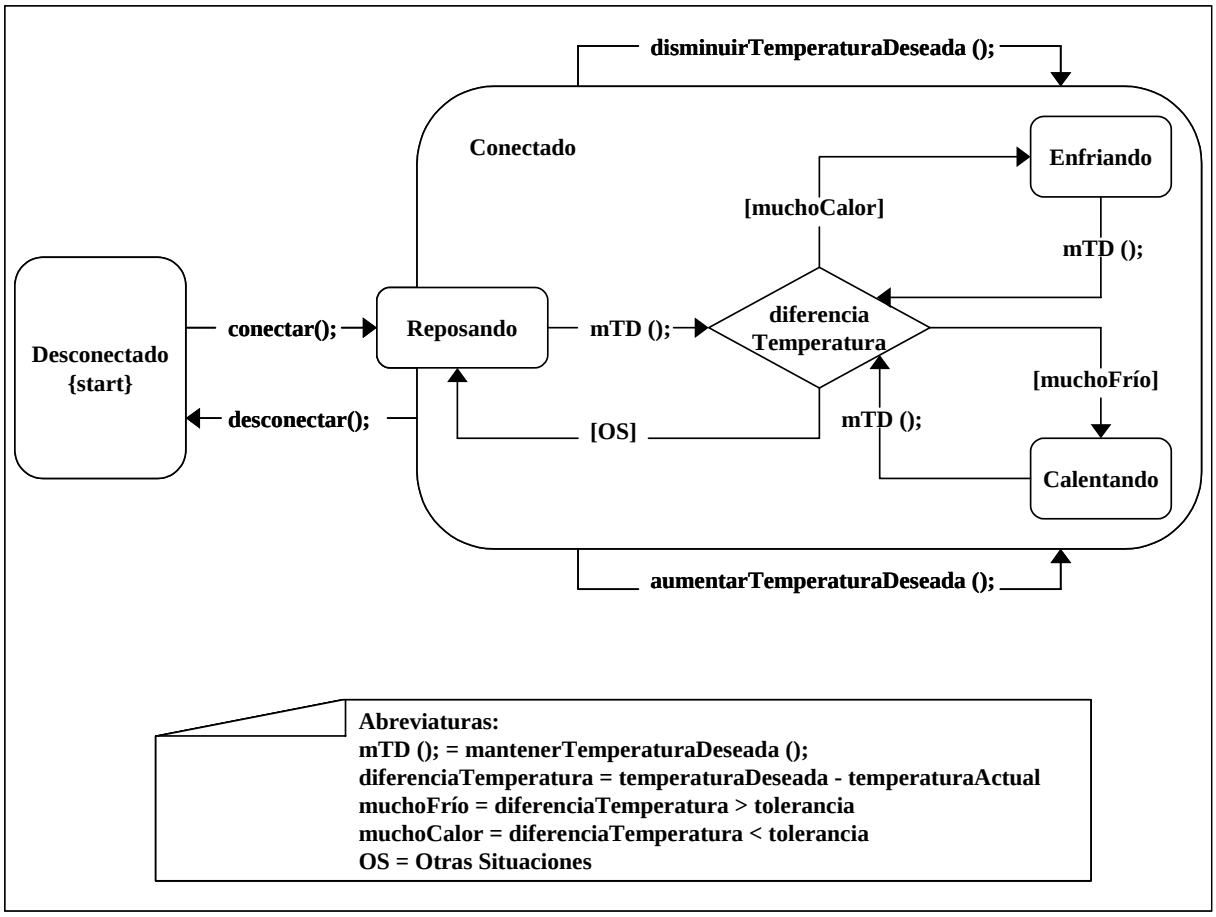


Figura 1.18. Ejemplo de un diagrama de transición de estado para la instancia de la clase Habitación

1.3. MÉTRICA y los lenguajes de modelado orientados a objetos

Desde hace años los ministerios de la Administración Pública utilizan una metodología propia denominada MÉTRICA, aunque las primeras versiones se basaron en SSADM (*Structured Systems Analysis and Systems Design*) [SSAD95]. Actualmente la última versión de MÉTRICA (versión 3) aborda tanto el desarrollo estructurado (datos + procesos) como el desarrollo orientado a objetos [MÉTR00, p. 2]. Para este último tipo de desarrollo esta metodología incorpora artefactos de otras metodologías orientadas a objetos, así como la mayoría de las técnicas que contempla UML v 1.2. Sin embargo, hay que resaltar que la notación que propone la metodología MÉTRICA para aplicar las técnicas en ningún caso se considera obligatoria, sea bien un desarrollo estructurado o bien orientado a objetos. En el caso concreto de desarrollos OO, donde se ha seguido la notación del lenguaje UML, las nuevas técnicas de diagramación contienen la siguiente nota: "Esta notación es la más habitual, pero MÉTRICA versión 3 no exige su utilización" [MÉTR00, p. 10, 17, 19, 20, 22, 31, 46, 49, 52, 54, 56, 67, 78]. En consecuencia, los profesionales de las empresas de

desarrollo de software pueden utilizar la notación que quieran en la especificación de sus artefactos.

Uno de los objetivos establecidos dentro del marco definido para MÉTRICA v 3 es: "facilitar la comunicación y el entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida" [MÉTR00, p. 1]. Para alcanzar este propósito, en la práctica puede que sea mejor construir modelos utilizando una notación diferente de UML, por ejemplo, la notación COMN del lenguaje OML. Pero antes de comenzar a construir en otra notación es importante explorar si los modelos especificados en OML transmiten la información de forma más segura y eficaz que los correspondientes en UML. Esta exploración consiste en analizar la comprensión que el sujeto extrae a partir de los modelos construidos tanto en OML como en UML, objetivo planteado al final de esta investigación.

PARTE II

ESTADO DE LA CUESTIÓN

Capítulo 2

El paradigma experimental en la ingeniería del software

2.1. Métodos de investigación en la ingeniería del software

Etimológicamente, "investigación" proviene del latín "in" (en) y "vestigium" (pista); su significado es la acción de estar "en la pista", averiguar algo siguiendo un rastro. Según esto, la investigación es la búsqueda del conocimiento de algo que ya no está presente, pero de lo que permanece algún vestigio. Para adquirir dicho conocimiento, en el contexto de la ingeniería del software existen cuatro métodos de investigación [Glas94]:

- El método científico

  Se observa el entorno y se construye un modelo basado en las observaciones. Por ejemplo, un modelo de simulación de una red de telecomunicaciones.

- El método ingenieril

  Se proponen cambios basados en soluciones ya existentes, y luego se evalúan. Por ejemplo, mejoras en la línea de producción de una fábrica.

- El método empírico

  Se propone y se evalúa un modelo mediante estudios empíricos. Por ejemplo, casos de estudio o experimentos.

- El método analítico

  Una teoría formal se propone y luego se compara con observaciones empíricas. Por ejemplo, las leyes de la física.

Según V. Basili [Basi93, p. 5] el método ingenieril y el método empírico se pueden considerar variantes del método científico, que es un enfoque inductivo. La simulación también se puede utilizar como mecanismo para la experimentación [VonM93].

2.1.1. Métodos empíricos

Los estudios empíricos se han estado utilizando en aquellas disciplinas relacionadas con el comportamiento humano, como son la sociología y la psicología. A diferencia de la física, en estas disciplinas es imposible formular leyes. Lo mismo sucede en la ingeniería del software, donde la observación del comportamiento humano es muy importante, puesto que son las personas las que desarrollan el software.

Si un ingeniero del software quiere investigar una técnica, un método o una herramienta informática, puede elegir entre las tres siguientes estrategias de investigación empírica [Pfle95, p. 219]:

- Experimento (*researh in the small*)
- Caso de estudio (*research in the typical*)
- Encuesta (*research in the large*)

La elección de una de estas estrategias empíricas depende de una serie de factores, que se resumen en la Tabla 2.1 y se explican más detalladamente en [Fent97, pp. 119-120] y [Wohl00, pp. 16-17].

Factor	Encuesta	Caso de estudio	Experimento
Control de la ejecución	No	No	Si
Control de la medida	No	Si	Si
Coste de investigación	Bajo	Medio	Alto
Facilidad de réplica	Alto	Bajo	Alto

Tabla 2.1. Factores para seleccionar un método empírico

2.1.2. Necesidad de experimentos

Hacia mediados de los años setenta ya existía constancia de algunos estudios empíricos aislados en la ingeniería del software ([Bela72], [Basi75], [Bela76] y [Shne77]). No obstante, Basili fue quien puso de manifiesto la necesidad de experimentar en dicha disciplina, al considerar la ingeniería del software como un laboratorio de ciencia [Basi86]. Posteriormente se han publicado otros artículos que han seguido apoyando esta necesidad de contrastar empíricamente, por ejemplo, métodos de análisis y diseño de desarrollo de software ([Basi96a], [Kitc96], [Zelk98], [Harr99] y [Pfle99]). Por el contrario, Alan Davis definió la experimentación como superflua [Davi96], afirmando que la intuición era suficiente para adoptar nuevas tecnologías software.

La ausencia de evidencia empírica en la investigación en ingeniería del software se constató en [Zelk97]. Este estudio concluyó que un 20% de las publicaciones en la revista *IEEE Transactions on Software Engineering* no incluía ningún componente de validación empírica y que otro tercio del total de artículos examinados contenían validaciones poco consistentes. Asimismo en un análisis más reciente de la literatura en ingeniería del software, donde se evaluaron 369 artículos de 6 revistas internacionales de esta disciplina (publicados entre los años 1995 y 1999), se concluye que la investigación en IS es "diversa" con respecto al tema y "reducida" con respecto al enfoque y al método de investigación [Glas02, p. 505]. Para justificar la falta de experimentación, los investigadores en el área defienden un conjunto de supuestas falacias, que son discutidas y refutadas en [Tich98]:

- El método científico tradicional no es aplicable
- El nivel actual de experimentación es suficiente
- Con una Demo es suficiente
- Existen demasiadas variables para controlar
- La experimentación demora el progreso
- La tecnología cambia demasiado rápido
- Nunca conseguirás publicarlo.

Por lo tanto, en la comunidad de la ingeniería del software nadie duda sobre la importancia de incluir la práctica empírica como mecanismo para validar nuestras propuestas, siempre y cuando sea viable. Un claro reflejo de que esta realidad empieza a cambiar es la reciente proliferación de libros sobre el proceso experimental en la ingeniería del software ([Wohl00] y [Juri01]) o bien de algún capítulo que aborde dicha temática ([Fent97, capítulos 4 y 15] y [Pfle01, capítulos 12-14]).

2.2. Experimentación en la ingeniería del software

En la ingeniería del software los experimentos que se utilizan se califican como controlados, según la clasificación realizada por [Zel97], "in vitro", empleando la terminología de [Basi96a], cuantitativos y formales, según los tutoriales de [Kitc96] y [Pfle95] respectivamente. Esta catalogación de los experimentos nos da una idea de que la experimentación no es una tarea sencilla, sino que requiere un desarrollo de actividades bien planificadas. Pero antes de profundizar en el proceso que conlleva conducir un experimento, es necesario definir en los siguientes apartados la terminología de algunos de sus componentes: *variables*, *tratamientos*, *unidades* y *sujetos experimentales*.

2.2.1. Variables

Las variables protagonistas en un experimento formal son la variable independiente (*state variable* [Pfle95] o *predictand variable* [Kish95]) y la dependiente (*response variable* [Pfle95] o *predictor variable* [Kish95]). Sin embargo, aunque la variable independiente es sobre la que se hipotetiza que influirá en la dependiente, existen otras de gran importancia que también representan una "fuente de variación" en dicha variable [Kish, 1995, p. 5]:

- controlada (*controlled*): es una variable extraña que puede ser adecuadamente controlada mediante el diseño experimental.
- enmascarada (*disturbing*): es una variable extraña no controlada que tiene la propiedad de variar simultáneamente con los cambios en los niveles de la variable independiente. Debido a ello rivaliza con la independiente como posible causa. También recibe el nombre de variable confundida, debido al anglicismo *confounded variable* [Pfle95].
- aleatoria (*randomized*): es una variable extraña no controlada que se trata como un error aleatorio. La aleatorización se puede considerar como una forma de control experimental, pero distinta de las que se aplican en las variables controladas.

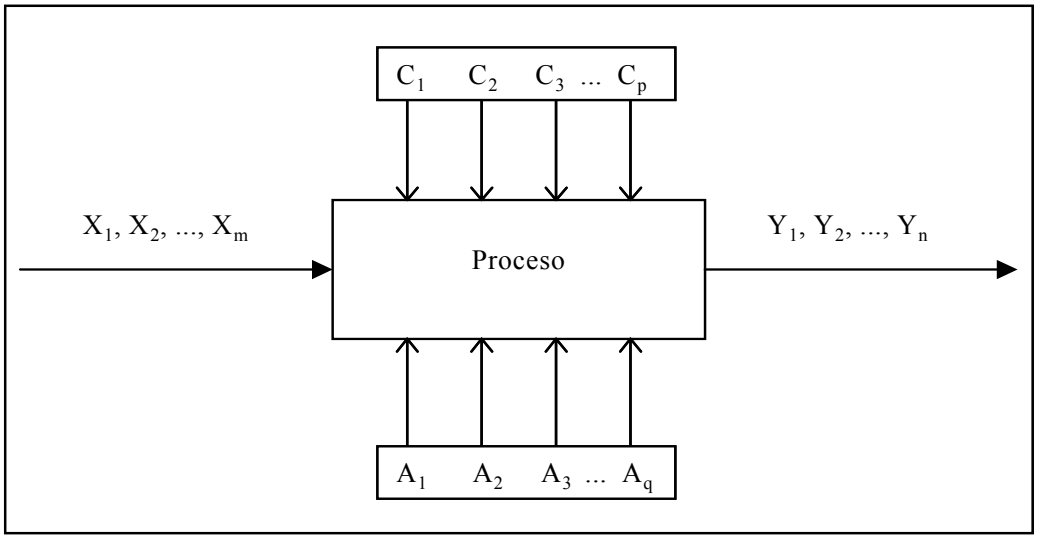


Figura 2.1. Variables de un experimento controlado

Después de identificar las variables de un experimento según la clasificación anterior, debemos minimizar el sesgo (*bias*) de las enmascaradas que sean conocidas. Para ello, se perfilan dos soluciones: bien pasan a ser controladas (C_i) mediante alguna técnica de control, la cual depende del tipo de diseño experimental, o bien se consideran aleatorias (A_j), convirtiéndose en una fuente de error debido al azar (ver Figura 2.1). Está claro, entonces,

que las técnicas para controlar las variables extrañas sirven para disminuir los errores aleatorios, o reducir los efectos de las enmascaradas, o ambos. Y, por último, una vez bien definidas todas las variables, debemos especificar en qué escala (nominal, ordinal, intervalo o ratio) van a ser medidas cada una de ellas, con el fin de aplicar el test estadístico adecuado en el momento del análisis [Fent97, p. 58].

En definitiva, un experimento es un proceso con variables independientes (X_1, X_2, ..., X_m) tan poderosas como sea posible, gracias a la aplicación de un control interno casi perfecto, y variables dependientes (Y_1, Y_2, ..., Y_n) lo suficientemente sensibles para detectar las diferencias.

2.2.2. Tratamientos, unidades y sujetos

Un experimento tiene por objeto descubrir las relaciones existentes entre las variables, en concreto, determinar el efecto de la variable independiente, también denominada factor, sobre la dependiente. Cada factor se divide en un número determinado de niveles. Un tratamiento se corresponde con un nivel o valor particular de un factor. Por otra parte, las unidades y los sujetos experimentales son, respectivamente, los objetos materiales y las personas participantes a los que se les aplica un tratamiento. Las características, tanto de los objetos como de los sujetos, también pueden ser variables independientes en el experimento.

Por ejemplo, en [Basi87] los objetos fueron programas escritos en Fortran y participaron tanto estudiantes como profesionales del mundo informático. Se estudiaron tres factores: técnica de verificación, tipo de programa y nivel de experiencia, cada uno con 3 niveles. Por lo tanto, se obtuvieron 3 x 3 x 3 = 27 condiciones experimentales o tratamientos diferentes.

2.3. Proceso experimental

Al igual que construir software requiere un proceso de desarrollo compuesto por un conjunto de actividades (tales como, análisis, diseño, implementación, etc.), conducir un experimento implica seguir varios pasos en un orden establecido. Lógicamente, estos pasos engloban el proceso experimental necesario para que un experimento controlado proporcione resultados útiles y significativos ([Fent97] y [Wohl00]). Las actividades que constituyen este proceso experimental se muestran en la Figura 2.2.

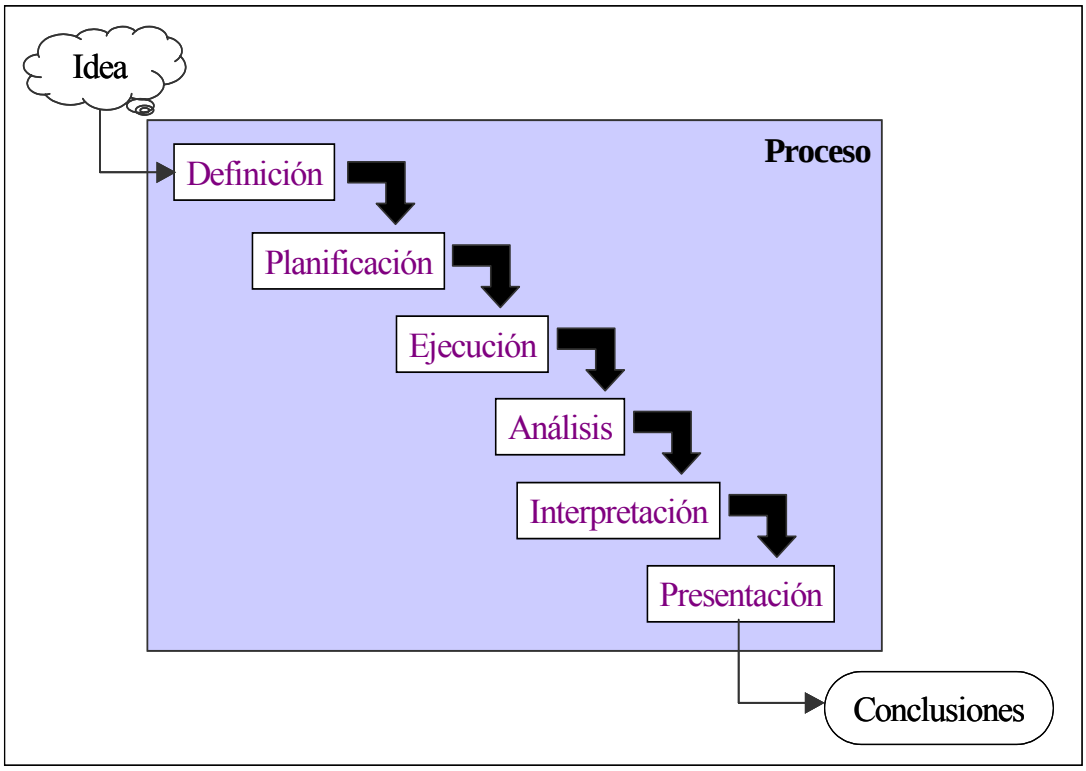


Figura 2.2. Proceso experimental

A priori, el proceso experimental es parecido al modelo "en cascada" del proceso de desarrollo de proyectos software, pero no debe interpretarse de la misma manera. Mientras que en un proceso "en cascada" o secuencial se debe terminar una fase antes de empezar con la siguiente, el proceso experimental es iterativo, pues puede ser preciso volver atrás y refinar una actividad anterior. Por supuesto, estas iteraciones sólo están permitidas en las fases de definición y planificación, siempre que no haya comenzado la ejecución del experimento.

La concepción de una **idea** es el punto de partida de todo proceso experimental. A partir de dicha idea, la primera fase es la **definición** del objetivo del estudio empírico. Para la formulación del objetivo vamos a recurrir al paradigma Objetivo/Pregunta/Métrica (GQM: *Goal/Question/Metric*). Mientras que la fase de definición determina "porqué" se realiza el experimento, la fase de planificación debe organizar "cómo" será ejecutado. La actividad de **planificación** comprende seis tareas, que deben ser cuidadosamente estudiadas: selección de las variables, planteamiento de las hipótesis, selección de los sujetos, preparación de los materiales, elección del diseño en función de las variables e hipótesis y, por último, evaluación de la validez del experimento. Para que la **ejecución** del estudio sea correcta se debe llevar a cabo de acuerdo al plan y diseño experimental. Los datos recogidos serán examinados en la fase de **análisis** mediante el contraste estadístico adecuado. En la siguiente

fase de **interpretación** se explicarán los resultados, teniendo en cuenta el rechazo o aceptación de las hipótesis nulas planteadas en la etapa de planificación. Por último, la fase de **presentación** incluye la documentación tanto de los resultados (por ejemplo, en un informe técnico) como de los materiales experimentales, con el fin de crear "paquetes de laboratorio" fácilmente replicables. El objetivo final es conseguir **conclusiones** fiables.

En los siguientes apartados vamos a profundizar en aquellas actividades más importantes del proceso experimental en la ingeniería del software. Posteriormente, en los capítulos 5 a 8 vamos a especificar dichas actividades para cada uno de los experimentos controlados cuantitativos de la investigación.

2.3.1. Definición del objetivo

2.3.1.1. Método GQM

El método Objetivo/Pregunta/Métrica, también denominado GQM (*Goal/Question/Metric*), sirve para definir e interpretar un conjunto de objetivos de un proyecto o de una organización usando la medición de sus productos, procesos o recursos [Basi94]. El paradigma GQM se trata de un enfoque arriba-abajo, donde primero se formulan los objetivos generales del proyecto y, luego, en base a ellos se desarrollan las preguntas, a las que se asocian métricas para obtener las respuestas. Hoy en día, su utilización todavía sigue vigente, pues tanto [VanS99] como vía web [GQM02] proporcionan guías de aplicación práctica del método GQM.

El enfoque GQM ofrece una plantilla para la correcta formulación del objetivo. Esta plantilla es una herramienta útil en la experimentación, pues facilita la definición del objetivo de cualquier estudio empírico. Los cinco parámetros que constituyen la plantilla están encerrados entre "<" y ">".

Analizar <*objeto(s) de estudio*>

con el fin de <*propósito*>

con respecto a su <*aspecto de calidad*>

desde el punto de vista del <*perspectiva*>

en el contexto de <*entorno*>.

Para entender y completar esta plantilla en la Tabla 2.2 se exponen algunos valores que pueden tomar los diferentes parámetros [Wohl00, p. 45].

Objeto(s) de estudio	*Propósito*	*Aspecto de calidad*	*Perspectiva*	*Entorno*
Producto	Caracterizar	Efectividad	Desarrollador	1. Sujetos
Proceso	Monitorizar	Coste	Programador	Estudiantes
Modelo	Evaluar	Facilidad de	Gestor del proyecto	Profesionales
Métrica	Predecir	mantenimiento	Investigador	2. Objetos
Teoría	Controlar	Comprensibilidad	Cliente	Complejidad
Lenguaje	Mejorar		Usuario	Dominio
	Validar			

Tabla 2.2. Ejemplos de los parámetros de la plantilla GQM

2.3.2. Planificación

A continuación, se describen los seis pasos en que se divide la planificación de un experimento formal.

2.3.2.1. Selección de las variables

Una vez definido el objetivo según la plantilla GQM, debemos elegir las variables que vamos a manipular (variables independientes) y que vamos a medir (variables dependientes) en el experimento. Para vincular el objetivo con las hipótesis, los parámetros de la plantilla GQM se relacionan con las variables que intervienen en las hipótesis. Así pues, los factores que caracterizan el <*objeto de estudio*> constituyen las variables independientes, mientras que las variables dependientes miden el <*aspecto de calidad*>. Asimismo para estas variables hay que escoger la escala de medida apropiada [Fent97, p. 59].

Además de seleccionar estas variables, hay que tener en cuenta la existencia de otro tipo de variables, que ya se han explicado en el apartado 2.2.1, cuya forma de control influye en la elección del diseño experimental.

2.3.2.2. Planteamiento de las hipótesis

En todo experimento se plantean dos tipos de hipótesis:

- Hipótesis nula (H_0): asume que las diferencias son nulas entre dos o más tratamientos de un factor (por ejemplo, entre dos o más técnicas cuyos efectos se quieran medir) con respecto a la variable dependiente que estamos midiendo (por ejemplo, productividad). Es la hipótesis que se quiere contrastar y es, por tanto, la que se acepta o rechaza como conclusión del contraste.

- Hipótesis alternativa (H_A): sostiene que las diferencias no son nulas. Es la hipótesis que nos sitúa frente a H_0, de forma que si se acepta H_A se rechaza H_0 y, recíprocamente, si se rechaza H_A, se acepta H_0.

Para decidir si hay o no diferencias, se realiza un contraste o prueba de hipótesis, que es un procedimiento estadístico mediante el cual se investiga la aceptación o el rechazo de la hipótesis nula. Cuando se efectúa una prueba de hipótesis, puede acontecer uno de los siguientes casos:

1. Que la H_0 sea cierta y que la prueba la acepte.
2. Que la H_0 sea cierta pero que la prueba la rechace.
3. Que la H_0 sea falsa pero que la prueba la acepte.
4. Que la H_0 sea falsa pero que la prueba la rechace.

Lógicamente el primer y cuarto caso están en lo correcto, mientras que el segundo y tercer caso implican un riesgo. Estos riesgos se refieren a tipos de error, que junto con otros factores, van a explicarse más detalladamente en la fase de análisis experimental (apartado 2.3.4).

2.3.2.3. Sujetos experimentales

En la investigación en la ingeniería del software tropezamos con un aspecto preocupante, que es conseguir sujetos para la experimentación. En aquellas disciplinas donde se estudia el comportamiento humano, como puede ser la sociología, es fácil disponer de sujetos pues, normalmente, no se requiere ninguna experiencia previa. En cambio, la ingeniería del software está limitada en cuanto a recursos humanos se refiere. El hecho se debe a que sólo un pequeño porcentaje de la población posee el conocimiento o la capacidad necesaria para desarrollar software y, en consecuencia, ser apto para la experimentación en este campo. Si en sociología se quiere estudiar empíricamente, por ejemplo, los trastornos del sueño, los estudiantes tanto de esta disciplina como de ingeniería informática serían válidos, salvo que se les pidiera un perfil concreto a la hora de dormir. Sin embargo, la situación contraria no sería correcta, al suponer que los estudiantes de sociología no tienen formación específica en la aplicación de métodos de diseño, programación, etc. informáticos.

Dado que se trata de una cuestión de experiencia, una solución es recurrir a los profesionales de la empresa. Pero, en general, las industrias de desarrollo de aplicaciones software aspiran a conseguir el triángulo de objetivos coste-tiempo-calidad, minimizando el precio y el tiempo invertidos y maximizando la calidad de sus productos o servicios. Estos

objetivos no son solamente factores de competencia entre las empresas de este campo, sino también de supervivencia de las mismas. Por supuesto, la experimentación bien planificada ayudaría a la consecución de dichos objetivos, si bien esta práctica todavía no ha calado con suficiente fuerza dentro de la comunidad profesional en ingeniería del software.

Por lo tanto, si del pequeño porcentaje antes mencionado también se eliminan los sujetos profesionales del mundo empresarial, como último recurso queda experimentar con estudiantes de cursos de ingeniería informática. Existe un estudio preliminar que indica que los estudiantes se pueden utilizar para realizar ciertas tareas experimentales en lugar de los profesionales bajo ciertas condiciones [Höst00], pero no queda claro cómo generalizar los resultados de los experimentos basados en estudiantes a los ingenieros de software profesionales [Harr00].

Aunque para suplir la dificultad de encontrar profesionales se utilizan estudiantes de ingeniería informática, no siempre es posible disponer de todos ellos. Entre las razones podemos citar las siguientes:

- Falta de experiencia: los alumnos de primer curso todavía no cuentan con los conocimientos necesarios. Por ejemplo, un requisito para conducir un experimento sobre la efectividad de dos técnicas de diseño sería que los alumnos hubiesen finalizado asignaturas de cursos anteriores.
- Ausencia de motivación: sencillamente hay estudiantes que no quieren participar.
- Dependencia de la matriculación: el número de alumnos que puede participar está limitado al número de matriculados en una asignatura concreta. La mayoría de los experimentos que se efectúan dentro de ingeniería del software se proponen a los estudiantes como parte de las prácticas de una asignatura. Por ejemplo, la asignatura "Enfoque Cuantitativo a la Ingeniería y Gestión del Software" impartida por V. Basili en la Universidad de Maryland desde 1999 y también ofertada en el 2002 incluye la realización de experimentos (ver http://www.cs.umd.edu/class/fall2001/cmsc735/), cuyos resultados forman parte de la investigación de las tesis de sus doctorandos(as).

Algunas de estas u otras razones han influido en que el número de sujetos experimentales que colaboran en los estudios empíricos no siempre sea elevado. Precisamente, en los experimentos de [Cart98], [Lait97], [Basi96b] y [Zend01] participaron 10, 11, 12 y 15 sujetos respectivamente. Esta escasa participación contrasta con la del estudio empírico de [Finn98], donde asistieron 147 estudiantes para evaluar la comprensión de especificaciones escritas en lenguaje Z.

2.3.2.4. Objetos experimentales

En primer lugar, hay que distinguir que los "objetos" y los "objetos de estudio" (parámetro de la plantilla GQM) son conceptos diferentes. Los objetos experimentales se refieren a los artefactos software que se confeccionan para ser utilizados como material experimental. Además de los objetos propiamente experimentales, es necesaria la elaboración de material de entrenamiento, que incluya las instrucciones para que los sujetos se familiaricen con los instrumentos y para que, luego, no surjan dudas que entorpezcan las tareas experimentales.

Un punto relevante de todo proceso experimental es la recopilación de datos. Para ello los sujetos tienen que anotar los datos solicitados en algún tipo de formulario. A la hora de confeccionar el formato de un cuestionario hay que pensar si deberían ser personalizados o anónimos. Cuando se trata de un experimento compuesto de varias sesiones donde cada sujeto va a ser medido en diferentes tratamientos, resulta apropiado preparar un conjunto personal de instrumentos para cada participante. En cambio, si la ejecución de un experimento es suficiente con una única sesión y no requiere estudios posteriores, es adecuado utilizar cuestionarios anónimos. Como contrapartida, luego no hay posibilidad de contactar con el participante si no ha cumplimentado alguna casilla de forma clara.

En todos los experimentos pertenecientes a la investigación de esta tesis se utilizaron cuestionarios personalizados y en cada sesión a cada participante se le distribuyó un "kit" conteniendo los instrumentos necesarios para desarrollar las tareas experimentales.

2.3.2.5. Diseño experimental

Gran parte de las ideas sobre diseño experimental nacen de la investigación en la agricultura, pionera en el diseño de experimentos. De hecho, el uso de los términos "filas" y "columnas" en un diseño de bloques aleatorios procede del concepto de dividir un campo en parcelas de tierra de igual tamaño y proteger el experimento de las diferencias en la fertilidad del suelo de distintas parcelas. En consecuencia, los diseños estándares están relacionados con esta área de aplicación, quedando incluso reminiscencias en la denominación de algunos de ellos, como es el caso, del diseño de parcelas divididas.

Al igual que otras disciplinas han desarrollado diseños acorde con sus propios requisitos, por ejemplo, experimentos "doble ciego" en medicina, para los investigadores de ingeniería del software es importante identificar cuáles son los diseños básicos que resuelven los problemas específicos de los experimentos de esta disciplina. Por ello, en los dos siguientes

apartados 2.3.2.5.1 y 2.3.2.5.2 vamos a describir los principios y los tipos básicos de diseño experimental.

2.3.2.5.1. Principios de diseño

En la relación causa-efecto que la variable independiente ejerce sobre la dependiente pueden mediar otras fuentes extrañas de variación. Algunas de estas fuentes de error se deben a la variabilidad que existe entre los participantes, la diferencia de concentración debido a ruidos molestos que los sujetos de un aula pueden sufrir con respecto a los que participan en otra clase, los fallos técnicos en el funcionamiento de los ordenadores, etc. Como, en realidad, la eliminación completa no es posible, hay que intentar disminuir esta variabilidad controlando tantas variables como sea posible. Mediante la aplicación de técnicas de control, tales como asignación aleatoria, bloqueo y equilibrado, se logra minimizar el error experimental. Los otros principios de diseño, replicación y selección aleatoria, permiten asegurar la validez de los resultados.

a) Aleatorización

En este punto es importante distinguir entre selección aleatoria (*random sampling*), utilizada en las encuestas, y asignación aleatoria (*random assignment*) en los métodos experimentales.

- **Selección aleatoria**

  En la selección aleatoria cada uno de los grupos debe ser una muestra extraída al azar de la población, lo que nos asegura una buena representación de los sujetos respecto a dicha población y, por consiguiente, una buena generalización de los resultados. Sin embargo, la selección aleatoria no es una estrategia utilizada en la experimentación controlada, ya que reunir, por ejemplo, a todos los alumnos del país que estudian ingeniería informática requiere mucho tiempo y dinero. Aunque nos interesa la representatividad del colectivo, nuestra principal preocupación es mostrar la influencia de la variable independiente en la dependiente y, para ello, asumimos que esta influencia es independiente de variables tales como la localización geográfica de los estudiantes o el tipo de facultad a la que asisten.

- **Asignación aleatoria**

  En este caso, la aleatorización se refiere a que los grupos de sujetos correspondientes a los distintos niveles de la variable independiente se forman mediante asignación aleatoria. Esta técnica constituye una forma de controlar las variables extrañas, por

equilibrado, que están presentes en los sujetos: inteligencia, experiencia, etc. Por ejemplo, en la réplica efectuada por [Daly94] aunque los estudiantes fueron asignados de forma aleatoria al grupo de control o al grupo experimental, un análisis posterior de los resultados de un test sobre programación, realizado antes del experimento, puso en evidencia la falta de control sobre la capacidad intelectual de los estudiantes. En uno de los grupos figuraban 8 sujetos de los 12 considerados como mejores y en el otro tan sólo 4. Esta variación pudo ser la causa de que los resultados fueran contrarios a los obtenidos en el experimento original.

b) Bloqueo

En el experimento de [Daly94] la variable extraña, habilidad individual en la programación, tenía una relación importante con la variable dependiente. Además, no estaba equilibrada en los dos grupos mediante el azar. En consecuencia, ni el diseño fue el adecuado ni los resultados fiables. La solución para evitar estos problemas es bloquear dicha variable.

La técnica de control con la que se consiguen grupos emparejados (*matched*) se denomina bloqueo, ya que se forman bloques en una variable relacionada con la variable dependiente. Ahora la variable no sería extraña, sino bloqueada. Por ejemplo, esta técnica se aplicó en [Tryg97], donde se midió a los estudiantes en su habilidad con el lenguaje C++ para asegurar una distribución homogénea en los dos grupos experimentales.

c) Equilibrado

Cuando en cada grupo hay el mismo número de individuos, se dice que el diseño experimental está equilibrado. En caso contrario, se dice que está desequilibrado. Por ejemplo, la mortalidad de sujetos en las sucesivas sesiones de ejecución del experimento de [Lait97] ocasionó que en cada celda no hubiese el mismo número de personas, lo que complicó el análisis estadístico de los resultados.

d) Replicación

La replicación es la repetición de un experimento. Para reproducirla de la forma más estricta posible es preciso generar paquetes de laboratorio con todo el material experimental. Sin embargo, la réplica de un estudio empírico no implica necesariamente la repetición bajo las mismas condiciones. Los diferentes tipos de réplicas que se pueden ejecutar en ingeniería del software se encuentran clasificados en [Basi99, pp. 469-470]: réplicas estrictas, réplicas que varían las variables independientes o dependientes y réplicas que cambian el contexto de experimentación.

2.3.2.5.2. Tipos de diseño

Los libros clásicos, tales como [Coch80] y [Box88], aún hoy en día son muy útiles para abordar el tipo de diseño experimental en cualquier disciplina. Aparte de estos libros, la práctica idónea sería que para la construcción de diseños se consultasen primero textos de diseño experimental específicos de cada disciplina, como pueden ser en el caso de la sicología [Judd91] y [Shau97]. En el contexto de la ingeniería del software los tutoriales de [Pfle95] y [Kitc96] tan sólo orientan sobre cómo derivar un diseño a partir del número de factores y de niveles. Esta orientación es muy elemental, pues se basa en dos relaciones básicas que se pueden establecer entre los factores: cruce o anidamiento. Tanto estos tutoriales como el libro de [Wohl00, pp. 54-62] carecen de una taxonomía lo suficientemente exhaustiva como para abarcar todos los tipos de diseños de experimentos formales empleados en ingeniería del software. Para contribuir a suplir esta carencia, en [Oter00, pp. 55-70] aparecen clasificados alrededor de 20 artículos de ingeniería del software, que fueron seleccionados entre los años 1994 y 1999 y que incluyeron experimentos controlados como medio de investigación.

Los tipos de diseño que se utilizaron en los experimentos realizados en la investigación de esta tesis se presentan y se describen en los apartados 4.5.1 y 4.5.2.

2.3.2.6. Validez experimental

Mediante un buen diseño experimental, debemos maximizar las propiedades de los experimentos formales, que son:

- Validez de la conclusión o Fiabilidad

  Se dice que un experimento es fiable cuando, al repetirse, se obtienen los mismos resultados. Para ello es preciso el control y la replicación, bien interna o externa, del experimento.
- Validez constructiva

  Es el grado con que tanto las variables independientes como dependientes reflejan o miden de manera certera las hipótesis del experimento.
- Validez interna

  Es el grado de seguridad con el que se puede atribuir que los cambios de la variable dependiente son debidos a la influencia de la variable independiente que se estudia.
- Validez externa

  Es el poder de generalización de los resultados obtenidos en el laboratorio a las condiciones normales.

En [Judd91, pp. 30-35], [Kish95, pp. 103-111] y [Wohl00, pp. 63-74] se citan los factores que hacen peligrar la validez del diseño de experimentos controlados. En un principio, el trabajo de [Camp73] define una primera lista de "12 amenazas" a la validez interna y externa de un experimento. Luego [Cook79] amplia esta lista, añadiendo amenazas a los cuatro tipos de validez experimental citados anteriormente.

Para maximizar las propiedades de los experimentos es necesario minimizar o amortiguar la presencia de las amenazas a la validez de los resultados experimentales. Por ello, a continuación, vamos a describir aquellas amenazas más relevantes en los experimentos de nuestro campo de investigación.

a) Validez de la conclusión

- Baja potencia estadística

  Cuando la potencia a posteriori de un contraste estadístico es baja, la probabilidad de que se obtengan conclusiones erróneas es alta. Por ejemplo, supongamos que la potencia estadística es del 30%. Esto significa que si el experimento se repitiese 100 veces, en 30 de ellas se rechazaría correctamente la hipótesis nula, mientras que en las 70 restantes el rechazo de la hipótesis sería erróneo.
- Violación de los supuestos de los contrastes estadísticos
- Heterogeneidad aleatoria de los sujetos

  Cuando los sujetos se asignan al azar a las condiciones o tratamientos experimentales se corre el riesgo de crear grupos élite, de mejor rendimiento en alguna de ellas. Un ejemplo donde se produjo esta amenaza fue en [Daly94], así como en [Cart98].

b) Validez interna

- Selección

  Los efectos de selección se deben a las variaciones naturales en el rendimiento de cada sujeto. Algunos ejemplos son la experiencia previa al experimento programando en un cierto lenguaje, la inteligencia o la habilidad (*skill*).
- Madurez o Aprendizaje

  Los efectos de madurez son causados por el paso del tiempo, concretamente por el cansancio o por el aprendizaje de los sujetos según avanza el experimento.
- Historia

  Los efectos de la historia se deben a eventos específicos que podrían ocurrir durante el transcurso de un experimento. Algunos ejemplos de ello son: el investigador ha caído

enfermo y debe ser reemplazado, un fallo en el ordenador o cualquier otra interrupción debido a una fuente no deseada.

- Instrumentación

  Los efectos de la instrumentación pueden tener su origen en las diferencias en los materiales experimentales empleados.

- Mortalidad

  Los efectos de mortalidad se deben a la pérdida de sujetos a lo largo de un experimento. Un ejemplo es la falta de motivación, originada por el trabajo reiterativo y su larga duración.

- Práctica u orden de presentación

  Los efectos de la práctica o del orden de presentación se deben a la cantidad de práctica que se lleva realizada antes de una presentación correspondiente a un tratamiento determinado. Para controlar el efecto de la práctica hay dos estrategias. La primera de ellas se denomina control mediante aleatorización, que sólo es válida para controlar. En cambio, la segunda se utiliza también para medir el efecto de la práctica, considerándola como si fuera otra variable independiente. En esta segunda estrategia cuando los niveles de la variable independiente son dos, un diseño reequilibrado (apartado 4.5.1.1) es el adecuado; pero si son tres o más, el diseño se llama cuadrado latino (apartado 4.5.1.2).

2.3.3. Ejecución

Previamente a la ejecución de un experimento, los investigadores se deben plantear si es preciso que los sujetos asistan primero a una etapa de formación. En algunos experimentos en ingeniería del software sí es necesario, mientras que en otros no. Sin embargo, un segundo paso muy aconsejable es que los sujetos deben participar en una etapa de entrenamiento, que les permita familiarizarse con todo el material experimental: herramientas de software, formatos de cuestionarios, etc.

En definitiva, la ejecución de un experimento controlado debe realizarse de forma rigurosa, de acuerdo al diseño planteado en la fase anterior de planificación. Durante la ejecución de un experimento, que puede abarcar varios días, los sujetos efectúan sus tareas experimentales según los diferentes tratamientos que les corresponden.

2.3.4. Análisis de los datos

En la fase de análisis estadístico se examinan los datos recogidos tras finalizar la ejecución del experimento. Normalmente los papeles de investigador y de estadístico son realizados por la misma persona, lo que a veces conlleva fallos en el análisis de los datos. En [Kitch02] se encuentran recogidas una serie de consejos o directrices para cada una de las fases de investigación empírica en ingeniería del software. Precisamente una razón importante para la redacción de estas directrices es la constatación, a través del estudio de un amplio conjunto de experimentos, de que los investigadores en el área informática cometen fallos estadísticos, así como interpretaciones erróneas de los datos. En [Kitc02, p. 729] se cita un ejemplo, en el que el método de análisis no fue el apropiado para el diseño del experimento realizado por [Port95], lo que implicó encontrar resultados significativos donde realmente no existían.

El objetivo que se pretende conseguir en esta fase es probar de forma correcta la veracidad o falsedad de las hipótesis nulas formuladas en la etapa de planificación. Por ejemplo, la hipótesis nula de la Tabla 2.3 es "H_0: $\mu \leq 50$", que establece que el valor desconocido de la media poblacional es inferior o igual a 50 (valor hipotético de la media poblacional o μ_0), mientras que la alternativa es "H_A: $\mu > 50$". Ambas hipótesis son exclusivas. El proceso de elegir entre la hipótesis nula y la alternativa se denomina contraste o prueba de hipótesis.

Para realizar una prueba de hipótesis se necesita: (1) disponer de un conjunto de hipótesis, (2) elegir un contraste estadístico, y (3) fijar una serie de valores para la toma de decisiones. Un contraste estadístico se clasifica en parámetrico, si los parámetros se pueden medir en una escala intervalo o ratio, y no paramétrico, cuando las métricas sólo están en una escala nominal u ordinal. Algunos contrastes estadísticos son: t-test, Chi-2, Kruskal-Wallis, ANOVA, Mann-Whitney, etc. Una introducción a los contrastes paramétricos y no paramétricos se puede encontrar en [Wohl00, pp. 96-112].

Una vez que se prueba la hipótesis nula, la decisión del investigador puede ser bien correcta o incorrecta. Una decisión incorrecta se puede tomar de dos formas. El investigador puede rechazar la hipótesis nula cuando es cierta (caso 2 del apartado 2.3.2.2), lo que se denomina error de tipo I. La otra posibilidad es que el investigador acepte la hipótesis nula siendo falsa (caso 3 del apartado 2.3.2.2), lo que se llama error de tipo II. En la Tabla 2.3 se muestran los dos tipos de decisiones correctas y los dos tipos de errores (decisiones incorrectas) que el investigador puede tomar en los supuestos de que el valor desconocido de la media poblacional sea: $\mu = 50$ y $\mu = 55{,}5$.

La probabilidad de cometer un error de tipo I es igual a α. Este valor de α, también denominado nivel de significación, es establecido por el investigador. En la Tabla 2.3 como α = 0,05, la probabilidad de tener un error de tipo I es 0,05.

La probabilidad de cometer un error de tipo II es igual a β. La probabilidad de que una hipótesis sea rechazada correctamente, es decir, la potencia, es igual a 1 – β. Ambas probabilidades vienen determinadas por las siguientes variables: el nivel de significación especificado, α; el tamaño de la muestra, n; la desviación estándar, σ; y la diferencia entre μ y μ_0. Para calcular la probabilidad de cometer un error de tipo II y la potencia, es necesario conocer el valor real de la media poblacional μ. Como este valor es desconocido, se toma como valor práctico de la media poblacional, la media muestral $\bar{y}$. Lógicamente la muestra seleccionada debe ser representativa de la población a investigar. En la Tabla 2.3 hemos supuesto dos valores prácticos de la desconocida media poblacional, que son 50 y 55,5.

		Situación Real	
		H_0 verdadera $\mu = 50$	H_0 falsa $\mu = 55,5$
Decisión del investigador	Aceptar H_0 H_0: $\mu \leq 50$	Aceptación correcta Probabilidad = 1 – α 1 – α = 0,95	Error tipo II Probabilidad = β β = 0,30
	Rechazar H_0 H_A: $\mu > 50$	Error tipo I Probabilidad = α α = 0,05	Rechazo correcto Probabilidad = 1 – β 1 – β = 0,70

Tabla 2.3. Diferentes tipos de decisión para las situaciones de μ = 50 y μ = 55,5

La prueba o contraste de hipótesis implica una serie de convenios, que luego el investigador puede adoptar o no en el diseño de sus experimentos. Uno de ellos es fijar la probabilidad de cometer un error de tipo I menor o igual a 0,05. Otro convenio es que la probabilidad de cometer un error de tipo II sea menor o igual a 0,20. Si β = 0,20, la potencia de la prueba de hipótesis es 1 – β = 0,80. Muchos investigadores consideran una potencia de 0,8 como la potencia mínima aceptable. La aceptación de estos dos convenios, β = 0,20 y α = 0,05, implica que los errores de tipo II son menos importantes que los errores de tipo I. Si consideramos el ratio *p(error tipo II)/p(error de tipo I)*, bajo las condiciones anteriores el resultado del cociente es *0,20/0,05 = 4*. Es decir, un investigador que adopta α = 0,05 y β = 0,20 está diciendo que un error de tipo I es cuatro veces más serio que un error de tipo II.

Existe otro convenio relacionado con el error de tipo I, que es fijar α = 0,01. Este criterio de adoptar un valor numérico más pequeño se basa en la idea de que un error de tipo I es muy

malo y se debe evitar. Pero, desgraciadamente, si la probabilidad de un error de tipo I es cada vez más pequeña, entonces se incrementa la probabilidad de un error de tipo II. Esta situación se puede comprobar en la Figura 2.3, donde se visualizan las regiones que corresponden en el caso de cometer un error de tipo I, α, y un error de tipo II, β. Ambas regiones están separadas por una línea vertical. Si esta línea que corta la parte superior de la región α se mueve hacia la derecha o hacia la izquierda, entonces la región β se hace, respectivamente, más grande o más pequeña.

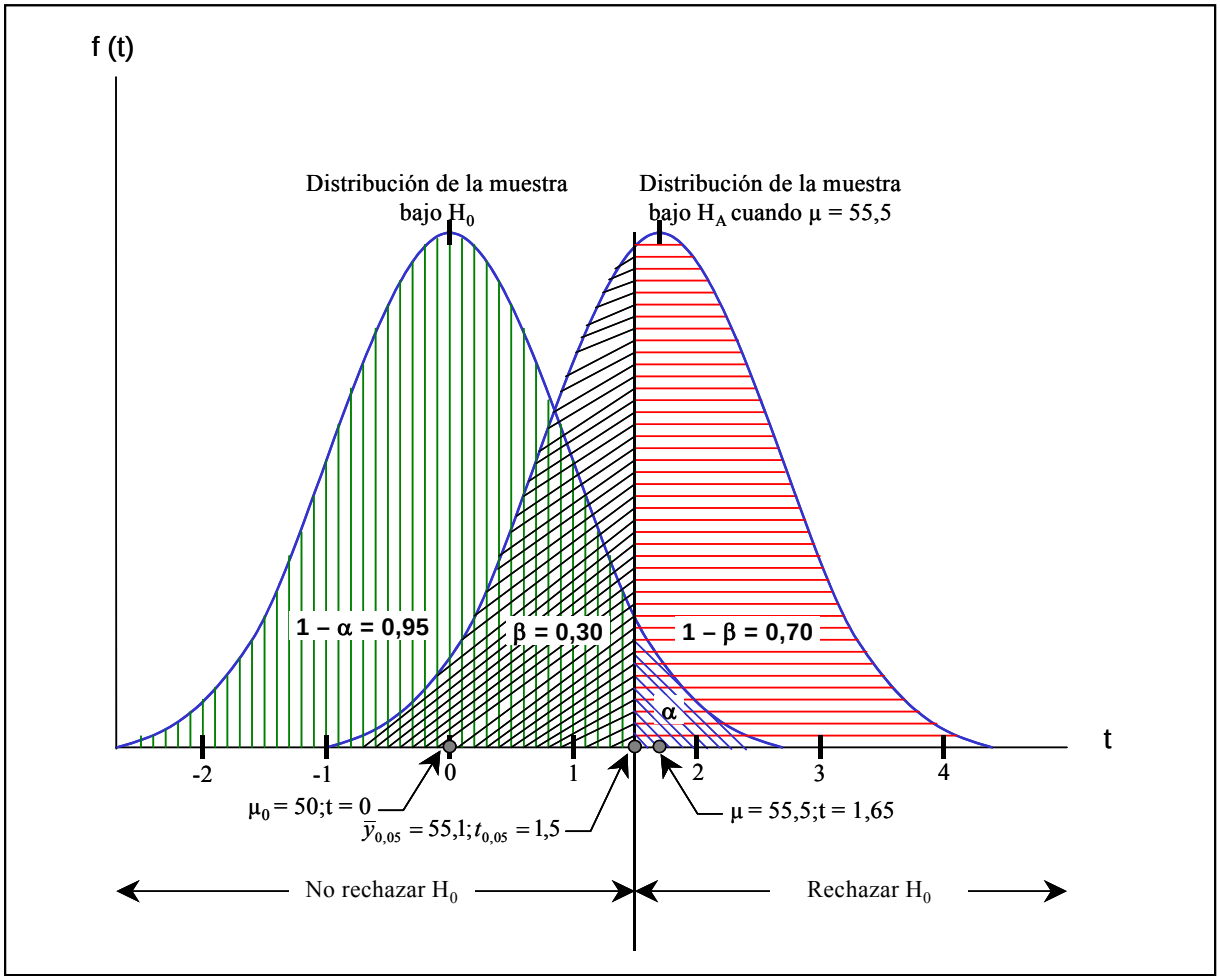


Figura 2.3. Regiones correspondientes a un error de tipo I (α) y a un error de tipo II (β)

Para aceptar o rechazar la hipótesis nula "H_0: $\mu \leq 50$", es decir, que la media poblacional es menor o igual a 50, se emplea la media muestral. El proceso para contrastar dicha hipótesis comprende seis pasos. En algunos de ellos se proporcionan una serie de datos hipotéticos, que coinciden con los de la Tabla 2.3 y la Figura 2.3. Estos pasos son los siguientes:

1º) Establecer las hipótesis: H_0: $\mu \leq 50$ y H_A: $\mu > 50$

2º) Especificar el contraste estadístico: $t = \frac{\bar{y} - \mu_0}{\sigma / \sqrt{n}}$

3º) Especificar el tamaño de la muestra: *n = 30*

4º) Especificar el nivel de significación: *α = 0,05*

5º) Acudir a la tabla *t* para obtener el valor crítico de *t* según el nivel de significación (*α = 0,05*) y los grados de libertad (*n - 1 = 30 – 1 = 29*). Supongamos que el valor crítico es $t_{0,05} = 1{,}5$. Precisamente en la Figura 2.3 hay una recta vertical que corta a las dos campanas de Gauss y al eje de abcisas t en el punto crítico $t_{0,05} = 1{,}5$. También aparece el valor $\bar{y}_{0,05} = 55{,}1$, media de la distribución muestral bajo la hipótesis nula que corresponde al punto crítico $t_{0,05} = 1{,}5$.

6º) Tomar una decisión

Como el valor de μ es desconocido, vamos a suponer como valores prácticos de la media poblacional los que figuran en la Tabla 2.3, que bien podrían ser la media muestral obtenida tras realizar el experimento con una muestra de la población.

a) Aceptar, es decir, no rechazar la hipótesis nula

Para μ = 50 = $\bar{y}$, el valor que se obtiene en el 2º apartado es: *t = 0*. Como se puede apreciar en la Figura 2.3, este valor de t cae a la izquierda de la línea recta del punto crítico (*0 < 1,5*), es decir, cae en la región marcada como no rechazar H_0.

En esta región hay 2 valores: 1 – α y β. Por lo tanto, en este caso se acepta la hipótesis nula con una seguridad del 95% y cometiendo un error de tipo II inferior al 30%.

b) Rechazar la hipótesis nula

Para μ = 55,5 = $\bar{y}$, el valor que se obtiene en el 2º apartado es: *t = 1,65*. Como se puede ver en la Figura 2.3, este valor de t cae a la derecha de la línea recta del punto crítico (*1,65 > 1,5*), es decir, cae en la región marcada como rechazar H_0.

En esta región hay 2 valores: α y 1 - β. Por lo tanto, en este caso se rechaza la hipótesis nula con una potencia del 70% y cometiendo un error de tipo I inferior al 5%.

Como se ha indicado antes, los cuatro factores que determinan la potencia de una prueba de hipótesis son: el nivel de significación, *α*; el tamaño de la muestra, *n*; la desviación estándar, *σ*; y la diferencia entre *μ* y μ_0. Para incrementar la potencia es suficiente con hacer cambios en alguno de estos factores. Tal como podemos comprobar en la Figura 2.3 una forma de aumentar la potencia, 1 – β, es adoptando un valor más alto para el nivel de α, como

puede ser 0,1 o 0,2. Otra posibilidades son: aumentar el tamaño de la muestra, controlar el mayor número posible de variables extrañas mediante el diseño experimental para disminuir la desviación típica o incrementar la diferencia entre μ y μ_0.

En ingeniería del software cabe destacar que en varios experimentos ([Bria97], [Lait97], [Corr00], [Lait00], [Prec02]) se utiliza el valor 0,1 en lugar del 0,05 como nivel de significación. Esta decisión viene justificada por la necesidad de aumentar la potencia del contraste de hipótesis. Por ello, no hay que omitir los dos tipos de error en que se puede incurrir en toda investigación empírica. El error de tipo I, que consiste en rechazar incorrectamente H_0, está bajo nuestro control a través del nivel de significación α. La magnitud β, que mide la probabilidad de cometer un error de tipo II, se controla principalmente mediante el diseño experimental. Si la potencia depende de α, del tamaño de la muestra, n y del tamaño del efecto $\gamma = (\mu - \mu_0) / \sigma$ [Mill97], dados tres de estos valores es fácil averiguar el cuarto. La situación ideal sería que el investigador fijase el nivel de significación, especificase el nivel de potencia deseado y, además, pudiese anticipar el tamaño del efecto. A partir de estos datos se podría calcular el tamaño de la muestra que se necesita para cumplir dichas especificaciones. Pero predecir el tamaño del efecto, que depende de la desviación típica σ y de la diferencia de la medias entre μ y μ_0, en ingeniería del software es casi imposible, debido principalmente a la no existencia de experimentos previos. Teniendo en cuenta que el tamaño de la muestra es fijo y que el tamaño del efecto es un factor difícil de pronosticar, la única posibilidad que queda es aumentar la potencia $1 - \beta$ a expensas de incrementar α (más detalles en [Mill97]).

Aunque se han establecido una serie de convenios, éstos son arbitrarios y, al final, el investigador es quien decide o no aceptarlos. Con respecto a los cuatro experimentos que engloba esta tesis se adoptó fijar $\alpha = 0,1$ y $\beta = 0,2$. Además, antes de ejecutar la réplica del primer experimento se calculó el tamaño de la muestra necesario. En esta ocasión se podía disponer de la media y desviación típica de un experimento previo y así obtener el valor mínimo del tamaño muestral.

En definitiva, en los cuatro experimentos las decisiones correctas se tomarán con una seguridad del 90% en el caso de aceptación de la hipótesis nula y con una potencia del 80% en el supuesto de rechazo de la hipótesis. En consecuencia, el ratio $p(\beta)/p(\alpha) = 0,20/0,10 = 2$ indica que cometer un error de tipo II es 2 veces menos importante que un error de tipo I. Por último, para garantizar que los resultados de los cuatro estudios empíricos se examinan correctamente, los datos se van a analizar de acuerdo al modelo matemático que corresponde

al tipo de diseño experimental de cada uno de ellos y utilizando tres paquetes de software estadístico: JMP, NCSS y SPSS.

2.3.5. Interpretación

Una vez que se ha terminado de analizar los datos, se deben interpretar los resultados obtenidos. Cuando se rechazan hipótesis nulas, la conclusión es que los factores influyen en las variables dependientes. Si, por el contrario, se aceptan, se deduce que las variables independientes no están ejerciendo ninguna influencia sobre las variables respuesta.

Además de disponer de los resultados estadísticos, éstos se pueden combinar con técnicas de visualización gráfica como ayuda a la interpretación de un conjunto de datos. Algunas de las herramientas estadísticas gráficas que se han utilizado en esta investigación son: histogramas, gráficos de barras, gráficos de dispersión, gráficos de caja y bigotes, etc.

En un *gráfico de dispersión* se dibujan nubes de puntos $P\ (x_i, y_i)$ en dos dimensiones. Este tipo de gráfico sirve para mostrar si hay o no dependencias entre variables. En el eje X de la Figura 2.4 se representa la puntuación y en el eje Y el tiempo. Cada punto corresponde a la puntuación y al tiempo totales obtenidos por cada estudiante tras realizar las tres tareas experimentales en el primer estudio $P\ (puntuación_i, tiempo_i)$, siendo i = 1, ..., 18 alumnos.

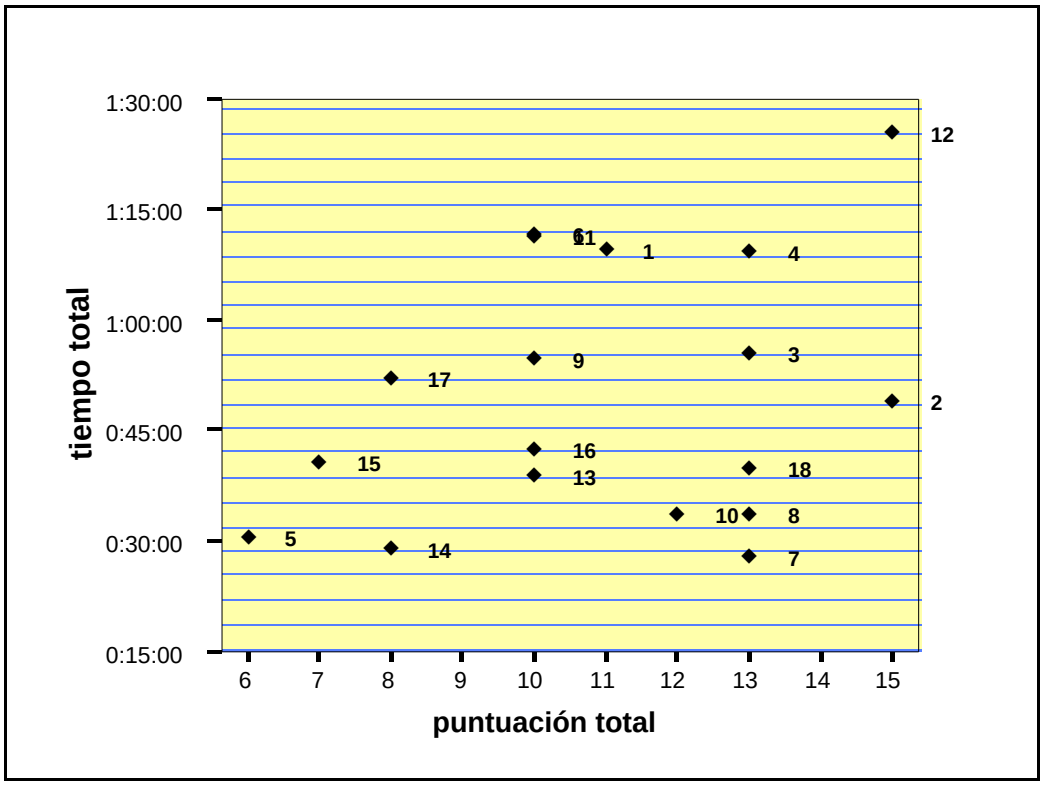


Figura 2.4. Diagrama de dispersión de las puntuaciones y tiempos del primer experimento

En la Figura 2.4 se puede observar que no hay correlación entre la puntuación y el tiempo, es decir, que no hay dependencia. La utilidad de este diagrama de dispersión en la investigación de esta tesis es que nos permite mostrar gráficamente que ambas métricas no son dependientes. Si se hubiese observado que existía una relación lineal entre ambas variables dependientes, sería suficiente con medir a los alumnos con respecto a una de las métricas. Por lo tanto, en cada uno de los cuatro experimentos se han tomado medidas a los estudiantes con respecto a las dos variables mencionadas antes.

En un *gráfico de caja y bigotes* (*box and whisker plot*) se resumen y comparan simultáneamente varios conjuntos de datos, correspondientes a los diferentes grupos en que se pueden subdividir los valores de una variable. Al ser una representación simultánea se puede comparar medias, medianas, rangos de todos los grupos, detectar valores atípicos y estudiar la simetría de los datos. Por ejemplo en la Figura 2.5 se pueden comparar estos valores con respecto a la variable tiempo total en cada uno de los grupos divididos por tipo de aplicación y tipo de diagrama.

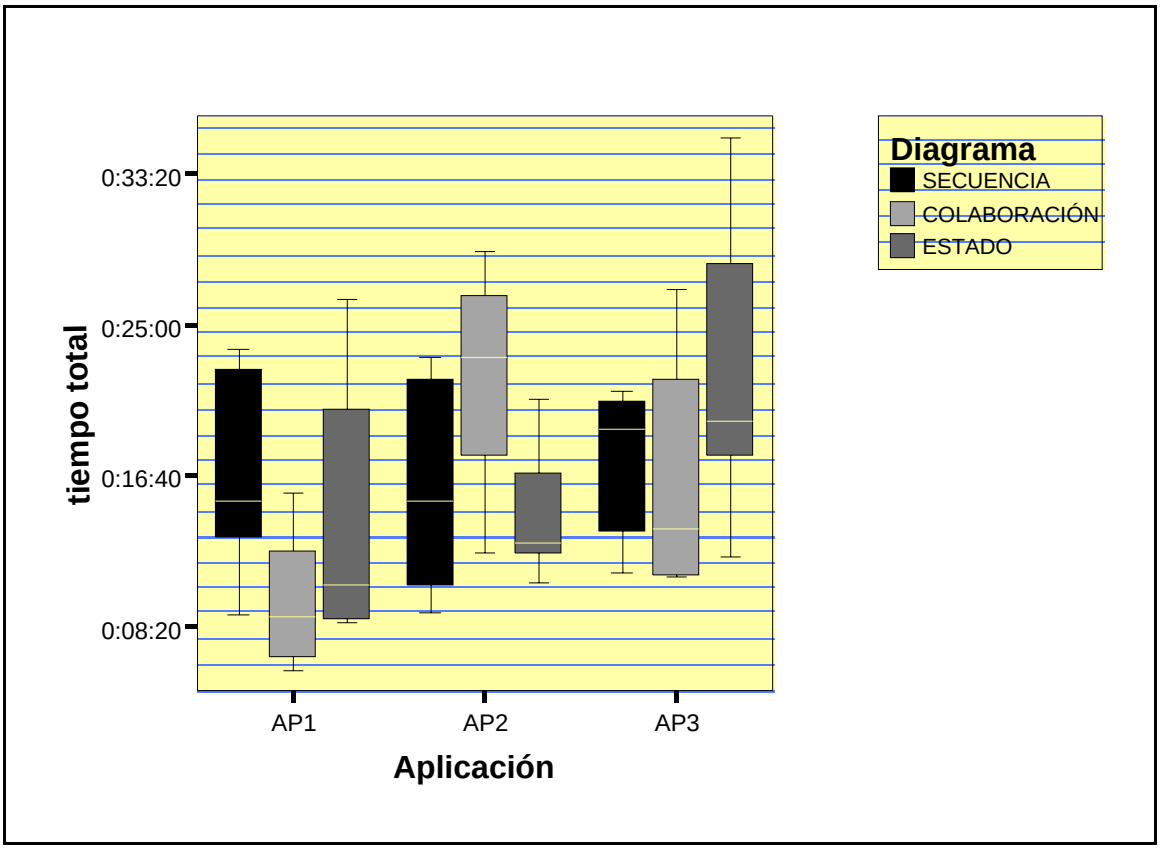


Figura 2.5. Diagrama de caja con respecto al tiempo

Esta herramienta resultó muy útil en el segundo experimento. Mediante su aplicación se pudo comprobar gráficamente que la clasificación de los alumnos en grupos de parecido rendimiento se llevó a cabo de forma correcta. Precisamente en el diagrama de caja y bigotes de la Figura E.1 se puede observar la simetría y distribución homogénea de cada grupo con respecto a la variable puntuación. Los detalles de construcción de un gráfico de caja se encuentran en el apartado E.1, explicados en relación a dicha figura y al experimento en cuestión.

2.3.6. Presentación

La fase de presentación no se refiere sólo a que todo el material experimental debe estar disponible en paquetes de laboratorio para su replicación, sino también a la divulgación de los resultados.

Con respecto a la primera tarea en esta tesis se han creado varias páginas web, cuyas direcciones se proporcionan en el último apartado de cada capítulo donde se describe el experimento.

Para la difusión de los resultados es muy importante la elaboración de artículos que recojan con el mayor detalle posible todo el proceso experimental. A continuación, vamos a citar algunos recursos que abordan de forma específica la experimentación en ingeniería del software:

- La revista *Empirical Software Engineering: An International Journal* publica estudios empíricos desde 1996.
- La conferencia *Empirical Assessment in Software Engineering*, que se celebra en la Universidad de Keele (UK) desde 1997, está orientada a que los investigadores en el área expongan sus experimentos.
- El portal ESERNET (*Experimental Software EngineeRing NETwork*) es una red de experimentación a nivel europeo donde, además de disponer de un tutorial sobre estudios empíricos, se informa de los congresos, las jornadas y conferencias relacionadas con la experimentación en esta disciplina.

Capítulo 3

Situación de la práctica experimental en UML

3.1. Introducción

Debido a que el Lenguaje Unificado de Modelado o UML evolucionó a partir de varios métodos Orientados a Objetos (OO), es un lenguaje que ha acomodado, más que unificado, elementos heterogéneos de las notaciones de sus creadores. Esta "unificación" ha favorecido que UML se convierta en un lenguaje universal y, en consecuencia, que esté abierto a múltiples interpretaciones, lo que origina inconsistencias y ambigüedades semánticas. Precisamente, en [Simo99, capítulo 7] se presenta un catálogo de problemas, que son ratificados por personas con experiencia en la utilización de UML tanto en el ambiente académico como en proyectos reales dentro de la industria del software.

Por otra parte, la amplia aceptación de UML como lenguaje para el análisis y diseño OO, desde su estandarización en 1997 por el consorcio OMG, se ha basado, fundamentalmente, en que ha servido y sirve como vehículo de comunicación entre todas las personas implicadas en el proceso de desarrollo de aplicaciones software. Para contribuir a que su éxito continúe no es suficiente con que todos los usuarios sigan utilizando un lenguaje común, sino que debe hacerse un uso apropiado del mismo tanto en la especificación como en la construcción y documentación de los artefactos OO. Además, dada la complejidad de los sistemas de información actuales, los ingenieros de software demandan métodos más fáciles de usar y modelos que resulten más fáciles de comprender que los sistemas finales. Así pues, el objetivo prioritario de nuestra investigación es lograr que se haga un uso apropiado de UML, ya que repercute en una mayor comprensión del lenguaje y, a su vez, en una mayor facilidad de uso en la tarea de construcción de los modelos. Una forma de verificar este uso apropiado es evaluar empíricamente la comprensión de la información que transmiten los artefactos, especificados en UML, a sus usuarios.

Este capítulo está organizado de la siguiente forma. En el apartado 3.2 exploramos cuál es el "estado de la práctica", desde una perspectiva empírica, de los artefactos OO en ingeniería del software. En el apartado 3.3 detallamos el contexto experimental de la investigación de esta tesis.

3.2. Estado de la práctica empírica de los artefactos construidos con UML

Al igual que sucede en otras disciplinas, la ingeniería del software adopta el paradigma experimental como instrumento para la investigación. No basta sólo con medir los productos software, sino que también es necesario aplicar métodos empíricos (tales como, experimentos controlados, casos de estudio o encuestas) en la evaluación y validación de resultados. Es decir, la experimentación, junto con la medición, proporcionan una base científica importante para la ingeniería del software [Wohl01].

La búsqueda bibliográfica, en un principio, se limitó a los artículos donde se aplicaron métodos empíricos a artefactos OO, obtenidos a lo largo de las etapas del ciclo de vida de un proyecto software. Asimismo en [Deli02] se revisaron una serie de artículos publicados sobre las investigaciones empíricas en la tecnología orientada a objetos. Esta revisión incluye una descripción clara de 18 experimentos controlados así como una crítica a los mismos. Al final, como nuestro interés se centra en la práctica empírica con UML, sólo se seleccionaron aquellas publicaciones que utilizaron artefactos (especificados en notación UML) como material experimental en la etapa de diseño, que son los que examinamos a continuación:

- En [Trav99b] se propusieron una serie de técnicas de lectura: vertical y horizontal, con el fin de que los revisores pudiesen detectar más fácilmente defectos (por ejemplo, inconsistencias tanto sintácticas como semánticas), entre los artefactos UML correspondientes al análisis de requisitos y los artefactos UML de la etapa de diseño. El objetivo de realizar una serie de experimentos controlados fue evaluar la viabilidad y efectividad de dichas técnicas de lectura. Los resultados obtenidos en dichos estudios derivaron en la definición de una plantilla (sencilla y efectiva, descrita en [Trav01]), del proceso de desarrollo de software orientado a objetos, que utiliza UML como lenguaje de modelado y las inspecciones como apoyo a la construcción y el mantenimiento de productos software.
- Otras técnicas de lectura, una basada en listas de comprobación (CBR: *Checklist-Based Reading*) y otra basada en la perspectiva (PBR: *Perspective-Based Reading*) se compararon en [Lait00], con el mismo objetivo que en el estudio empírico anterior. El

resultado del experimento demostró la eficacia de los escenarios PBR para detectar defectos en los documentos de diseño UML.

- Un modelo cognitivo para la integración de diagramas se propuso en [Hahn99]. Los autores plantearon que las características "descomposición" (*decomposition*) y "distribución" (*layout*) de la información diagramática podrían influir en la selección del método a aplicar, tanto en el análisis de los diagramas individuales como en el diseño del proceso de integración de los diagramas. El objetivo del experimento formal que se llevó a cabo fue explorar los efectos de dichas características, referidas a las representaciones diagramáticas en UML, en el proceso cognitivo de manipular e integrar diagramas. Como conclusión se dedujo que la descomposición de la representación diagramática UML era más importante que su distribución. En consecuencia, era un factor a tener en cuenta por los usuarios (por ejemplo, ingenieros de software a la hora de elegir un diagrama para una tarea concreta) y por los diseñadores de estos diagramas (por ejemplo, desarrolladores de metodologías de ingeniería de software a la hora de definir un proceso que integre varios tipos diferentes de diagramas).
- En un estudio posterior realizado por [Purc01] se evaluó el efecto de la estética en la disposición (*layout*) de los elementos gráficos en los diagramas UML. Desde el punto de vista de la usabilidad, se investigan empíricamente las preferencias estéticas en los diagramas de clases y en los diagramas de colaboración, con el fin de reducir el número de cruces e incrementar la visualización de la simetría. Al final se obtuvo una clasificación de los aspectos estéticos a considerar en el diseño futuro de algoritmos de dibujo de grafos UML.
- En [Zend01] se analizaron los conceptos de granulado grueso (*coarse-grained*) de los enfoques orientados a objetos UML, OML (OPEN Modeling Language) y TOS (Taxonomic ObjectSystem). El objetivo fue comparar experimentalmente estos conceptos de modelado desde la perspectiva del desarrollador de software. Los resultados mostraron que cuando se modela una aplicación orientada a bases de datos, los conceptos de granulado grueso de OML y TOS fueron mejores que los de UML.

En resumen, los diagramas UML incluidos como material experimental en dichos experimentos son los que se muestran en la Tabla 3.1.

Referencia	Diagrama de clases	Diagrama de secuencia	Diagrama de colaboración	Diagrama de estado	Diagrama de actividad
[Trav99b]	☑	☑		☑	

[Lait00]	☑	☑		☑	
[Hahn99]		☑	☑		☑
[Purc01]	☑		☑		
[Zend01]	☑				

Tabla 3.1. Artefactos de UML utilizados como material en las referencias citadas

Por otra parte, es importante asociar el objetivo de los estudios empíricos de la Tabla 3.1 con algún aspecto dentro del contexto de la medición en la ingeniería del software. Cualquier producto de desarrollo de software se puede medir en función de tres características: tamaño, funcionalidad y complejidad. Considerando la perspectiva de los objetivos de los experimentos anteriores, éstos están relacionados con la complejidad, pues se puede interpretar de tres formas [Fent97, p. 245]:

- La complejidad *operacional* mide el rendimiento o eficacia del software.

  En realidad, los experimentos de [Trav99b] y [Lait00] no midieron la eficacia de varios de los artefactos UML, sino de una serie de técnicas de lectura de los documentos de diseño donde venían especificados diagramas de UML con el fin de detectar defectos.
- La complejidad *estructural* mide la estructura del software.

  Para medir los diseños orientados a objetos existen un amplio abanico de métricas presentadas en [Brit96], [Chid94], [Lore94] y [Marc98].

  Con respecto a los experimentos de [Hahn99] y [Purc01] no se propusieron nuevas métricas, sino que se evaluaron diferentes características de las estructuras diagramáticas en UML. Es decir, evaluaron la complejidad estructural y definieron un modelo cognitivo, con el fin de mejorar la comprensión y el uso de los modelos UML .
- La complejidad *psicológica o cognitiva* mide la dificultad o el esfuerzo de las personas para entender el software. También se denomina *comprensión o facilidad de comprensión*.

  En [Zend01] se evaluó la facilidad de comprensión de un concepto orientado a objetos en los diagramas de clases. Estos diagramas estaban especificadas en tres lenguajes de modelado. La comparación se realizó en función del dominio de aplicación de los modelos empleados en el experimento.

3.3. Motivación y contexto experimental de esta investigación

Es evidente que los modelos UML son flujos de información bidireccionales. Por una parte, deben ser capaces de captar el significado de una aplicación software como una red de construcciones lógicas (por ejemplo, de clases, asociaciones, estados, casos de uso y mensajes). Por otra parte, deben ser capaces de transmitir toda esa información, referida a la

aplicación, a sus usuarios (por ejemplo, desarrolladores de software, personal de mantenimiento, etc.). Hasta donde he investigado, no he encontrado evidencia empírica sobre cómo es esa capacidad de comunicación de los artefactos UML. Así pues, nuestro objetivo es aplicar métodos empíricos para evaluar la complejidad cognitiva o comprensión de los modelos UML obtenidos en la etapa de diseño.

El modelado de un proyecto software contempla tanto su estructura estática como su comportamiento dinámico. Por ello, el lenguaje UML incluye un conjunto de diagramas estáticos y dinámicos para la documentación del análisis y diseño OO de un proyecto. Estas "representaciones diagramáticas" poseen dos aspectos importantes: información semántica (*semántica*) y presentación visual (*notación*). Asimismo son aspectos dependientes, pues la presentación visual de los diagramas (en notación UML) muestra la información semántica que aportan dichos diagramas, de tal modo que sea comprensible por las personas. En consecuencia, nuestra práctica empírica va a consistir en evaluar la comprensión "semántica" de los artefactos UML.

En el lenguaje UML se utilizan básicamente los diagramas de clases para representar la estructura estática de un sistema software. Mientras que para la especificación del comportamiento dinámico de dicho sistema, incorpora cuatro tipos de diagramas: de *secuencia*, de *colaboración*, de *estado* y de *actividad*. Este número de diagramas dinámicos es el origen de algunas de las preguntas que nos planteamos. En un proyecto software cualquiera, ¿cuál de estos diagramas aporta una información semántica más segura del diseño? ¿Y cuál de ellos en el menor tiempo? ¿Alguno de estos diagramas hace más comprensible semánticamente el diseño a medida que éste es más complejo? Para dar respuesta a estos interrogantes, pensamos en experimentar "in vitro" la comprensión semántica de los diseños UML, cuando el modelado dinámico de los mismos se describe mediante los tres primeros tipos de diagramas enunciados antes. Sin embargo, esta evaluación debe realizarse en base a una serie de factores bien definidos.

En primer lugar, vamos a exponer algunas de las recomendaciones publicadas, acerca de cuál es el diagrama más adecuado, para modelar aspectos del comportamiento dinámico. Posteriormente, en el apartado 6.6.2 se hará referencia a estas recomendaciones, que serán contrastadas con los resultados obtenidos en los dos primeros experimentos.

1. Los diagramas de secuencia, que muestran la ordenación temporal de los mensajes enviados entre los objetos, son mejores para entender las especificaciones en tiempo real y los escenarios complejos ([OMG00, p. 91] y [OMG01, p. 100]).

2. Los diagramas de colaboración, que ayudan a comprender las conexiones (enlaces) entre los objetos, son mejores para entender el diseño de los procedimientos ([OMG00, p. 103] y [OMG01, p. 115]).
3. Los diagramas de estado, que describen las posibles secuencias de estados y acciones a lo largo del tiempo de vida de una instancia de una clase como resultado de su reacción a eventos discretos, se utilizan normalmente en aquellas situaciones en que ocurren eventos asíncronos ([OMG00, p. 146] y [OMG01, p. 157]).
4. En [Rose99, p. 109] los autores citan que los diagramas de colaboración y de estado son más útiles en el diseño de sistemas en tiempo real, o cuando es necesario explicar los aspectos en tiempo real de sistemas cliente/servidor o de otros sistemas distribuidos.
5. En [Fowl00, pp. 111-112] Fowler prefiere utilizar los diagramas de interacción (de secuencia o de colaboración) cuando representan un proceso secuencial simple sin muchos comportamientos condicionales ni repetitivos. Estos diagramas pierden claridad en escenarios con comportamientos más complejos. Por ejemplo, para especificar diferentes comportamientos condicionales es mejor utilizar diagramas separados para cada escenario.

En general, los factores que subyacen en las recomendaciones anteriores son el dominio de aplicación al que pertenecen los proyectos software y la complejidad de los escenarios. Estos criterios que debemos tener en cuenta en nuestra investigación experimental son, en definitiva, características inherentes a los diseños. Por lo tanto, para cada diseño en notación UML hay que indicar su dominio de aplicación, así como la complejidad de sus escenarios.

Por último, tal como se mencionó en el Capítulo 1, existen dos lenguajes estándares de modelado orientado a objetos: UML y OML. Con respecto al segundo lenguaje, apenas se han ejecutado experimentos con artefactos especificados en OML (excepto con artefactos estáticos en [Zend01]). Así pues, esta investigación finaliza con una evaluación empírica desde la perspectiva de la facilidad de comprensión de los diferentes modelos para la especificación del diseño tanto en UML como en OML.

PARTE III

EVALUACIÓN DEL MODELADO DINÁMICO

Capítulo 4

Planificación experimental para la evaluación del modelado dinámico

4.1. Introducción

A la hora de abordar una investigación empírica, es importante que los experimentos no se consideren estudios aislados, sino que formen parte de familias de estudios sobre algún tema de interés científico, en nuestro caso, sobre ingeniería del software. El objetivo final es crear un cuerpo de conocimiento [Basi99], tal como existe en otras disciplinas como la medicina o la física. Para ello es necesario disponer de una estructura formada por un conjunto de principios que unifique tanto las familias de experimentos como sus resultados. Mediante esta integración es posible extraer conocimiento útil a partir de los experimentos y extrapolar ese conocimiento.

Con el fin de que los experimentos de esta investigación formen parte de dicho cuerpo de conocimiento, este capítulo se organiza de la siguiente forma. En el apartado 4.2 se formula el objetivo general. En el apartado 4.3 se explica todo el proceso de planificación experimental que se deriva de este objetivo y que integra los experimentos realizados para evaluar el modelado dinámico. En los apartados 4.4 y 4.5 se discuten los principios y los diferentes tipos de diseño experimental utilizados en los estudios empíricos de esta investigación.

4.2. Objetivo general de la investigación

La primera idea que subyace bajo esta investigación tuvo su origen en el estudio del lenguaje de modelado unificado UML. En este lenguaje, para representar la estructura estática de un sistema software, se utilizan básicamente los diagramas de clases. Mientras que para la especificación del comportamiento dinámico de dicho sistema, este lenguaje incluye cuatro tipos de diagramas: de secuencia, de colaboración, de estado y de actividad. ¿Para qué

utilizamos tantos diagramas para diseñar una aplicación software? La respuesta es muy simple, para que nos sirva como vehículo de comunicación.

La comunicabilidad que transmite una forma diagramática, que en nuestro caso representa a un sistema software, recae en una serie de factores que hay que considerar: rapidez en la lectura, facilidad de comprensión y seguridad en la interpretación. Estas cuestiones tienen que estudiarse para mejorar la capacidad de comunicación de los diagramas. Precisamente la existencia de cuatro tipos diferentes de diagramas dinámicos es el motivo de algunas de las preguntas que nos planteamos. En una aplicación software cualquiera, ¿cuál de estos diagramas aporta una información más segura del diseño? ¿Y cuál de ellos en el menor tiempo? ¿Alguno de estos diagramas hace más comprensible el diseño a medida que éste es más complejo? Para dar respuesta a estos interrogantes pensamos en efectuar experimentos donde uno de los factores sea el tipo de modelo dinámico.

Para que esta primea idea adquiera consistencia es preciso plantear formalmente el objetivo a estudiar. En este sentido la plantilla GQM, definida en el apartado 2.3.1.1, constituye una herramienta de ayuda en la definición del objetivo, tanto a nivel general como después a nivel específico, de una investigación experimental. Aplicando dicha plantilla, el objetivo general de este trabajo es el siguiente:

Objetivo	Analizar el modelado dinámico para evaluar su comprensión desde el punto de vista del desarrollador o diseñador de aplicaciones software en el contexto de un conjunto de variables de entorno.
Pregunta	¿Cómo se define la comprensión de un diagrama?
Métrica	• Tiempo • Puntuación

Tabla 4.1. Formulación del objetivo general de este trabajo de investigación

Los pasos que comprende todo proceso de comunicación son primero la lectura, luego la comprensión y, por último, una interpretación correcta de la forma diagramática. Teniendo en cuenta estos pasos, la comprensión se define en base a estas dos métricas:

- La métrica del *tiempo* mide los minutos y segundos invertidos en las tareas de leer y comprender un diagrama dinámico.

- La métrica de la *puntuación* proporciona el número de respuestas correctas tras la interpretación del diagrama dinámico.

4.3. Planificación experimental

El objetivo general de un trabajo empírico también se denomina objetivo de alto nivel [Basi99]. El siguiente paso es refinar este objetivo en objetivos más específicos. Este refinamiento se consigue mediante la especificación concreta del <*objeto de estudio*> del objetivo general. Al principio, el <*modelado dinámico*> se concretó sólo para los diagramas dinámicos de UML considerándolos individualmente. Esta primera especificación resultó en la formulación del objetivo del primer experimento para evaluar el modelado dinámico. Los siguientes objetivos específicos derivaron en una serie de estudios empíricos que, junto con el primero, conforman el proceso de desarrollo experimental de esta investigación. Todos estos objetivos específicos se encuentran definidos en el apartado 5.2.1 para el primer estudio y su réplica, y en los apartados 7.2.1 y 8.2.1 para el tercer y cuarto experimento respectivamente.

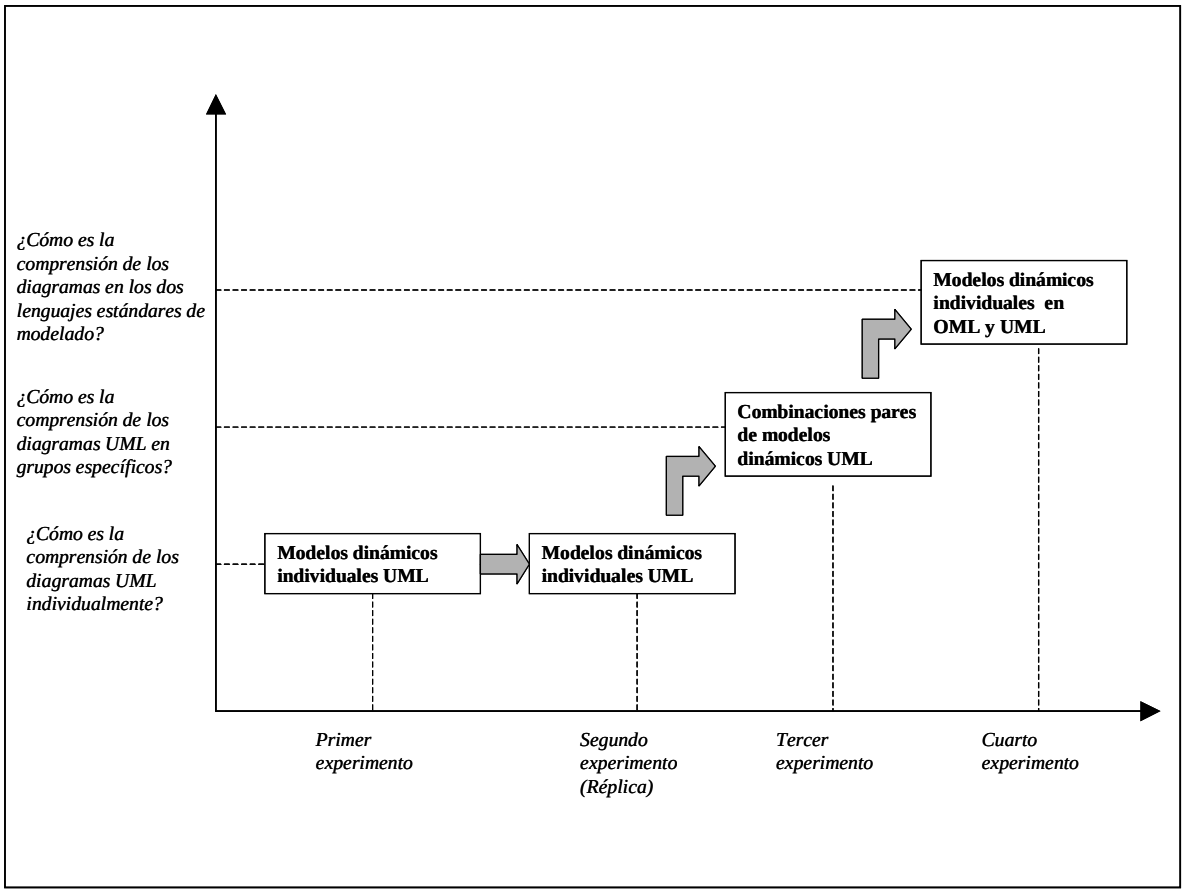


Figura 4.1. Proceso de planificación experimental para la evaluación del modelado dinámico

La Figura 4.1 muestra la evolución de dicho proceso de planificación, donde el recuadro que contiene el objetivo específico corresponde a la pregunta cada vez más compleja del eje Y y al experimento en el eje X que se realizó para obtener la respuesta. Esta planificación experimental para evaluar el modelado dinámico no se concibió así desde un primer momento, sino que se fue desarrollando a medida que se iban ejecutando los experimentos. Así pues, el proceso de desarrollo de esta planificación se trató como un enfoque evolutivo. Por ejemplo, los resultados del primer estudio empírico condicionaron el planteamiento del segundo experimento, que fue una réplica del primero.

Asimismo, en la planificación de cada experimento se tuvieron en cuenta los principios y las técnicas de control, explicadas en el capítulo 2, con el fin de obtener el tipo de diseño más adecuado. Los principios y los tipos de diseño que se aplicaron en cada experimento se detallan en los siguientes apartados 4.4 y 4.5 respectivamente.

4.4. Principios de diseño experimental

Dentro de la planificación de esta investigación se ha tratado de aplicar a los experimentos todos los principios de diseño experimental comentados en el apartado 2.3.2.5.1.

Las técnicas de control para reducir el error experimental se aplicaron en los cuatro estudios empíricos: asignación aleatoria de los sujetos a los tratamientos en el primer y cuarto experimento y bloqueo para conseguir grupos emparejados de sujetos en el segundo y tercer experimento. Por otra parte, todos los diseños fueron equilibrados.

Un principio de diseño importante relacionado con la validez de los resultados es la replicación. Concretamente el segundo experimento es una réplica interna del primero, con el fin de comprobar el grado de fiabilidad de las conclusiones obtenidas en el primer estudio empírico.

4.5. Tipos de diseño experimental empleados en esta investigación

Teniendo en cuenta cualquier clasificación general de diseño experimental ([Wine91] y [Dean99]), el cuarto experimento se engloba, en un primer nivel dentro de los denominados diseños simples, pues sólo hay una fuente de variación o factor objeto de estudio. El primer, segundo y tercer experimento, al considerar dos factores o variables independientes, sus diseños corresponden a los llamados diseños complejos o factoriales. Pormenorizando la designación de los diseños de estos experimentos, los tipos de diseños se denominan:

- Diseño factorial confundido de cuadrado latino: Primer experimento
- Diseño factorial de parcelas divididas: Segundo (réplica) y tercer experimento
- Diseño *cross-over* de 2 tratamientos: Cuarto experimento

A continuación, vamos a explicar los conceptos, el diagrama y el modelo estadístico correspondiente a cada experimento según la nomenclatura de [Kirk95]: diseño *cross-over* de 2 tratamientos (apartado 4.5.1.1), diseño factorial confundido de cuadrado latino (apartado 4.5.2.1) y diseño factorial de parcelas divididas (apartado 4.5.2.2). El orden elegido es un orden lógico en cuanto que empezamos por el de menor dificultad o complejidad, continuamos con el del primer experimento y finalizamos con el diseño factorial de parcelas divididas, pues su diseño se justifica en base a los resultados obtenidos a partir del diseño del primer experimento.

4.5.1. Diseño cross-over

En un diseño cross-over los sujetos reciben primero un nivel o tratamiento del factor a investigar y luego un segundo tratamiento, que es el inverso al primero, y quizás un tercer o incluso un cuarto nivel. Como los sujetos se miden en los diferentes niveles de la variable independiente, un diseño cross-over es un diseño intra-sujetos o de medidas repetidas.

Los diseños de medidas repetidas minimizan las amenazas a la validez interna debidas a las diferencias individuales (selección). Como contrapartida, la recurrencia de los mismos sujetos en todos los tratamientos ocasiona la aparición de otros peligros. Es decir, los efectos distorsionantes en los diseños intra-sujetos causados por la maduración, la instrumentación, la mortalidad y la práctica tienen que ser controlados. Una forma de amortiguar el efecto del aprendizaje (maduración) consiste en que todos los sujetos participen en todos los tratamientos de forma rotatoria y en días diferentes, para contrarrestar el cansancio. Sin embargo, pese a los esfuerzos por controlarlos, siempre está presente el efecto de la práctica, que se puede medir aplicando un plan en cuanto al orden de presentación de los niveles de la variable independiente. Cuando los niveles son dos, el diseño se denomina reequilibrado (*counter balancing*); pero si son tres o más, el diseño es un cuadrado latino (*latin square*).

Vamos a abordar dos casos de diseños Cross-Over o CO-p, donde p indica el número de niveles del factor a estudiar: diseño CO-2 o reequilibrado (apartado 4.5.1.1) y diseño CO-3 o cuadrado latino (apartado 4.5.1.2).

4.5.1.1. Diseño cross-over de 2 tratamientos: CO-2

El diseño cross-over más sencillo es el que corresponde al de un factor con 2 niveles o tratamientos, abreviadamente diseño CO-2 [Kirk95, pp. 349-350]. También recibe la denominación de diseño reequilibrado. La técnica del reequilibrado consiste en repetir los dos tratamientos experimentales, a_1 y a_2, de tal forma que primero se presentan en un orden y después en el inverso. Es decir, la mitad de los sujetos reciben el tratamiento a_1 seguido de a_2, y la otra mitad reciben a_2 seguido de a_1, tal como muestra la Tabla 4.2. Si hay un efecto del primer tratamiento sobre el segundo, produciendo un desequilibrio, entonces se anula al invertirse dicho efecto en la segunda repetición, quedando reequilibrado. Aún así, una de las amenazas potenciales a la validez interna de un diseño cross-over es el efecto de persistencia, que se trata de un efecto residual al perdurar todavía en el segundo tratamiento el efecto del primero. Para eliminar, o por lo menos minimizar, esta amenaza, en este tipo de diseño se planifica un período de descanso (*washout period*) entre la administración del primer tratamiento y la del siguiente.

4.5.1.1.1. Diagrama de CO-2

	Período b_1	**Período** b_2
	Nivel o Tratamiento	**Nivel o Tratamiento**
$Bloque_1$	a_1	a_2
$Bloque_2$	a_2	a_1

Tabla 4.2. Diseño cross-over de 2 tratamientos o CO-2

Los diseños cross-over, denotados por CO-p, son muy utilizados en la investigación farmacológica, agrícola y en las ciencias sociales. Estos diseños son apropiados para experimentos que cumplan las siguientes condiciones [Kirk95, pp. 350-351]:

1. Hay un factor con $p \geq 2$ niveles o tratamientos y 2 variables extrañas. Una de las variables extrañas son las unidades experimentales (bloques); la otra son los períodos de tiempo y debe tener p niveles.
2. Cada tratamiento tiene que ocurrir un número igual de veces en cada período de tiempo. Así pues, el diseño requiere $n = kp$ bloques, donde k es un entero positivo.
3. Las unidades experimentales se asignan aleatoriamente a los n bloques. Cada unidad experimental se observa p veces.

4. Los efectos de un tratamiento deben disiparse antes de que el siguiente tratamiento sea distribuido.
5. Es razonable asumir que no hay interacciones entre los bloques, los períodos de tiempo y los niveles del factor objeto de estudio. Si este supuesto no se cumple, entonces el contraste de alguno de los efectos que corresponda está sesgado.

Este diseño cruzado simple (*cross-over* con 2 tratamientos o *cross-over* 2x2 con 2 filas y 2 columnas) se puede considerar un diseño que tiene en cuenta 1 factor y 2 fuentes extrañas de variación (las filas y las columnas) o bien un diseño factorial que permite estimar los efectos principales de 3 factores (las filas, las columnas y el factor con los 2 tratamientos), tal como se puede ver en [Kirk95] y [Kueh00]. A la hora de aplicar este tipo de diseño al cuarto experimento, se planteó como un diseño factorial de 3 factores. Pero se trata de un factorial incompleto, pues no se van a estimar todas las interacciones posibles entre los 3 factores, al no ser estas interacciones independientes de los factores principales.

La aplicación de este diseño al cuarto experimento se puede comprobar en la Figura 8.1. El factor filas o bloques de la Tabla 4.2 se corresponde con el factor grupo o secuencia de la tabla de la izquierda de la Figura 8.1. Asimismo el factor columnas o períodos de la Tabla 4.2 equivale al factor período en la Figura 8.1, que mide el efecto de aprendizaje de una sesión a otra (del Período I al II). Los tratamientos a_1 y a_2 de la Tabla 4.2 representan al lenguaje UML y OML respectivamente en la Figura 8.1.

4.5.1.1.2. Modelo Lineal Estadístico de CO-2

Para efectuar un análisis estadístico correcto de los datos, se partió en un principio del modelo lineal completo de un cross-over 2x2 [Kueh00, pp. 536-538], que incluye el efecto de carryover. Sin embargo, el análisis de este modelo adolece de una serie de problemas. Estos problemas se explican en el apartado 8.5.1 dentro del contexto del cuarto experimento. Para resolverlos, se adopta como solución alternativa el modelo lineal de un CO-p [Kirk95, p. 351].

$$Y_{ijk} = \mu + \alpha_j + \beta_k + \pi_i + \varepsilon_{ijk}$$

(i = 1, …, n; j = 1, …, p; k = 1, …, p)

donde

Y_{ijk} Es la respuesta para un sujeto en el bloque *i*, en el nivel de tratamiento *j*, y en el período *k*.

μ Es la media general.

α_j Es el efecto del factor objeto de estudio para la población *j*, que tiene *p* niveles.

β_k Es el efecto del factor columna (período) para la población *k*.

π_i Es el efecto del factor fila (bloque) para la población *i*.

ε_{ijk} Es el efecto del error aleatorio.

4.5.1.2. Diseño cross-over de 3 tratamientos: CO-3

El diseño cross-over correspondiente al de un factor con 3 niveles se denomina diseño CO-3 [Kirk95, pp. 349-350]. También recibe la denominación de diseño de cuadrado latino 3x3 [Wine91, pp. 674-679], pues tiene 3 filas y 3 columnas.

La lógica implícita en la técnica de control mediante reequilibrado en el caso de dos niveles en la variable independiente, es la de compensar los efectos de la práctica mediante la aplicación de dos tratamientos en los dos órdenes posibles: a_1a_2 y a_2a_1. Seguir esta lógica en un diseño donde la variable independiente tuviese tres niveles implicaría aplicar dichos niveles en seis secuencias diferentes: $a_1a_2a_3$, $a_1a_3a_2$, $a_2a_1a_3$, $a_2a_3a_1$, $a_3a_1a_2$ y $a_3a_2a_1$. Como requiere un gran esfuerzo se han desarrollado técnicas para controlar el posible efecto de la práctica cuando son más de dos los tratamientos presentados. Se denominan diseños de cuadrado latino [Wine91]. En este tipo de diseños el control del efecto de la práctica se lleva a cabo mediante la selección de un grupo de secuencias de presentación de entre todas las posibles. Se eligen tantas secuencias como tratamientos haya en el experimento, de tal forma que cada tratamiento sólo ocurra una vez en cada posición dentro del conjunto de secuencias elegidas (bloque) y dentro de cada período. En la Tabla 4.3 se presentan los tres órdenes de presentación para la construcción de un cross-over de tres niveles, que debe cumplir las 5 condiciones expuestas en el apartado 4.5.1.1.

4.5.1.2.1. Diagrama de CO-3

	Período b_1	**Período** b_2	**Período** b_3
	Nivel o Tratamiento	**Nivel o Tratamiento**	**Nivel o Tratamiento**
Bloque$_1$	a_1	a_2	a_3
Bloque$_2$	a_2	a_3	a_1
Bloque$_3$	a_3	a_1	a_2

Tabla 4.3. Diseño cross-over de 3 tratamientos o CO-3

4.5.2. Diseño factorial confundido

Un diseño factorial se caracteriza por estudiar dos o más variables independientes, denominadas factores, que a su vez se dividen en niveles o tratamientos. Entonces si el número de factores es 2 y cada uno tiene 3 niveles, habrá 3x3=9 combinaciones diferentes de tratamientos o condiciones experimentales.

El primer factor que condiciona la elección de un tipo u otro de diseño factorial viene determinado por la abundancia o escasez de recursos humanos. Cuando hay un número alto de sujetos experimentales, por ejemplo, 90, se puede recurrir a un diseño factorial completamente aleatorio, donde cada sujeto se asigna al azar a una combinación de tratamiento y únicamente se mide una vez en dicho tratamiento. Entonces, dividiendo 90 entre 9 se obtienen 10 medidas en cada una de las condiciones experimentales. Por el contrario, si no es posible disponer de un gran número de sujetos, tal como ya se manifestó en el apartado 2.3.2.3, esta carencia de recursos repercute en la utilización de un diseño factorial de medidas repetidas.

Por otra parte, en ingeniería del software, medir a todos los sujetos en todas las combinaciones tampoco es la solución, ya que la propia naturaleza de los tratamientos condiciona el número de veces que un sujeto puede participar en un experimento. Ante esta dificultad de conseguir un factorial completo, se recurre a un diseño factorial confundido. La técnica de la confusión tiene como objetivo lograr una reducción en el número de combinaciones de tratamiento que se deben asignar a los bloques. En un diseño de medidas repetidas un bloque contiene un sujeto, que se mide en las combinaciones del bloque, mientras que en un diseño de medidas no repetidas un bloque contiene sujetos emparejados, donde cada uno se mide únicamente en una de las condiciones experimentales del bloque.

Precisamente el tipo de confusión en un diseño es otro de los factores utilizados para clasificar un diseño factorial. Los tipos o esquemas de confusión utilizados en los tres primeros experimentos de esta investigación se llaman, según el texto de [Kirk95]:

- Confusión grupo-interacción
- Confusión grupo-tratamiento.

La confusión grupo-interacción es característico de un diseño factorial confundido de grupos aleatorios y de un diseño factorial confundido de cuadrado latino, al igual que la confusión grupo-tratamiento lo es de un diseño factorial de parcelas divididas. Sea un diseño de medidas repetidas o no repetidas un grupo contiene un conjunto de bloques o personas

[Kirk95], mientras que en otros textos el bloque es sinónimo del concepto de grupo anterior. Para evitar confusiones en este trabajo vamos a adoptar el concepto de grupo de [Kirk95].

En los dos apartados siguientes vamos a describir el diseño factorial confundido de cuadrado latino (apartado 4.5.2.1), que corresponde al diseño del primer experimento, y el diseño factorial de parcelas divididas (apartado 4.5.2.2.), utilizado en el segundo y en el tercer experimento.

4.5.2.1. Diseño factorial confundido de cuadrado latino

Este diseño se denomina factorial confundido de cuadrado latino (LSCF: *Latin Square Confounded Factorial*), pues como esquema de confusión utiliza un cuadrado latino [Kirk95, capítulo 13]. El diagrama de la Tabla 4.4. muestra el diseño factorial de un cuadrado latino o LSCF-3^2 con bloques de tamaño 3 [Kirk95, pp. 650-652]. Un diseño LSCF-3^2 indica que tiene 2 factores tratamiento, A y B, cada uno con 3 niveles. Este diseño también se denomina diseño de cuadrado latino 3x3 de medidas repetidas [Wine91, pp.702-704], diseño factorial 3^2 en 3 bloques de tamaño 3 [Dean99, pp. 462-464] o diseño de bloques incompletos para un factorial 3^2 [Kueh00, pp. 383-385].

4.5.2.1.1. Diagrama de LSCF-3^2

		b_0 **Combinación de tratamiento** a_jb_k	b_1 **Combinación de tratamiento** a_jb_k	b_2 **Combinación de tratamiento** a_jb_k
$(AB^2)_0$ Grupo$_0$		a_0	a_1	a_2
	Bloque$_0$	00	11	22
	...	...	...	...
	Bloque$_{n-1}$	00	11	22
$(AB^2)_1$ Grupo$_1$		a_1	a_2	a_0
	Bloque$_n$	10	21	02
	...	...	...	...
	Bloque$_{2n-1}$	10	21	02
$(AB^2)_2$ Grupo$_2$		a_2	a_0	a_1
	Bloque$_{2n}$	20	01	12
	...	...	...	...
	Bloque$_{3n-1}$	20	01	12

Tabla 4.4. Diseño factorial confundido de cuadrado latino o LSCF-3^2

La interacción A x B en un diseño LSCF-3^2 tiene (3 – 1)(3 – 1) = 4 grados de libertad y se puede dividir en dos componentes ortogonales: (AB) y (AB^2), cada uno de ellos con (3 –1) = 2 grados de libertad. La utilidad de estas componentes es que un diseño LSCF-3^2 se construye

confundiendo la componente (AB) o (AB^2) de la interacción con los grupos de bloques (personas). Si nos fijamos en la Tabla 4.4 los grupos $Grupo_0$, $Grupo_1$ y $Grupo_2$ coinciden con la componente (AB^2).

Un diseño factorial confundido de cuadrado latino es adecuado para experimentos que cumplan las siguientes condiciones [Kirk95, pp. 588-589]:

1. Hay dos o más factores y cada uno de ellos tiene $p \geq 2$ niveles o tratamientos.

2. El número de combinaciones de tratamientos es mayor que el tamaño deseado de cada bloque.

3. La variación entre grupos está confundida con una o más interacciones. Los efectos confundidos normalmente se evalúan con menos potencia que los efectos no confundidos.

4. Si se obtienen medidas repetidas en las unidades o sujetos experimentales, cada bloque contiene un sujeto que es observado v veces, donde v es el número de combinaciones de tratamiento en cada bloque. Si no se obtienen medidas repetidas, cada bloque contiene v sujetos homogéneos o emparejados.

5. En el caso de medidas repetidas, nw bloques (sujetos) se asignan al azar a los w grupos, con n en cada grupo. El orden de administración de las v combinaciones de tratamiento dentro de cada bloque es independientemente aleatorio para cada bloque.

6. En el caso de medidas no repetidas, nw bloques, donde cada uno contiene v sujetos emparejados, se asignan al azar a los w grupos. Asimismo, los v sujetos emparejados dentro de un bloque se asignan aleatoriamente a las v combinaciones de tratamiento.

En el diseño de la Tabla 4.4 el patrón de confusión coincide con la componente (AB^2) de la interacción, que es el esquema que se utilizó en el diseño del primer experimento. Las combinaciones de los niveles de los factores A y B, que constituyen la componente (AB^2) de la interacción A x B, satisfacen la siguiente relación:

$$\left.\begin{array}{l} a_j + 2b_k = 0 (mod\ 3) \\ a_j + 2b_k = 1 (mod\ 3) \\ a_j + 2b_k = 2 (mod\ 3) \end{array}\right\} \Rightarrow \begin{array}{cccll} 00 & 11 & 22 & (AB^2)_0 & \Rightarrow Grupo_0 \\ 10 & 21 & 02 & (AB^2)_1 & \Rightarrow Grupo_1 \\ 20 & 01 & 12 & (AB^2)_2 & \Rightarrow Grupo_2 \end{array}$$

Así pues, en un factorial 3x3 mediante la confusión de la componente (AB) o (AB^2) con el grupo se consigue reducir el tamaño del bloque de nueve a tres. Cada sujeto se mide en las tres combinaciones de tratamientos correspondientes al bloque. En el apartado 5.3.5 sobre el diseño del primer experimento se detallan qué sujetos se asignaron a los bloques, qué condiciones experimentales correspondieron a cada bloque y qué bloques se asignaron a los grupos según la ecuación de la componente (AB^2) de la interacción.

4.5.2.1.2. Modelo Lineal Estadístico de LSCF-3²

El modelo lineal que corresponde al diseño factorial confundido de cuadrado latino o LSCF-3^2 descrito en la Tabla 4.4 es el siguiente:

$$Y_{ijkm} = \mu + \gamma_m + \pi_{i(m)} + \alpha_j + \beta_k + (\alpha\beta)'_{jk} + \varepsilon_{ijkm}$$

(i = 1, …, n; j = 1, …, 3; k = 1, …, 3 ; m = 1, …, p)

donde

Y_{ijkm} Es la respuesta en el sujeto o bloque *i*, en la combinación de tratamiento a_jb_k y en el grupo *m*.

μ Es la media general.

γ_m Es el efecto del grupo, confundido con la componente (AB^2), para la población *m*.

$\pi_{i(m)}$ Es el efecto del sujeto o bloque *i* dentro del grupo *m*.

α_j Es el efecto principal del factor A para la población *j*, que tiene 3 niveles

β_k Es el efecto principal del factor B para la población *k*, que tiene 3 niveles.

$(\alpha\beta)'_{jk}$ Es el efecto de los niveles de los tratamientos *j* y *k* que no están confundidos con los grupos, es decir, el efecto de la componente (AB) no confundida.

ε_{ijkm} Es el efecto del error aleatorio.

Este tipo de diseño LSCF-3^2, clasificado como diseño de confusión grupo-interacción, fue el elegido para el primer experimento. La primera razón es que si los factores A y B influían en las variables dependientes, estos factores estarían libres de confusión. Por otra parte, si la interacción A x B resultaba significativa, ésta sólo estaría parcialmente confundida con los grupos de bloques. El análisis estadístico del primer experimento reveló que la interacción fue significativa. Este resultado implicó que nuestro interés se centrara más en el efecto de la interacción A x B que en los efectos del factores A o B. Así pues, en la planificación del segundo experimento, réplica del primero, el objetivo era que la interacción estuviese libre de confusión. Teniendo en cuenta la recomendación que figura en [Kirk95, p. 652] consideramos un diseño de parcelas divididas, que corresponde al tipo de confusión grupo-tratamiento, donde los grupos de personas están confundidos con el factor A.

4.5.2.2. Diseño factorial de parcelas divididas

El diseño factorial de parcelas divididas (SPF: *Split-Plot Factorial*) tuvo su origen en la investigación en el campo de la agricultura, tal como ya se mencionó en el apartado 2.3.2.5.

Su utilización se extendió a la investigación en educación, donde [Lind53] lo denominó diseño mixto por la mezcla de efectos entre-sujetos e intra-sujetos. Si nos fijamos en el factor A de la Tabla 4.5 la diferencia entre los niveles a_0, a_1 y a_2 implica la diferencia entre grupos de bloques. Por ello los efectos del factor A también se llaman efectos entre-grupos o efectos entre-bloques. En cambio, las diferencias entre los niveles del factor B (b_0, b_1 y b_2) no son diferencias entre grupos de sujetos (bloques), pues cada bloque aparece con todos los niveles de B. Por ello los efectos del factor B también se denominan efectos intra-grupos o efectos intra-bloques.

Este tipo de diseño se puede consultar en [Wine91, pp. 497-580], [Dean99, pp. 675-691] y [Kueh00, pp. 492-513]. Pero, por su claridad, vamos a utilizar la notación de [Kirk95]. Así pues, la notación de un diseño factorial de parcelas divididas de 2 factores es SPF-p.q [Kirk95, pp. 512-580], donde p representa los niveles del factor entre-bloques (factor A) y q los niveles del factor intra-bloques (factor B).

4.5.2.2.1. Diagrama de SPF-3.3

		b_0 **Combinación de tratamiento** a_jb_0	b_1 **Combinación de tratamiento** a_jb_1	b_2 **Combinación de tratamiento** a_jb_2
a_0 $Grupo_0$	$Bloque_0$	00	01	02
	...	...	...	...
	$Bloque_{n-1}$	00	01	02
a_1 $Grupo_1$	$Bloque_n$	10	11	12
	...	...	...	...
	$Bloque_{2n-1}$	10	11	12
a_2 $Grupo_2$	$Bloque_{2n}$	20	21	22
	...	...	...	...
	$Bloque_{3n-1}$	20	21	22

Tabla 4.5. Diseño factorial de parcelas divididas o SPF-3.3

El diseño de la Tabla 4.5 corresponde a un diseño SPF-3.3, factorial que tiene 2 factores y cada uno con 3 niveles. De igual manera, la notación para un diseño con dos factores entre-sujetos (factor A con p niveles y factor C con r niveles) y un factor intra-sujetos (factor B con q niveles) es SPF-pr.q, que fue el diseño utilizado en la réplica.

Un diseño de parcelas divididas es apropiado para experimentos que cumplan las siguientes condiciones [Kirk95, pp. 514-515]:

1. Hay dos o más factores y cada uno de ellos tiene 2 o más niveles o tratamientos.
2. El número de combinaciones de tratamientos es mayor que el tamaño de cada bloque.
3. Si se obtienen medidas repetidas de las unidades o sujetos experimentales en todos los niveles del factor B, cada bloque contiene un sujeto que es observado q veces. Si no se obtienen medidas repetidas, cada bloque contiene q sujetos homogéneos o emparejados.
4. En el caso de medidas repetidas, n bloques se asignan al azar a cada nivel del factor A. El orden de presentación de los q niveles del factor B es aleatorio para cada bloque.
5. En el caso de medidas no repetidas, n bloques, donde cada bloque contiene q sujetos homogéneos, se asignan al azar a cada nivel del factor A. Luego, los q sujetos dentro de cada bloque se asignan aleatoriamente a los niveles del factor B.

En el diseño de la Tabla 4.5 se puede comprobar que los efectos de los niveles del factor A no se pueden separar de los efectos de los grupos de bloques. Por ello se dice que el factor A y los grupos de sujetos están completamente confundidos y que el factor B y la interacción A x B están libres de confusión. Pero, ¿cómo afecta la confusión a los efectos de los factores principales, A y B, y de su interacción A x B? En el caso del factorial confundido LSCF-3^2 los factores A y B se evalúan con igual potencia y el contraste de hipótesis de la interacción confundida es de menor potencia. Por el contrario, en el caso del factorial confundido SPF-3.3 la potencia de los contrastes estadísticos correspondientes al factor B y a la interacción A x B es mayor que la potencia del efecto del factor A confundido. Así pues, un diseño factorial de parcelas divididas es una buena elección cuando el interés del investigador implica al factor B o a la interacción A x B.

4.5.2.2.2. Modelo Lineal Estadístico de SPF-3.3

El modelo lineal que corresponde al diseño factorial de parcelas divididas o SPF-3.3 descrito en la Tabla 4.5 es el siguiente:

$$Y_{ijk} = \mu + \alpha_j + \pi_{i(j)} + \beta_k + (\alpha\beta)_{jk} + \varepsilon_{ijk}$$

(i = 1, …, n; j = 1, …, 3; k = 1, …,3)

donde

Y_{ijk} Es la respuesta en el sujeto o bloque *i* y en la combinación de tratamiento $a_j b_k$.

μ Es la media general.

α_j Es el efecto principal del factor A para la población *j*, que tiene 3 niveles

$\pi_{i(j)}$ Es el efecto del sujeto o bloque *i* dentro del factor A.

β_k Es el efecto principal del factor B para la población *k*, que tiene 3 niveles.

$(\alpha\beta)_{jk}$ Es el efecto de los niveles de las combinaciones de los niveles *j* y *k*, es decir, el efecto de la interacción A x B.

ε_{ijk} Es el efecto del error aleatorio.

Teniendo en cuenta la relevancia estadística de la interacción de los dos factores del primer estudio empírico, la réplica no se ejecutó utilizando el mismo diseño del experimento original, sino utilizando un diseño que sirviese a nuestro propósito, tal como ya se indicó en el apartado 4.5.2.1. En un principio, se pensó en un SPF-3.3 de medidas repetidas, donde el proceso para conseguir medidas repetidas fuese el explicado en el paso 4 anterior. Este procedimiento consiste en que el orden de presentación de los niveles del factor B no confundido sea aleatorio para cada bloque (sujeto). Si la presentación de un nivel del factor B afecta al rendimiento de un sujeto en los siguientes niveles, entonces se dice que el experimento presenta efectos de persistencia (*carry-over effects*). Una parte de los efectos de persistencia se atribuyen a que los niveles del factor B se presentan en un orden determinado, lo que se denomina efectos de secuencia, de la práctica o del orden de presentación. Otra parte de los efectos de persistencia se debe a que las medidas repetidas en cada sujeto se toman únicamente en un nivel del factor A, lo que se denomina efectos de aprendizaje.

Tal como ya se comentó en el apartado 2.3.2.6 hay dos estrategias de control del efecto de la práctica. La aleatorización del orden de presentación de los niveles del factor B para cada sujeto es una forma de controlar los efectos de secuencia. Una segunda estrategia de control de los efectos de la práctica es incluir la secuencia como otro factor más en el diseño. Esta última alternativa es la que se utilizó en el diseño del segundo experimento (réplica).

Concretamente, en el caso que nos ocupa un diseño SPF-3.3 tiene q! = 3! = 6 secuencias diferentes en las que se pueden presentar los niveles del factor B. Por lo tanto, en el diseño se incluye un tercer factor C, cuyos niveles equivalen a los seis órdenes de presentación de B. Al ser el factor C un factor entre-sujetos con r = 6 niveles, entonces el diseño del segundo experimento se denota como SPF-36.3. En el apartado 6.3.5 se explica cómo se aplicó este tipo de diseño a los factores objeto de estudio en la réplica. Las ventajas de incluir los efectos de secuencia como un factor más en el diseño son que controlamos dichos efectos y que

podemos medir no sólo si los efectos de secuencia afectan a las variables dependientes, sino también la influencia de los efectos de aprendizaje (madurez) y los efectos de persistencia sobre las mismas [Kueh00]. Precisamente, en el apartado 6.5.2 se detalla cómo se codificaron estos dos últimos tipos de efectos antes de realizar el primer análisis estadístico de los datos. Por otra parte, en el apartado 6.5.1 se muestra el modelo lineal del diseño SPF-36.6 aplicado a los conceptos y factores de la réplica.

Por último, el tercer experimento, al ser una prolongación del segundo experimento, estaba condicionado por el diseño y los resultados de éste último. En la réplica la interacción A x B resultó significativa, lo que implicó que el diseño del tercer estudio empírico fuese también un factorial de parcelas divididas. En concreto, se trata de un diseño SPF-3.2, que tiene un factor entre-sujetos (factor A) de 3 niveles y un factor intra-sujetos (factor B) de 2 niveles. No se planificó un diseño factorial de parcelas divididas más complejo, puesto que los resultados estadísticos del segundo experimento revelaron que los efectos de madurez, de la práctica y de persistencia no ejercieron influencia alguna sobre las variables respuesta. En el apartado 7.3.5 se explica qué bloques (sujetos) se asignaron a los grupos, o lo que es igual a los niveles del factor A, y cuál fue el orden de presentación de los niveles del factor B para cada bloque.

Capítulo 5

Evaluación inicial del modelado dinámico en UML

5.1. Introducción

Una de las razones que los creadores de UML argumentan para que se utilice como lenguaje de modelado es que resulta fácil de interpretar por sus usuarios. Sin embargo, la existencia de diferentes representaciones dinámicas en UML permite que haya variaciones en dicha interpretación. Por ello, es importante conseguir que estos modelos dinámicos comuniquen la información de forma consistente a todas las personas implicadas en un sistema informático.

En este capítulo presentamos una primera investigación empírica con el objeto de comparar los diferentes modelos que UML incluye para la especificación del comportamiento dinámico de una aplicación software. Esta comparación se planteó desde la perspectiva de la comprensión de los artefactos obtenidos en la etapa de diseño. Así pues, todas las fases del proceso experimental para llevar a buen término nuestra primera evaluación empírica se describen en los apartados 5.2 a 5.7.

5.2. Definición del objetivo del primer experimento

5.2.1. Objetivo

Si trasladamos a nuestra investigación los cinco parámetros correspondientes a la plantilla del método GQM, entonces nuestra meta queda definida de la siguiente forma:

"Analizar *los modelos dinámicos de los diseños OO escritos en UML* para *evaluar* su *comprensión* desde el punto de vista del *desarrollador de software* en el contexto de *una clase de segundo ciclo de ingeniería informática*."

A continuación, ya estamos en disposición de decidir las variables experimentales, así como de concretar las hipótesis que intervienen en nuestra primera investigación.

5.3. Planificación del primer experimento

5.3.1. Variables experimentales

Este experimento inicial manipula dos variables independientes:

- DIAGRAM: Es el tipo de diagrama empleado en los documentos de diseño para representar el comportamiento dinámico en UML. Se evaluaron los diagramas de secuencia, de colaboración y de estado.
- APPLICATION: Es el tipo de dominio de aplicación de los documentos de diseños. Se usaron tres diseños, de diferentes dominios.

Las variables dependientes que vamos a examinar son:

- Tiempo global (TSEC): Es la suma del tiempo transcurrido en contestar a cada una de las preguntas. Cada tiempo se mide en centésimas de segundo y luego el tiempo total se redondea a segundos.
- Puntuación total (NRESP): Es la cuantía de respuestas acertadas. Una respuesta se contabiliza como 1 cuando es correcta y como 0 si es incorrecta.

Las variables independientes están en una escala nominal y las dependientes se miden en una escala ratio.

5.3.2. Hipótesis nulas

Es evidente que en este estudio empírico la comprensión del modelado dinámico se mide en base a las dos variables dependientes descritas en el apartado 5.3.1. La Tabla 5.1 recoge las hipótesis nulas de este primer experimento, donde las filas corresponden a los factores o a su interacción y las columnas a las variables respuesta. La intersección de una fila y de una columna es una hipótesis nula, cuyo primer subíndice en mayúsculas indica la variable independiente (o interacción entre varios factores) y segundo subíndice en minúsculas, la variable dependiente.

	TSEC	NRESP
DIAGRAM x APPLICATION No hay diferencias entre las 3x3 condiciones experimentales con respecto a ...	$H_{0\text{-DIAGxAPPLIC-tsec}}$	$H_{0\text{-DIAGxAPPLIC-nresp}}$
DIAGRAM No hay diferencias entre los sujetos que utilizan los tres tipos de diagramas dinámicos con respecto a ...	$H_{0\text{-DIAGRAM-tsec}}$	$H_{0\text{-DIAGRAM-nresp}}$
APPLICATION No hay diferencias entre los sujetos que leen los tres documentos de diseño UML con respecto a ...	$H_{0\text{-APPLICATION-tsec}}$	$H_{0\text{-APPLICATION-nresp}}$

Tabla 5.1. Hipótesis nulas del primer experimento

5.3.3. Sujetos experimentales

Teniendo en cuenta la dificultad que conlleva experimentar entre los profesionales del mundo informático, este estudio empírico lo realizamos con 18 estudiantes del segundo ciclo de Ingeniería Informática en la Facultad de Informática de la Universidad del País Vasco.

5.3.4. Material experimental

En este apartado se presentan los instrumentos que han sido necesarios para la realización del primer experimento. Asimismo, el material específico de esta práctica empírica se encuentra detallado en el Apéndice A:

- Apuntes acerca de la notación UML.
- Tres documentos de diseño escritos en el lenguaje de modelado UML.

  Los tres documentos corresponden a tres aplicaciones diferentes: un Teléfono Móvil [Mart98], un Sistema de Biblioteca [Erik98] y un Dictáfono Digital [Porr99]. Estos diseños solamente hacen referencia a una parte de dichos sistemas software. Como mínimo, cada uno de ellos contiene el diagrama de todos los casos de uso, un diagrama de paquetes y un diagrama de clases. Sin embargo, para el modelado dinámico se especifica el comportamiento de un único caso de uso, que va a ser el mismo en los tres tipos de diagramas. Este hecho implica la transformación de la notación de un comportamiento en otra. Por ejemplo, es necesario convertir un modelo dinámico expresado mediante un diagrama de secuencia en uno de colaboración (ver Figuras A.3 y A.4 correspondientes al diagrama de secuencia y de colaboración del caso de uso HacerLlamadaTelefónica de la aplicación Teléfono Móvil). También hay que considerar la construcción de un diagrama de secuencia en uno de estado. Para esta conversión puede ser necesario incluir uno o varios diagramas de estado. En la especificación del comportamiento de un caso de uso específico se instancian varias clases. Algunas de ellas cambian de estado y, en

consecuencia, se deben añadir los diagramas de estado correspondientes a dichas clases (ver Figuras A.10 y A.12 pertenecientes al diagrama de secuencia y a los de estado del caso de uso PrestarEjemplar de la aplicación Sistema de Biblioteca).

- Tres cuestionarios de preguntas y respuestas de elección múltiple.
  Cada cuestionario se compone de 5 preguntas, cada una de ellas redactada en una página distinta junto con las 4 posibles respuestas.
- Cronómetros digitales.
  Cada alumno dispuso de un cronómetro digital CASIO HS-3 para la medición del tiempo en cada respuesta. Estos cronómetros fueron facilitados por el Departamento de Física Aplicada de la E.U.I.T.I. e I.T.T. de Vitoria-Gasteiz.

5.3.5. Diseño experimental

Como los dos factores a manipular tienen tres niveles, el número total de condiciones experimentales es 3 x 3 = 9. Dado que el número de individuos es escaso, en concreto 18, no es posible plantear un diseño factorial 3 x 3 completo, puesto que para cada tratamiento o condición experimental sólo podríamos disponer de 18 / 9 = 2 datos. Por otra parte, tampoco es factible medir a los sujetos en todos los tratamientos (ver Tabla 5.2). La razón se debe a los propios límites que imponemos a nuestro experimento para evitar los efectos de aprendizaje. Una vez que un individuo ha visto un documento de diseño en particular, es incorrecto permitir que utilice el mismo documento en la siguiente sesión del experimento. Si esto sucediese adquiriría conocimiento sobre el material elaborado, lo que implicaría una amenaza a la validez de los resultados. Bajo estas premisas, planteamos un diseño factorial 3 x 3 de medidas repetidas, donde cada sujeto se midió tres veces, lo que nos proporciona 54 / 9 = 6 mediciones. Pero para conseguir un factorial completo, nuestro diseño se divide en 3 grupos de bloques de tamaño 3, tal como aparece en [Wine91, pp. 590-595] y en [Dean99, pp. 462-464]. Según la nomenclatura de [Kirk95, pp. 650-654] se trata de un factorial confundido de cuadrado latino 3^2 o LSCF-3^2 (*Latin Square Confounded Factorial design*).

Con la técnica del bloqueo en un factorial 3 x 3 reducimos el número de combinaciones de tratamientos, que luego se asignan a los bloques y, a su vez, éstos a los grupos. En contrapartida, esta reducción viene acompañada de la confusión del efecto de estos grupos de bloques bien con el efecto de uno de los factores o bien con el efecto de su interacción, lo que repercute en la pérdida de información. Pero, ¿qué criterio debemos adoptar a la hora de construir dichos bloques? A priori, el principio de confusión recomienda hacer coincidir los

bloques con las interacciones de mayor orden, tal como sucede en [Basi96b] y [Rope97]. Ambos experimentos utilizan la confusión completa con la interacción A x B, asumiendo que el efecto de la interacción es despreciable. El análisis del factorial 2 x 2 de [Basi96b], donde se emplea la confusión completa pues en un factorial 2 x 2 no hay otra posibilidad, confirma que la interacción sí se puede desechar. Por el contrario, en el análisis del factorial 3 x 3 de [Rope97] la interacción de los dos factores resultó ser significativa estadísticamente y por lo tanto no debería ser despreciada.

Siempre que sea posible, ninguna interacción debe confundirse completamente [Coch80, p. 246]. Por esta razón, preferimos el diseño para un factorial 3 x 3 con A x B (DIAGRAM x APPLICATION) parcialmente confundida. Y puesto que la interacción A x B tiene dos componentes independientes: AB y AB^2 (más detalles en [Wine91, p. 591]), para la construcción de los grupos de bloques elegimos confundirlos con la componente AB^2, tal como se explica en el apartado 4.5.2.1. El procedimiento para ubicar los tratamientos en los bloques se realizó de acuerdo a la ecuación que define esta componente: a + 2b = r (mod 3), mientras que los bloques (sujetos) se asignaron al azar a los grupos.

	Sesión I			Sesión II			Sesión III		
	TEL	BIB	DIC	TEL	BIB	DIC	TEL	BIB	DIC
Secuencia	1,10	4,13	7,16	3,12	6,15	9,18	2,11	5,14	8,17
Colaboración	8,17	2,11	5,14	7,16	1,10	4,13	9,18	3,12	6,15
Estado	6,15	9,18	3,12	5,14	8,17	2,11	4,13	7,16	1,10

Tabla 5.2. Diseño factorial confundido de cuadrado latino 3^2

Los sujetos numerados 1, 2, 3, 10, 11 y 12 en la Tabla 5.2 se asignaron a los bloques del Grupo 0 (r = 0), que está formado por las siguientes condiciones experimentales: Secuencia (a = 0) y TEL (b = 0), Colaboración (a = 1) y BIB (b = 1), Estado (a = 2) y DIC (b = 2). Los sujetos 7, 8, 9, 16, 17 y 18 a los del Grupo 1 (r = 1); y los sujetos 4, 5, 6, 13, 14 y 15 a los bloques del Grupo 3 (r = 2). Como hay tres tratamientos en cada bloque (tamaño del bloque = 3), cada estudiante se midió tres veces. El hecho de tomar tres medidas implicó que no fuese lógico utilizar un único documento de diseño. Así pues, se elaboraron tres documentos de diseño diferentes, correspondientes a las siguientes aplicaciones: TELéfono Móvil, Sistema de BIBlioteca y DICtáfono Digital.

5.3.6. Validez experimental

Un paso crucial en el diseño experimental es minimizar el impacto de las amenazas a la validez experimental, es decir, de aquellos factores que puedan afectar a las variables dependientes sin el conocimiento del investigador. A continuación se explica la manera en que controlamos algunos de estos factores en este primer experimento.

5.3.6.1. Validez constructiva

En la construcción de un experimento es importante elegir bien las variables independientes y dependientes. Precisamente [Scan89] demostró que las conclusiones de un experimento anterior no podían ser válidas debido a una serie de fallos. Uno de ellos tuvo su origen en la construcción del estudio empírico, pues no consideraron el tiempo como variable dependiente. Por ello en el proceso de seleccionar las variables dependientes tuvimos en cuenta no sólo la puntuación obtenida, sino también el tiempo de comprensión.

5.3.6.2. Validez interna

Mediante nuestro diseño hemos controlado las amenazas a la validez interna, y también medido algunas de ellas, de la siguiente forma:

- Efectos de selección: control mediante la técnica de aleatorización

  Cada alumno se asignó al azar a una de las 9 condiciones experimentales. De esta forma las diferencias en la capacidad intelectual de cada sujeto se diseminan por igual a través de todos los tratamientos del experimento.
- Efectos de instrumentación: medición

  Este factor se va a medir mediante el análisis de varianza en la variable APPLICATION.
- Efectos de madurez: control y medición

  Manipulando el orden en que se estudian tanto los tipos de diagramas como los documentos de diseño, se minimizan estos efectos ya que los sujetos no pueden mejorar su rendimiento en las sesiones sucesivas. Además para evitar que el cansancio y la práctica perjudicasen los resultados, el experimento se llevó a cabo en tres días.

  Esta amenaza no sólo se controló mediante el diseño experimental, sino que también se midió en la variable ROUND (madurez). Esta variable es independiente del factor APPLICATION (instrumentación), puesto que en cada sesión se utilizaron tres diseños diferentes. Por ello, ROUND se puede considerar como otra variable independiente.

  En el apartado 5.5.2 se verificó que no hubo efecto de madurez o de aprendizaje.

- Efectos de la práctica o del orden de presentación: control

  En este diseño no se eligió ningún orden de presentación para ninguno de los dos factores (DIAGRAM y APPLICATION). Tal como ya se indicó en el apartado 5.3.5, cada día se asignó a cada sujeto un tipo de diagrama y de documento de diseño diferentes. De esta manera se controló esta amenaza. Sin embargo, podríamos correr el riesgo de que los sujetos compartiesen información sobre el material experimental de una sesión a otra, es decir, lo que se denomina el efecto de plagio.

5.3.6.3. Validez externa

La validez externa consiste en poder generalizar los resultados obtenidos a otras situaciones de la vida cotidiana. Pero debido a que los experimentos controlados se realizan como trabajo de laboratorio, es difícil obtener resultados que se puedan extrapolar al mundo real. En nuestro caso, lo ideal sería plantear estudios empíricos con profesionales y utilizando documentos de diseño propios de la industria del software. En primer lugar, hay pocas empresas que tengan implantado UML como lenguaje de modelado en sus diseños. Por otra parte, mientras la industria del software no vislumbre que mediante la experimentación vaya a lograr beneficios en la generación de sus artefactos, no va a incorporar esta actividad como parte de la política de su negocio. Así pues, una alternativa es ejecutar prácticas empíricas en el ámbito académico con estudiantes de ingeniería informática y con material elaborado por profesionales de la enseñanza.

5.4. Ejecución del primer experimento

La ejecución de cualquier experimento se compone de tres fases:

- Formación

  Los sujetos asistieron a clases de modelado estático y dinámico en notación UML.

- Entrenamiento

  También participaron en un ejercicio de entrenamiento sobre la correcta utilización de los instrumentos. En consecuencia, para esta etapa se utilizó un documento de diseño OO y un cuestionario diferentes a los empleados en el experimento.

  Se llevó a cabo una semana antes, con un diseño relacionado con el funcionamiento de una Máquina de Café [Figu98], que se presenta en el apartado B.1.2 junto con el cuestionario.

- Experimental

En esta etapa a los sujetos se les explicó unas instrucciones de forma oral (ver apartado C.1), y otras por escrito (ver apartado A.1). Cada sesión se efectuó en días diferentes y sin límite de tiempo. Para que no se produjesen equivocaciones en la distribución de los materiales a los alumnos, en cada paquete se introdujo primero el cuestionario, que estaba personalizado para cada alumno, identificado mediante su código y su nombre y apellidos (ver cuestionarios de los apartados A.2, A.3 y A.4).

5.5. Análisis de los datos

Nuestro objetivo es averiguar en qué medida los factores influyen en las dos variables dependientes, que en este estudio empírico explican la comprensión semántica de los diseños escritos en notación UML. Dadas las restricciones del apartado 5.3.5, para la configuración de los datos se utiliza el enfoque univariado, tal como se indica en [SPSS90, pp. 806-807]. Aplicando la técnica del análisis de varianza a dichos datos, vamos a contrastar las hipótesis nulas planteadas en la Tabla 5.1, fijado previamente un nivel de significación $\alpha = 0{,}1$.

En realidad, el análisis se va a realizar bajo dos perspectivas. En la primera de ellas se tiene en cuenta el grupo de bloques como otro factor más. La información obtenida bajo este primer enfoque justifica la necesidad de un segundo análisis, donde la construcción de los bloques de tamaño 3 forma parte del error aleatorio. Los resultados de ambos análisis de varianza se han verificado mediante la utilización de los siguientes paquetes estadísticos: JMP v.4, NCSS 2000 y SPSS v.10.

5.5.1. Primer análisis de varianza

Nuestro diseño factorial coincide con el plan experimental 5 de un cuadrado latino 3 x 3 de medidas repetidas, que se puede consultar en [Wine91, pp. 702-704]. Considerando que las interacciones con el factor grupo no van a ser importantes, el modelo ANOVA para el análisis del experimento es el descrito en el apartado 4.5.2.1:

$$Y_{ijkm} = \mu + C_m + D(C)_{i(m)} + A_j + B_k + (AB)'_{jk} + \varepsilon_{ijkm}$$

(i = 1, ..., 18; j = 1, ..., 3; k = 1, ...,3; m = 1, …, 3)

donde Y_{ijkm} es la respuesta observada (con respecto a una de las dos variables dependientes), μ es la media general, A_j es el efecto principal del tipo de diagrama, B_k es el efecto principal del tipo de dominio de aplicación, C_m es el efecto asociado al factor grupo de bloques, $D(C)_{i(m)}$ son los efectos asociados a los sujetos dentro de los grupos y ε_{ijkm} es el

error experimental. El símbolo $(AB)'_{jk}$ indica que sólo se dispone de información parcial acerca de la interacción A x B, que corresponde a la componente AB y que está libre de confusión. La otra componente AB² está confundida con los grupos de bloques.

Teniendo en cuenta que el factor D, correspondiente a los sujetos, es aleatorio y que los otros son fijos, se trata de un modelo mixto, además de balanceado o equilibrado, al haber el mismo número de observaciones en cada celda. Las razones por las que elegimos este modelo experimental son las siguientes:

- Si la interacción de los factores resulta ser significativa, sólo hay confusión parcial con el efecto principal del grupo de bloques.
- En caso contrario, los factores son independientes y así se consigue que el efecto principal debido bien a DIAGRAM o bien a APPLICATION tenga mayor potencia de contraste, o sea, mayor probabilidad para rechazar correctamente H_0.

5.5.1.1. Variable dependiente: TSEC

Observando los resultados de la Tabla 5.3, el análisis de varianza o ANOVA revela que la componente no confundida AB de la interacción entre el tipo de diagrama y el dominio de aplicación sí es significativa estadísticamente ($F = 3{,}001$; $p = 0{,}065 < \alpha$). Pero el estudio de la otra componente AB² indica que no lo es. Por ello en el apartado 5.5.2.1 se analiza si la interacción DIAGRAM x APPLICATION es estadísticamente significativa o no.

Factor		SC Tipo III	GL	MC	F	p	Potencia
Intersección	Hipótesis	53465429,578	1	53465429,600	164,766	0,000	1,000
	Error	4867405,967	15	324493,731			
A (o DIAGRAM)	Hipótesis	25091,033	2	12545,516	0,409	0,668	0,189
	Error	921127,549	30	30704,252			
B (o APPLICATION)	Hipótesis	967517,851	2	483758,926	15,755	0,000	1,000
	Error	921127,549	30	30704,252			
GROUP (o Componente AB²)	Hipótesis	1394648,182	2	697324,091	2,149	0,151	0,512
	Error	4867405,967	15	324493,731			
Componente AB	Hipótesis	184285,991	2	92142,996	3,001	0,065	0,671
	Error	921127,549	30	30704,252			
SUBJECT (GROUP)	Hipótesis	4867405,967	15	324493,731	10,568	0,000	1,000
	Error	921127,549	30	30704,252			

Tabla 5.3. Primer ANOVA para la variable TSEC (Tiempo total)

5.5.1.2. Variable dependiente: NRESP

Los resultados del análisis de varianza con respecto a NRESP, resumidos en la Tabla 5.4, son inversos a los obtenidos con TSEC. La exploración de sus datos indica que es estadísticamente significativa la componente AB^2 de la interacción entre los dos factores objeto de estudio ($F = 4{,}015$; $p = 0{,}040 < \alpha$), cuyo efecto está completamente confundido con el efecto principal del grupo. Por el contrario, la otra componente AB no resulta de importancia. En el apartado 5.5.2.2 se analiza estadísticamente la interacción A x B.

Factor		SC Tipo III	GL	MC	F	p	Potencia
Intersección	Hipótesis	718,685	1	718,685	424,606	0,000	1,000
	Error	25,389	15	1,693			
A (o DIAGRAM)	Hipótesis	0,481	2	0,241	0,202	0,818	0,144
	Error	35,778	30	1,193			
B (o APPLICATION)	Hipótesis	6,481	2	3,241	2,717	0,082	0,631
	Error	35,778	30	1,193			
GROUP (o Componente AB^2)	Hipótesis	13,592	2	6,796	4,015	0,040	0,756
	Error	25,592	15	1,693			
Componente AB	Hipótesis	4,593	2	2,296	1,925	0,163	0,501
	Error	35,778	30	1,193			
SUBJECT (GROUP)	Hipótesis	25,389	15	1,693	1,419	0,201	0,796
	Error	35,778	30	1,193			

Tabla 5.4. Primer ANOVA para la variable NRESP (Puntuación total)

5.5.2. Segundo análisis de varianza

Dado el efecto significativo de una de las dos componentes con respecto a TSEC y a NRESP, es necesario obtener información conjunta acerca de los 4 grados de libertad (GL) de la interacción A x B. Para ello los datos se deben analizar como un factorial 3 x 3 de bloques al azar [Coch80, p. 227]. En este caso se considera que la asignación de los tratamientos a los bloques no obedece a ley alguna, sino que se realiza aleatoriamente. Por otra parte, el análisis de varianza demostró que no hubo aprendizaje, pues no fue estadísticamente significativa con respecto a las variables TSEC ($F = 0{,}188$; $p = 0{,}830 > \alpha$) y NRESP ($F = 0{,}860$; $p = 0{,}429 > \alpha$).

5.5.2.1. Variable dependiente: TSEC

Según el ANOVA de la Tabla 5.5, debemos rechazar la hipótesis nula $H_{0\text{-DIAGxAPPLIC-tsec}}$ con respecto al tiempo global. El análisis revela que la interacción entre el tipo de diagrama y el dominio de aplicación es significativa ($F = 3{,}069$; $p = 0{,}026 < \alpha$). Asimismo dicho análisis confirma que:

- Se cumple la relación indicada en [Wine91, p. 704]:

$$SC_{AxB} = SC_{GROUP\ or\ AB^2} + SC_{AB},$$

donde la suma de cuadrados (SC) de las dos cifras numéricas que aparecen subrayadas en la Tabla 5.3 verifica que 1394648,128 + 184285,991 = 1578934,173, cuyo resultado corresponde a la columna SC de la interacción DIAGRAM x APPLICATION en la Tabla 5.5.

- En el supuesto de que la interacción anterior no fuese de importancia, la potencia observada en los 2 factores (DIAGRAM y APPLICATION) es superior en el primer análisis de varianza que en el segundo (comparar Tablas 5.3 y 5.5).

Factor	SC Tipo III	GL	MC	F	p	Potencia
Modelo	2571543,057	8	321422,882	2,499	0,025	0,921
Intersección	53465429,578	1	53465429,678	415,640	0,000	1,000
DIAGRAM	25091,033	2	12545,516	0,098	0,907	0,121
APPLICATION	967517,851	2	483758,926	3,761	0,031	0,771
DIAGRAM x APPLICATION	1578934,173	4	394733,543	3,069	0,026	0,857
Error	5788533,513	45	128634,078			
Total	61825506,151	54				

Tabla 5.5. Segundo ANOVA para la variable TSEC (Tiempo total)

Cuando una interacción es estadísticamente significativa, no se analizan los efectos principales de las variables que intervienen en dicha interacción. En otras palabras, no es posible investigar cualquier diferencia significativa entre los tres tipos de modelos dinámicos mientras no sea posible separar su efecto del dominio de aplicación estudiado. Así pues, se deben examinar los efectos simples para cada variable independiente (DIAGRAM y APPLICATION). Es una consecuencia lógica que se deriva del descubrimiento de una interacción significativa entre dichos factores. Los resultados se presentan en la Tabla 5.6 y su significado se estudia en el apartado 5.6.

Factor	SC Tipo III	GL	MC	F	p
WITHIN + RESIDUAL	5812723,19	47	123674,96		
APPLICATION within Secuencia	29624,76	2	14812,38	0,12	0,887
APPLICATION within Colaboración	1632750,73	2	816375,36	6,60	0,003
APPLICATION within Estado	883991,61	2	441995,80	3,57	0,036
Modelo	2546367,09	6	424394,52	3,43	0,007
Total	8359090,28	53	157718,68		

Tabla 5.6. Efectos simples de TSEC para el factor APPLICATION

5.5.2.2. Variable dependiente: NRESP

Según el ANOVA de la Tabla 5.7, debemos rechazar la hipótesis nula $H_{0\text{-DIAGxAPPLIC-nresp}}$ con respecto a la puntuación total. El análisis ratifica la interacción significativa entre el tipo de diagrama y el dominio de aplicación del diseño OO. Análogamente, dicho estudio confirma la relación indicada en [Wine91, p. 704], donde la columna SC de la interacción DIAGRAM x APPLICATION en la Tabla 5.7 es igual a la suma de las SC de sus dos componentes: 18,185= 13,592+ 4,593, cantidades subrayadas en la Tabla 5.4. También se observa que en el caso de que la interacción no resultase relevante, con el primer análisis se hubiese logrado mayor potencia correspondiente a los efectos principales de los factores que se investigan (comparar Tablas 5.4 y 5.7).

Factor	SC Tipo III	GL	MC	F	p	Potencia
Modelo	25,148	8	3,144	2,313	0,036	0,898
Intersección	718,685	1	718,685	528,733	0,000	1,000
DIAGRAM	0,481	2	0,241	0,177	0,838	0,139
APPLICATION	6,481	2	3,241	2,384	0,104	0,589
DIAGRAM x APPLICATION	18,185	4	4,546	3,345	0,018	0,885
Error	61,167	45	1,359			
Total	805,000	54				

Tabla 5.7. Segundo ANOVA para la variable NRESP (Puntuación total)

Al igual que en el apartado 5.5.2.1, es imposible separar el efecto de DIAGRAM del efecto de APPLICATION. En un principio esto se podría considerar como un fallo en el experimento (por ejemplo, los tres documentos de diseño no son lo bastante similares). Sin embargo, una interpretación más reflexiva es que cada diagrama dinámico contiene unas semánticas más acordes según la complejidad y el dominio de aplicación de los diseños OO. Para verificarlo, se presenta un análisis de varianza de efectos simples para cada tipo de dominio en la Tabla 5.8, así como una comparación de estos resultados junto con los anteriores en el apartado 5.6.

Factor	SC Tipo III	GL	MC	F	p
WITHIN + RESIDUAL	61,65	47	1,31		
APPLICATION within Secuencia	7,44	2	3,72	2,84	0,069
APPLICATION within Colaboración	10,11	2	5,06	3,85	0,028
APPLICATION within Estado	7,11	2	3,56	2,71	0,077
Modelo	24,67	6	4,11	3,13	0,011
Total	86,31	53	1,63		

Tabla 5.8. Efectos simples de NRESP para el factor APPLICATION

5.6. Interpretación de los resultados

Un resumen de los datos recopilados con respecto al tiempo se muestra en la Tabla 5.9. La intersección de una fila con una columna proporciona la media y la desviación típica del tiempo total: $\overline{X}_{t\,\mathrm{sec}}(S_{t\,\mathrm{sec}})$, para cada tipo de diagrama dinámico y dominio de aplicación.

	Diagrama de Secuencia	Diagrama de Colaboración	Diagrama de Estado
Teléfono Móvil	16:31 (05:39)	09:46 (03:55)	14:14 (07:26)
Sistema de Biblioteca	15:56 (05:55)	22:02 (06:20)	14:27 (03:42)
Dictáfono Digital	17:34 (04:06)	16:29 (06:35)	22:11 (08:18)

Tabla 5.9. Resumen estadístico del primer experimento con respecto al tiempo total (mm:ss)

Tanto en la Tabla 5.9 como en la Figura 5.1 se observa que el tiempo medio en contestar a todas las preguntas de los diferentes diagramas no se distribuye uniformemente a lo largo de los documentos. La excepción es el diagrama de secuencia, donde la rapidez en las respuestas no depende del dominio de aplicación de los diseños, hecho que se constata también en el análisis de los efectos simples de la Tabla 5.6 con respecto a TSEC ($F = 0{,}12$; $p = 0{,}887 > \alpha$). Por otra parte, sí existen diferencias significativas entre los tres diseños en los otros dos tipos de diagramas, tal como muestra el análisis de varianza de los efectos simples de la variable TSEC en la Tabla 5.6.

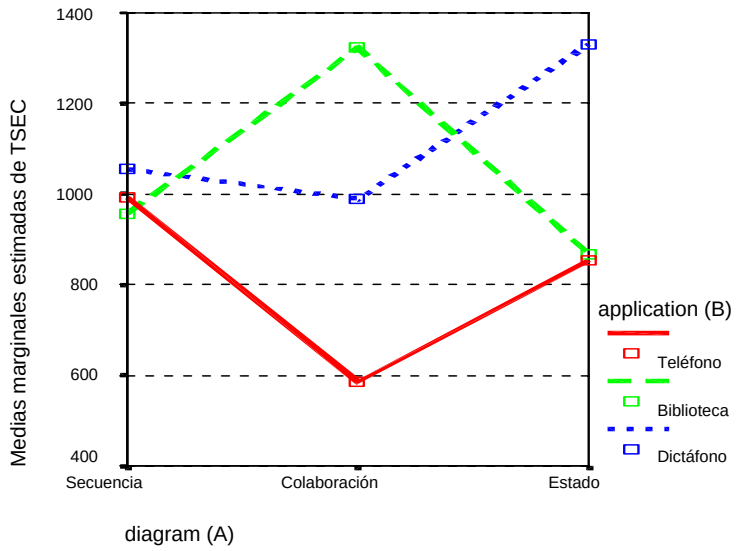


Figura 5.1. Gráfico de perfil del tiempo total (en segundos)

En la Tabla 5.10 se presentan estadísticos descriptivos de las puntuaciones obtenidas por los sujetos al responder a cada cuestionario. La intersección de una fila con una columna indica la media y la desviación típica de la puntuación total: $\overline{X}_{nresp}(S_{nresp})$, para cada tipo de diagrama dinámico y dominio de aplicación.

	Diagrama de Secuencia	Diagrama de Colaboración	Diagrama de Estado
Teléfono Móvil	4,5 (0,55)	4,5 (0,84)	3,33 (1,21)
Sistema de Biblioteca	3 (1,26)	3,5 (1,38)	3,33 (1,37)
Dictáfono Digital	3,33 (1,21)	2,67 (1,63)	4,67 (0,52)

Tabla 5.10. Resumen estadístico del primer experimento con respecto a la puntuación total

En la Figura 5.2 y en la Tabla 5.10 se observa que la puntuación más elevada se consigue con los diagramas de secuencia y de colaboración en la aplicación Teléfono Móvil y con el diagrama de estado en el Dictáfono Digital. En el caso del diseño del Teléfono Móvil no hay diferencias reales entre las interacciones: (a) entre los objetos representados de forma secuencial a través del tiempo (diagrama de secuencia) o (b) entre los objetos representados de forma espacial (diagrama de colaboración). Algo parecido sucede con el diseño del Sistema de Biblioteca, puesto que el número total de respuestas correctas es similar independientemente del tipo de diagrama dinámico (ver Tabla 5.10 y la línea punteada correspondiente a la Biblioteca en la Figura 5.2).

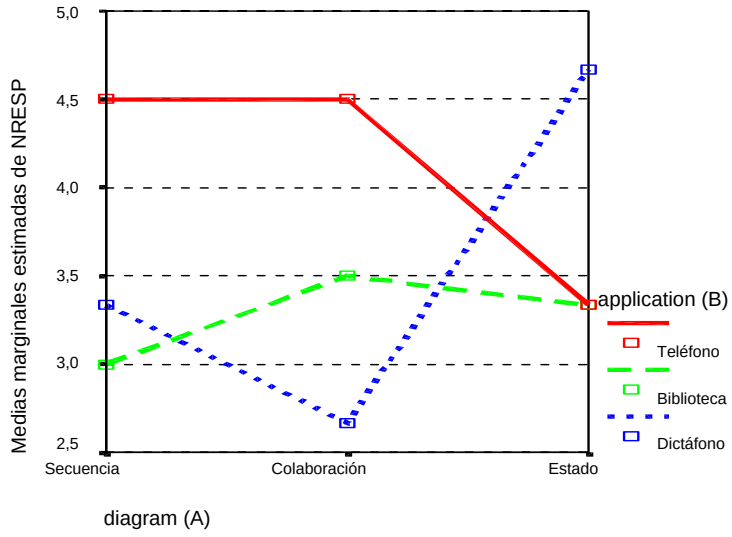


Figura 5.2. Gráfico de perfil de la puntuación total (5 puntos máximo)

En los gráficos de perfil de las Figuras 5.1 y 5.2 no hay líneas paralelas. Esta característica ratifica el resultado estadístico significativo de la interacción entre los factores DIAGRAM y APPLICATION (ver apartado 7.6). Por ello, si observamos estos gráficos de forma paralela, podemos hacer una comparación de los documentos de diseño para cada tipo de diagrama:

- Diagrama de secuencia

  Tal como muestra la Figura 5.1, la complejidad de los diseños y de los dominios de aplicación no influye en el tiempo necesario para comprenderlos. En la Figura 5.2, sin embargo, hay diferencias a favor del Teléfono con respecto a la variable NRESP (puntuación total), debido a la menor complejidad del documento del Teléfono comparado con los diseños de la Biblioteca y del Dictáfono (ver apartado 6.3.4.1).

- Diagrama de colaboración

  A primera vista el diagrama de colaboración parece ser el menos comprensible, puesto que en la Figura 5.1 el tiempo más alto se consigue con el Sistema de Biblioteca y en la Figura 5.2 la puntuación más baja se logra con el Dictáfono Digital. Pero una observación más detallada de la Figura 5.1 nos indica que la relación entre los diseños es TEL<DIC<BIB, considerando el tiempo total de menor a mayor. Si ahora consideramos la puntuación total de mayor a menor, la relación anterior es TEL>BIB>DIC, tal como se puede verificar en la Figura 5.2. Es decir, a medida que se incrementa la complejidad del escenario, cuantificada de menor a mayor como TELéfono<BIBlioteca<DICtáfono en la Tabla 6.3, más tiempo es necesario invertir en la comprensión de este diagrama, y menos aciertos se obtienen. Además, estas diferencias estadísticamente significativas se pueden comprobar en la Tabla 5.6 para el tiempo total ($F = 6,60$; $p = 0,003 < \alpha$) y en la Tabla 5.8 para la puntuación global ($F = 3,85$; $p = 0,028 < \alpha$).

- Diagrama de estado

  Examinando las Figuras 5.1 y 5.2, el tiempo total para comprender el Teléfono Móvil y el Sistema de Biblioteca es similar, así como el número de respuestas correctas. Pero, el ANOVA revela diferencias significativas en la Tabla 5.6 para TSEC ($F = 3,57$; $p = 0,036 < \alpha$) y en la Tabla 5.8 para NRESP ($F = 2,71$; $p = 0,011 < \alpha$). Estos resultados se deben al diseño del Dictáfono Digital, ya que el diagrama de estado demuestra ser el más adecuado para especificar el comportamiento de un sistema en tiempo real reactivo. Este diagrama proporciona el mayor número de aciertos, a expensas también de consumir más tiempo.

5.7. Presentación

Con el fin de que el experimento pueda ser replicado, los materiales elaborados deben estar disponibles. Asimismo es importante que los datos recopilados estén accesibles. Toda esta información se puede encontrar:

- En este documento

  Material: Apéndice A

  Datos: Apéndice D
- En el CD dentro de la carpeta Experimento1
- En la página WEB http://www.sc.ehu.es/jiwdocoj/ieadm/ieadmUML.htm (en inglés).

Capítulo 6

Réplica del modelado dinámico en UML

6.1. Introducción

Tras una evaluación inicial de la comprensión del modelado dinámico es aconsejable replicar el experimento del Capítulo 5. Aunque, en un principio, por "replicar" se entiende repetir el experimento bajo las mismas condiciones iniciales, lo cierto es que las réplicas se agrupan en las tres categorías siguientes [Basi99]:

1. Réplicas que no varían ninguna de las hipótesis de la investigación, es decir, no varían ninguna de las variables independientes y dependientes del experimento original.
   1.1. Réplicas estrictas.
   1.2. Réplicas que varían la manera en que se ejecuta el experimento.
2. Réplicas que varían las hipótesis de la investigación.
   2.1. Réplicas que cambian las variables independientes.
   2.2. Réplicas que cambian las variables dependientes.
   2.3. Réplicas que modifican las variables de contexto en el entorno en que se evaluó la solución: por ejemplo, utilizar profesionales, en lugar de estudiantes, en una réplica con los mismos procesos y productos para ver si se obtienen los mismos resultados.
3. Réplicas que amplían la teoría; es decir, se realizan varios cambios para comprobar si los resultados permanecen.

En nuestro caso, según la clasificación anterior, la réplica se engloba dentro del tipo 2.1, pues incorpora una nueva variable independiente, tal como se explica en el apartado 6.3.1. Además, esta réplica forma parte de una práctica empírica más compleja, desde el momento en que el objetivo planteado en el experimento original deriva en la formulación de dos objetivos más específicos, que originan las dos hipótesis nulas de alto nivel del apartado 6.2.1.

6.2. Definición de la réplica y del tercer experimento

6.2.1. Hipótesis de alto nivel

Tras el objetivo general del primer experimento, descrito en el apartado 5.2.1, subyacen dos cuestiones importantes, que constituyen las dos hipótesis de alto nivel. Dichas hipótesis consisten en examinar si los modelos dinámicos de los diseños UML expresados mediante

- diagramas de secuencia, de colaboración o de estado (Hipótesis nula 1), o bien
- combinaciones de pares de diagramas: colaboración-estado, secuencia-estado o secuencia-colaboración (Hipótesis nula 2),

difieren en cuanto al número de respuestas correctas y al tiempo total en contestar a todas las preguntas. Para dar respuesta a estas dos hipótesis de alto nivel se planifica una investigación donde los mismos sujetos participan en dos experimentos que, además de complementarse, están condicionados entre sí, tal como muestra la Tabla 6.1. La restricción impuesta es que si a un individuo le corresponde en el segundo experimento (réplica), por ejemplo, el diagrama de secuencia, entonces en el tercer experimento se excluye este tipo de diagrama y le corresponde el par colaboración-estado. De esta forma se evita que haya un efecto de aprendizaje con respecto al diagrama de modelado dinámico. También entre ambos experimentos se planificó un intervalo de diez días.

Segundo Experimento (Réplica)	Tercer Experimento
Secuencia	Colaboración-Estado
Colaboración	Secuencia-Estado
Estado	Secuencia-Colaboración

Tabla 6.1. Restricción impuesta al segundo y tercer experimento

El **Segundo Experimento (Réplica)** para contrastar la Hipótesis nula 1 se describe en este Capítulo 6, así como el análisis estadístico de los resultados, mientras que el diseño y la exploración de los datos recopilados del **Tercer Experimento** correspondiente a la Hipótesis nula 2 se explican en el Capítulo 7.

6.3. Planificación de la réplica

Este segundo experimento es una replicación interna de un estudio empírico anterior [Oter02]. Ambos son idénticos en cuanto al diseño básico y a los tipos de diagramas y de documentos de diseño empleados. Ahora bien, esta réplica se puede considerar como una

extensión del experimento original, pues se ha añadido una variable independiente, que permite medir el efecto debido a una de las amenazas a la validez interna. Dicha ampliación implica no sólo un incremento en el número de las hipótesis concretas derivadas de la Hipótesis nula 1 de alto nivel, sino también una mayor complejidad en el diseño experimental.

6.3.1. Variables experimentales

El experimento de la réplica manipula tres variables independientes, las dos primeras ya explicadas en el apartado 5.3.1:

- DIAGRAM: Se estudiaron el diagrama de secuencia, de colaboración y de estado.
- APPLICATION: Se usaron tres diseños diferentes en cuanto a su complejidad y a su área de aplicación: un Teléfono Móvil (sistema empotrado), un Sistema de Biblioteca (sistema de gestión) y un Dictáfono Digital (sistema en tiempo real reactivo).
- GROUP: Es la secuencia o el orden de presentación del documento de diseño correspondiente a cada aplicación. Como la variable independiente APPLICATION tiene 3 niveles, se obtienen 3! = 6 secuencias diferentes. Por ejemplo, el grupo 1 (G1) corresponde a la secuencia Teléfono-Biblioteca-Dictáfono y así para las restantes combinaciones, tal como figura en la Tabla 6.4. De esta forma se puede detectar y medir la presencia del efecto debido a la práctica o a la secuencia.

En cada sesión se midieron las dos variables dependientes: TSEC y NRESP, tal como se llevó a cabo en el experimento original. Las variables independientes se codifican en una escala nominal, mientras que las dependientes en una escala ratio.

6.3.2. Hipótesis nulas concretas derivadas de la Hipótesis nula 1

Continuando con la aplicación del método GQM (Objetivo/Pregunta/Métrica), para contrastar la Hipótesis nula 1 nos planteamos la siguiente cuestión: ¿Qué influencia tiene cada variable independiente (DIAGRAM, APPLICATION y GROUP) en el número de respuestas acertadas y en el tiempo total requerido por los sujetos para comprender los documentos de diseño? La respuesta a esta pregunta implica el refinamiento de dicha Hipótesis en las hipótesis nulas concretas de la Tabla 6.2. La formulación de una hipótesis nula se obtiene por medio de la intersección de una fila (o variable independiente) con una columna (o variable dependiente).

	TSEC	NRESP
DIAGRAM x APPLICATION No hay diferencias entre las 3x3 condiciones experimentales con respecto a ...	$H_{0\text{-DIAGxAPPLIC-tsec}}$	$H_{0\text{-DIAGxAPPLIC-nresp}}$
DIAGRAM No hay diferencias entre los sujetos que utilizan los tres tipos de diagramas dinámicos con respecto a ...	$H_{0\text{-DIAGRAM-tsec}}$	$H_{0\text{-DIAGRAM-nresp}}$
APPLICATION No hay diferencias entre los sujetos que leen los tres documentos de diseño UML con respecto a ...	$H_{0\text{-APPLICATION-tsec}}$	$H_{0\text{-APPLICATION-nresp}}$
GROUP No hay diferencias entre las seis secuencias en que los sujetos leen los documentos de diseño UML con respecto a ...	$H_{0\text{-GROUP-tsec}}$	$H_{0\text{-GROUP-nresp}}$

Tabla 6.2. Hipótesis nulas concretas del segundo experimento (réplica)

Sólo se va a analizar la interacción DIAGRAM x APPLICATION, pues no tiene sentido estudiar la interacción de factores que no se cruzan totalmente, es decir, cuando todos los niveles de un factor no se cruzan con todos los niveles del otro (ver apartado 6.3.5).

6.3.3. Sujetos experimentales

Aunque para la experimentación en ingeniería del software sea más lógico utilizar como sujetos experimentales a los propios profesionales, en la realidad resulta difícil disponer de tales sujetos. Un estudio comparativo entre estudiantes y profesionales concluyó que las diferencias eran sólo menores y que la investigación empírica con estudiantes era viable bajo ciertas condiciones, que deben estar claramente explicadas [Höst00]. Por lo tanto, en el segundo (réplica) y tercer experimento participaron en un principio 31 estudiantes (ver apartado 6.3.6.1 sobre la validez interna). Estos sujetos cursaban estudios del segundo ciclo de Ingeniería Informática, por lo que ya habían completado los tres primeros años correspondientes al primer ciclo de dicha carrera.

6.3.4. Material experimental

Evidentemente los instrumentos necesarios para ejecutar esta réplica fueron los mismos que los empleados en el experimento original:

- Tres documentos de diseño escritos en UML.
- Tres cuestionarios de preguntas y respuestas de elección múltiple.
- Cronómetros digitales.

Además se elaboró un cuestionario acerca del lenguaje de modelado UML, que se detalla en el apartado B.1.1, con el fin de clasificar a los sujetos experimentales en cuanto a su conocimiento previo sobre dicho lenguaje.

Dado que al final de este capítulo vamos a comparar los resultados del estudio original con los obtenidos en la réplica, en el apartado 6.3.4.1 tratamos de perfilar el dominio de aplicación y la complejidad del escenario especificado, correspondientes a cada uno de los documentos de diseño empleados.

6.3.4.1. Documentos de diseño UML

Las diferentes aplicaciones utilizadas en el experimento son: un Teléfono Móvil [Mart98], un Sistema de Biblioteca [Erik98] y un Dictáfono Digital [Porr99]. Estos documentos de diseño están formados por modelos que representan tanto la estructura estática como dinámica de dichos proyectos. Con respecto a la primera, cada diseño contiene un diagrama de paquetes y un diagrama de clases. Tal como se menciona en el apartado 5.3.4, la parte dinámica está formada por el diagrama de todos los casos de uso, junto con la especificación de un único escenario.

Estos diseños corresponden a dominios de aplicación diferentes. El documento de diseño de la Biblioteca es un sistema de gestión de información, donde todos los mensajes del escenario especificado son síncronos. Los otros dos documentos pertenecientes al Teléfono y al Dictáfono, en principio, son sistemas en tiempo real. Sin embargo, se diferencian en que el Dictáfono es un sistema reactivo en tiempo real, mientras que el Teléfono no lo es. Esta diferencia implica que los escenarios que implementan cada dominio de aplicación sean distintos. El comportamiento del escenario del Teléfono modela el procedimiento necesario para efectuar una llamada telefónica sin que se produzcan eventos no esperados. Por ello todos sus mensajes son síncronos. En cambio, en el escenario del Dictáfono no sólo se modela el procedimiento para reproducir un mensaje, sino que, además, se incorporan mensajes asíncronos que pueden interrumpir (tener prioridad sobre) dicha reproducción. Por ello la mayoría de los mensajes son síncronos y algunos asíncronos.

Por otra parte, también es importante cuantificar la complejidad de estos diseños. Pero resulta difícil, pues se trata de diseños que documentan una parte de dichas aplicaciones. Además, la mayoría de las métricas de diseño OO a nivel de clase, a nivel de herencia, a nivel de operación, etc. están referidas a la estructura estática de los diseños ([Brit96], [Chid94], [Lore94] y [Marc98]). En nuestro caso, medir el tamaño o la complejidad basándonos en los

diagramas de clases no tiene sentido práctico, ya que las clases incorporan más operaciones y atributos de los que se necesitan para el modelado del comportamiento del escenario.

Aunque se constata la necesidad de tener métricas de complejidad para los diagramas que capturan los aspectos dinámicos de una aplicación software OO [Brit01], existen pocas referencias donde se desarrollen métricas orientadas a los diagramas dinámicos. En [Gene02] se definen y, además, se validan un conjunto de métricas para los diagramas de estado en UML, con el objeto de medir su complejidad. Estas métricas, aunque se refieren a modelos de comportamiento dinámico, se consideran estáticas. Por el contrario, en [Yaco99] se presenta un conjunto de métricas para ROOM (*Real-Time Object Oriented Modelling*) [Seli94], denominadas dinámicas, puesto que se obtienen a partir de modelos de diseño ejecutables.

Como el objetivo es medir la complejidad de un escenario implementado bien en un diagrama de secuencia o bien en uno de colaboración, en nuestra búsqueda hemos encontrado dos referencias que resuelven esta cuestión. Una primera solución es utilizar las métricas OO a nivel de operación propuestas en [Lore94]: tamaño medio de operación (*average operation size*) y número medio de parámetros por operación (*average number of parameters per operation*). Aunque estas métricas se obtienen a partir de los diagramas de clases, se pueden aplicar a los diagramas de interacción tal como se sugiere en [Trav99a]. La otra solución es utilizar las métricas para los casos de uso definidas en [Kim02]: número de actores, número de mensajes y número de clases del sistema asociados a un escenario de un caso de uso. Ambas soluciones son idénticas con respecto a una de las métricas, pues el cálculo del tamaño de operación empleado en [Trav99a] equivale a contabilizar el número de mensajes que intervienen en un escenario en [Kim02]. La Tabla 6.3 muestra el resultado de aplicar dichas métricas a los modelos dinámicos de nuestros diseños, así como el número de mensajes con condiciones de bifurcación y de iteración en dichos escenarios. Por lo tanto, podemos cuantificar la complejidad de los escenarios de mayor a menor según la siguiente desigualdad: Dictáfono > Biblioteca > Teléfono.

Aplicación	Escenario	Actores	Clases	Mensajes	Tipo de mensajes
Teléfono	HacerLlamadaTelefónica	0	6	8	1 repetitivo
Biblioteca	PrestarCopia	1	6	9	1 alternativo
Dictáfono	ReproducirMensaje	1	7	16	4 repetitivos

Tabla 6.3. Métricas correspondientes a cada escenario de nuestros diseños

6.3.5. Diseño experimental

Como el número de factores a manipular es tres, dos de ellos con 3 niveles y el otro con 6, se obtiene un total de 54 condiciones experimentales. Teniendo en cuenta que desarrollar un factorial completo resulta inviable en cuanto a la disponibilidad de recursos humanos, planteamos un diseño factorial fraccional 3^2 x 6, aleatorio y de medidas repetidas en el factor APPLICATION [Wine91, pp. 547-551], tal como se muestra en la Tabla 6.4. A este diseño también se le denomina diseño de parcelas divididas o split-plot SPF 36.3 [Kirk95, pp. 563-567].

Este experimento replicado es aleatorio en cuanto a que cada alumno se asignó al azar a uno de los seis grupos (tercer factor GROUP) y, a su vez, cada grupo se asignó también de forma aleatoria a un tipo de diagrama (primer factor DIAGRAM). Además, es de medidas repetidas dado que el número de alumnos (31) es insuficiente para cubrir el total de tratamientos (54). Por ello, cada alumno se midió tres veces, una vez en cada sesión y para un dominio de aplicación diferente (segundo factor APPLICATION). Por último, se trata de un diseño factorial fraccional, ya que ningún sujeto examinó los tres tipos de diagramas en los tres documentos de diseño y en las seis secuencias diferentes del factor APPLICATION.

		Sesión I	Sesión II	Sesión III
Secuencia	G1 G4	Teléfono Teléfono	Biblioteca Dictáfono	Dictáfono Biblioteca
Colaboración	G2 G5	Biblioteca Biblioteca	Teléfono Dictáfono	Dictáfono Teléfono
Estado	G3 G6	Dictáfono Dictáfono	Teléfono Biblioteca	Biblioteca Teléfono

Tabla 6.4. Diseño factorial split-plot 36.3

En este diseño factorial los sujetos se pueden considerar como un cuarto factor. Esta nueva variable, denominada SUBJECT, está anidada en los niveles del factor GROUP y, por ello, no puede interaccionar con los niveles de GROUP. Lo mismo sucede con la variable GROUP, anidada dentro del factor DIAGRAM. Mientras que la variable APPLICATION es la única que se cruza totalmente con la variable DIAGRAM. Esquemáticamente se muestra en la Figura 6.1 según [Wine91, capítulo 7].

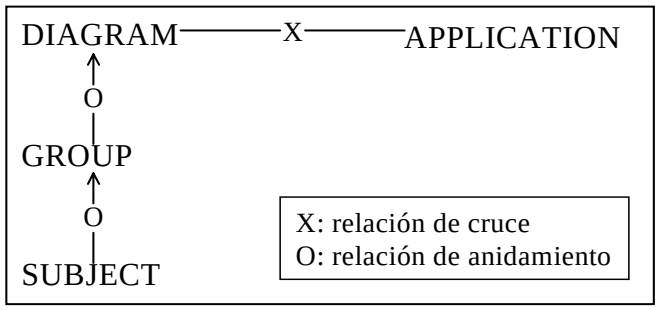


Figura 6.1. Relación entre los factores de la réplica

En el diseño original los sujetos se asignaron a los tres bloques, que se formaron teniendo en cuenta una ecuación (ver apartado 5.3.5) donde estaban implicados los niveles de los dos factores DIAGRAM y APPLICATION. El análisis estadístico reveló que la interacción DIAGRAM x APPLICATION fue significativa, pero dicha interacción estaba parcialmente confundida con el efecto principal del bloque. En consecuencia se pensó en un diseño donde la interacción estuviese libre de confusión. Así pues, en el diseño de la réplica los sujetos (en concreto, grupos de sujetos al haberse incluido un nuevo factor) se asignaron a los tres tipos de diagramas, que en este caso son los tres bloques. Esta correspondencia implica que en el experimento de la réplica el efecto principal del factor DIAGRAM esté confundido con el factor principal del bloque.

6.3.6. Validez experimental

Una de las cuestiones que se deben tener en cuenta a la hora de experimentar en cualquier disciplina es saber cuán válidos son los resultados que hemos obtenido. Por ello es conveniente en la fase de planificación plantearse la validez experimental de nuestros resultados. En general, los tipos de amenazas a la validez se engloban dentro de cuatro categorías: validez de la conclusión, validez constructiva, validez interna y validez externa. Las amenazas a la validez constructiva y externa de la réplica son las mismas que en el primer experimento, que ya se explicaron respectivamente en los apartados 5.3.6.1 y 5.3.6.3. Dado que el diseño experimental de la réplica varió con respecto al diseño original, la forma en que se controlaron las amenazas a la validez interna se detallan en el apartado 6.3.6.1. La fiabilidad de las conclusiones se aborda en el apartado 6.3.6.2.

6.3.6.1. Validez interna

Mediante el diseño experimental hemos controlado la influencia de variables extrañas que pudiesen influir en los resultados sin nuestro conocimiento de la siguiente manera:

- Efectos de selección: control mediante la técnica de grupos emparejados

  Para controlar esta amenaza se crearon grupos emparejados, es decir, de sujetos de parecido rendimiento. La aplicación de la técnica de creación de grupos emparejados a nuestra réplica se encuentra totalmente detallada en el apartado E.1.

- Efectos de instrumentación: medición

  Este factor se va a medir mediante el análisis de varianza en la variable APPLICATION.

- Efectos de mortalidad: control

  Cuando un estudio empírico se realiza durante varios días es lógico que algunos sujetos abandonen la tarea experimental. En nuestro caso, empezaron 31 estudiantes, participando 30 en todas las sesiones y 1 sólo en la primera. Por ello, los datos de este último sujeto no se tuvieron en cuenta en el posterior análisis estadístico.

- Efectos de aprendizaje: control y medición

  En cada sesión no se utilizó un único documento, sino que se emplearon tres documentos diferentes, tal como se muestra en la Tabla 6.4. Esta decisión en el diseño implicó que la variable ROUND no estuviese confundida con (o sea, no dependiese de) APPLICATION y, que se pudiese considerar como otra variable independiente. Así, era posible averiguar mediante la variable ROUND si esta amenaza estaba o no presente en la réplica. Sin embargo, el inconveniente de esta decisión reside en que no se pudo evitar el efecto potencial debido al plagio, en cuanto a que los estudiantes pudieron compartir alguna información sobre el experimento (aunque no tuvieron disponibles materiales de trabajo).

- Efectos de la práctica o del orden de presentación: control y medición

  En la Tabla 6.4 se puede comprobar que para evitar el efecto de la práctica con respecto al factor APPLICATION se controló su orden de presentación y, además se midió su influencia en la variable GROUP. Sin embargo, dicho efecto con respecto al factor DIAGRAM no se manipuló.

  No era factible que un sujeto en la réplica (segundo experimento) examinase los tres tipos de diagramas y, luego, en el siguiente (tercer experimento) estudiase combinaciones pares de diagramas de los ya vistos anteriormente. El principal motivo era la posible aparición del efecto de persistencia. Es decir, el rendimiento de un sujeto podría ser mejor en el tercer experimento, debido a que el efecto del aprendizaje de los tipos de diagramas empleados en la réplica podría persistir todavía en el siguiente estudio empírico. Para controlar dicho efecto fue necesario imponer una restricción al segundo experimento (réplica): los grupos tenían que estudiar el mismo tipo de diagrama en las tres sesiones.

Por ejemplo, los grupos G1 y G4 que utilizaron el diagrama de secuencia en todas las sesiones de la réplica, sólo pudieron examinar el par colaboración-estado en el tercer experimento (ver Tablas 6.4 y 7.4).

- Efectos de persistencia: control y medición

  Debido a la limitación impuesta a la réplica y al tercer experimento (ver Tabla 6.1), es posible que un efecto debido al aprendizaje del tipo de diagrama dinámico perdure en la réplica de una sesión a otra. Esta amenaza del efecto de persistencia (*carryover*) se midió bien en las covariables C1 y C2 [Mill92] o en la variable CARRY [Ratk93].

6.3.6.2. Validez de la conclusión

La validez de la conclusión o, lo que es lo mismo, la validez de la conclusión estadística se refiere a la seguridad de que hay una relación estadística entre el factor objeto de estudio y el resultado obtenido. Algunas de las amenazas que pueden afectar a la validez de la conclusión son, por ejemplo, la baja potencia estadística y el no cumplimiento de los supuestos de un contraste estadístico. Para evitar estas amenazas se deben estudiar cuidadosamente, entre otras, aquellas cuestiones relacionadas con los supuestos del test de hipótesis, con la elección de la potencia y también la del tamaño muestral. Dado que disponemos de datos de un estudio empírico anterior [Oter02], donde la validez de las conclusiones alcanzó una potencia del 80%, se examinan estas dos últimas cuestiones en el apartado 6.3.7, puesto que el diseño de la réplica no fue idéntico al diseño original.

6.3.7. Estimación del tamaño de la muestra de la réplica

Uno de los aspectos más importantes a la hora de diseñar un experimento es estimar el tamaño de la muestra necesario para alcanzar una potencia aceptable y controlar la probabilidad de cometer un error de tipo I. Para nuestro diseño split-plot SPF 36.3 no hemos encontrado las fórmulas que se puedan usar para calcular el tamaño muestral de su modelo ANOVA. Una de las posibilidades es seleccionar un modelo más conservador y sencillo para este diseño [Fost94]. En nuestro caso podría ser el modelo lineal correspondiente a un diseño split-plot SPF 3.3, en el que habríamos eliminado el factor GROUP de 6 niveles. De los tres factores objeto de estudio, el factor GROUP no es tan importante como los otros dos, pues su objetivo es la evaluación de los efectos de secuencia. Con esta eliminación suponemos que los sujetos se asignaron al azar al factor entre-sujetos DIAGRAM.

Concretamente para el modelo de un diseño split-plot SPF p.q (p = 3 niveles del factor entre-sujetos, q = 3 niveles del factor intra-sujetos) sólo encontramos en la bibliografía estadística fórmulas de determinación del tamaño muestral para el factor entre-sujetos DIAGRAM y el factor intra-sujetos APPLICATION ([Fost94] y [Kirk95, pp. 518-520]). Pero dado que en la investigación original la interacción de ambos tratamientos fue significativa [Oter02], necesitamos estimar el tamaño muestral no sólo para los factores principales, sino también para su interacción. Este último requisito sí se puede obtener utilizando el programa PASS 2002 (*Power Analysis and Sample Size*) del paquete estadístico NCSS [PASS02]. La salida obtenida de la aplicación de PASS con respecto al tiempo total (TSEC) se muestra en la Figura 6.2, que es un gráfico de potencia versus tamaño muestral por grupo para los dos factores (DIAGRAM y APPLICATION) y su interacción. Con respecto a la puntuación total (NRESP) dicha salida se presenta en la Tabla 6.5, donde se muestran los resultados numéricos para la potencia y el tamaño muestral para cada factor principal y su interacción.

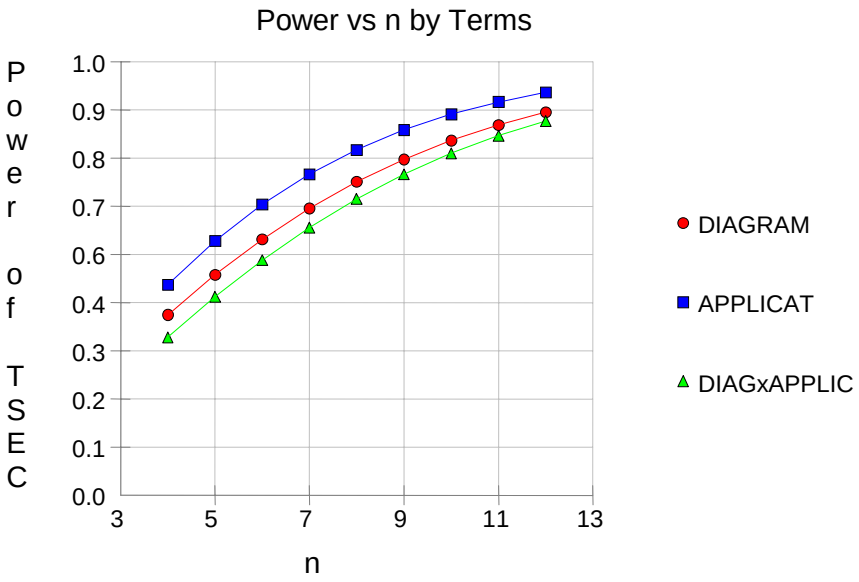


Figura 6.2. Gráfico de potencia versus tamaño muestral por grupo (n) para TSEC

Con el fin de estimar el tamaño muestral de este segundo experimento (réplica) ejecutamos el módulo de medidas repetidas del PASS, fijando un nivel de significación $\alpha = 0{,}1$ y una potencia $1 - \beta = 0{,}8$. Aunque la práctica habitual es poner $\alpha = 0{,}05$, la elección de dicho nivel es arbitraria. Como en el contexto de la ingeniería del software la mayoría de las investigaciones implican tamaños de efectos pequeños o medianos, fijamos un nivel de significación más relajado de $\alpha = 0{,}1$. Este nivel resulta apropiado en aquellas investigaciones

que incluyen pruebas estadísticas de los efectos de la interacción [Cohe88, p. 375], tal como sucede en este experimento. Además de estos parámetros, es necesario disponer de datos de estudios similares. En nuestro caso, especificamos las medias y las desviaciones típicas de una práctica empírica previa en las dos variables dependientes TSEC y NRESP, obtenidas a partir de las Tablas 5.9 y 5.10. Con respecto al tiempo total, el tamaño muestral debe ser como mínimo N = 30, n = 10 sujetos por grupo, tal como se puede comprobar en la Figura 6.2. Mientras que para la variable puntuación total, es suficiente con N = 27, n = 9 sujetos por grupo, tal como muestra la Tabla 6.5. El grupo se refiere al factor entre-sujetos DIAGRAM, y cada sujeto se mide 3 veces en el factor intra-sujetos APPLICATION.

Terms (Levels)	Power	n	N
DIAGRAM (3)	0.4909		
APPLICAT (3)	0.5630	4	12
DIAGxAPPLIC	0.4283		
DIAGRAM (3)	0.8142		
APPLICAT (3)	0.8697	8	24
DIAGxAPPLIC	0.7729		
DIAGRAM (3)	0.8596		
APPLICAT (3)	0.9064	9	27
DIAGxAPPLIC	0.8254		
DIAGRAM (3)	0.8949		
APPLICAT (3)	0.9335	10	30
DIAGxAPPLIC	0.8672		
DIAGRAM (3)	0.9220		
APPLICAT (3)	0.9531	11	33
DIAGxAPPLIC	0.9000		

Tabla 6.5. Tabla de potencia versus n para NRESP (α = 0,1 y SD = 1)

Así pues, con un tamaño muestral de 30 sujetos, la probabilidad de cometer un error de tipo I, llamado α, en las pruebas F es del 10% (es decir, aceptar la hipótesis nula cuando realmente es falsa). Y la probabilidad de cometer un error de tipo II, denominado β, es del 20% (o sea, rechazar la hipótesis nula cuando realmente es verdadera), siempre que el experimento se llevase a cabo utilizando un diseño split-plot 3.3. Como para nuestro estudio se planificó un diseño más complejo split-plot 36.3, el valor de α seguirá siendo igual a 0,1, pero β y 1 - β tendrán valores aproximados a 0,20 y a 0,80 respectivamente.

6.4. Ejecución de la réplica

Una vez planificado el experimento de forma rigurosa, la ejecución de la réplica se realiza en tres etapas:

- Formación

  Todos los sujetos asistieron a una etapa de formación, donde se impartieron varias clases de 1 hora y 30 minutos sobre el modelado estático y dinámico en UML.

- Entrenamiento

  En esta fase de entrenamiento, todos los alumnos contestaron primero a un cuestionario de 7 preguntas de carácter general sobre UML, sin tener en cuenta el tiempo transcurrido. Luego, se les repartió una hoja de instrucciones, un documento de diseño OO relativo al funcionamiento de una máquina de café, un listado de 3 cuestiones y un cronómetro digital. Todo este material se presenta en el apartado B.1.

  Esta etapa sirvió para familiarizar a los sujetos con los instrumentos y también para clasificar a dichos sujetos, evitando así que la asignación al azar de los mismos a las condiciones experimentales originase algún grupo elite. Esta amenaza fue descubierta en [Daly96], donde la mayoría de los sujetos de mejor rendimiento se asignó aleatoriamente a uno de los grupos, hecho que produjo que los resultados obtenidos estuviesen obviamente sesgados. Una forma de controlar dicha amenaza en nuestro experimento, debido al efecto de selección, fue crear grupos de sujetos de parecido rendimiento. El procedimiento que se practicó y los resultados que se obtuvieron para conseguir grupos emparejados se encuentra descrito en el apartado E.1.

- Experimental

  En la fase experimental, dividida en tres sesiones diferentes, a los estudiantes se les repartió un paquete conteniendo un documento de diseño en UML y un cuestionario, además de un cronómetro digital. Asimismo, para el buen desempeño de cada tarea experimental, se les informó oralmente (apartado C.1) y por escrito (apartado A.1) de los pasos que debían seguir. En cada sesión cada sujeto tuvo que contestar a las preguntas relativas a un documento de diseño, así como anotar el tiempo que tardó en responderlas en el cuadro correspondiente.

6.5. Análisis de los datos

Los datos recopilados primero deben ser codificados y luego analizados estadísticamente. El objeto es estudiar la influencia de los factores en las variables respuesta: tiempo total y puntuación global, métricas que miden la comprensión del modelado dinámico en los diseños UML. Para establecer dicha influencia se deben considerar una serie de cuestiones relacionadas con el análisis de los datos: nivel de significación y potencia (ya tratadas en el

apartado 6.3.7), configuración de los datos y modelo ANOVA a aplicar, cuestiones que se van a explicar en el apartado 6.5.1.

6.5.1. Estrategia del análisis de varianza

Con respecto a la configuración de los datos, como el factor APPLICATION tiene más de dos niveles, es de medidas repetidas y además cumple el supuesto de esfericidad de Mauchly (similitud de las correlaciones de las variables dependientes en los tres niveles de la variable intra-sujetos APPLICATION), entonces para la configuración y el análisis de los datos se puede adoptar el enfoque univariado ([JMP00, pp. 498-502], [NCSS00, pp. 2023-2025] y [SPSS97, pp. 149-151]). En la Tabla 6.6 la prueba de esfericidad no es significativa para TSEC (Sig. = 0,321 > α) ni para NRESP (Sig. = 0,367 > α) y, por lo tanto, no hay evidencia suficiente para rechazar el supuesto de esfericidad.

Factor intra-sujetos	Medida	W de Mauchly	Chi-cuadrado aprox.	GL	Significación
APPLICATION					
	TSEC	0,915	2,309	2	0,312
	NRESP	0,926	1,988	2	0,367

Tabla 6.6. Resultados del contraste sobre el supuesto de esfericidad

La otra cuestión se refiere al modelo ANOVA a aplicar. En el diseño original se manipuló tanto la secuencia del factor DIAGRAM como la del factor APPLICATION (ver Tabla 5.2). Así, el efecto debido al aprendizaje, o a la madurez, no estuvo presente en el modelo ANOVA. En cambio, en el diseño de la réplica sólo se controló la secuencia del factor APPLICATION (ver Tabla 6.4). Por ello en cada sesión a cada sujeto se le asignó un documento diferente, pero siempre el mismo tipo de diagrama. Esta forma de asignación puede contaminar los resultados por dos razones. La primera de ellas se debe al efecto de aprendizaje de DIAGRAM y la segunda a un efecto de persistencia (*carryover effect*) en el factor de medidas repetidas APPLICATION, efecto residual debido a que dicho aprendizaje pueda persistir de una sesión a otra. En consecuencia, ante la sospecha de que dichos efectos puedan estar presentes, nuestro diseño se estudia como un diseño *crossover* [Kueh00, pp. 525-530], cuyo análisis considera dos modelos ANOVA diferentes:

- Modelo reducido (sin efectos de aprendizaje y de carryover)

  Suponiendo que dichos efectos no son importantes [Wine91, p. 498] y que además las medidas repetidas en un sujeto están igualmente correlacionadas (igualdad de

covarianzas) [Kueh00, p. 499], nuestro diseño factorial de medidas repetidas se puede analizar como un diseño de parcelas divididas (*split-plot*). El factor entre-sujetos (DIAGRAM) en el diseño de medidas repetidas es equivalente al factor parcela completa (*whole plot*) en el diseño split-plot. Y la medida repetida en un sujeto, correspondiente al factor intra-sujetos (APPLICATION), equivale a la subparcela (*subplot*) en el diseño split-plot. En este caso el modelo ANOVA para la réplica sería:

$$Y_{ijkm} = \mu + A_j + C(A)_{m(j)} + D(C(A))_{i(m(j))} + B_k + (AB)_{jk} + \varepsilon_{ijkm}$$

(i = 1, ..., 30; j = 1, ..., 3; k = 1, ..., 3; m = 1, …, 6)

donde Y_{ijkm} es la respuesta observada (con respecto a una de las dos variables dependientes), μ es la media general, A_j es el efecto principal del tipo de diagrama, B_k es el efecto principal del tipo de dominio de aplicación, $(AB)_{jk}$ es el efecto correspondiente a la interacción DIAGRAM x APPLICATION, $C(A)_{m(j)}$ es el efecto asociado al factor grupo anidado dentro del diagrama, $D(C(A))_{i(m(j))}$ son los efectos asociados a los sujetos anidados dentro de los grupos y ε_{ijkm} es el error experimental. Se trata de un modelo mixto, puesto que los factores C y D, correspondientes a los grupos y a los sujetos respectivamente, son aleatorios y los otros dos fijos. Además está equilibrado, al tener el mismo número de observaciones en cada celda.

- Modelo completo (con efectos de aprendizaje y de carryover)

  Para descubrir la presencia de dichos efectos, el modelo completo ANOVA añade dos nuevos factores con respecto al modelo reducido. Uno de estos factores es ROUND, que se puede considerar como otra variable independiente (apartado 6.3.6.1) y que detecta si hay o no efecto de aprendizaje. Asimismo es posible codificar otro factor, denominado CARRY, tal como se indica en [Kueh00, p. 527], que contrasta si es o no significativo estadísticamente este efecto de carryover en la réplica.

6.5.2. Análisis de varianza completo

Lógicamente los datos recopilados de la réplica se han examinado primero considerando el modelo ANOVA completo. Para ello, dichos datos fueron codificados de la siguiente forma:

- El efecto debido al aprendizaje del tipo de diagrama se midió en la variable independiente ROUND, cuyos valores 1, 2 y 3 correspondieron al primer, segundo y tercer período respectivamente.

- Para la codificación del efecto de persistencia encontramos dos estrategias [Kueh00, pp. 545-546]:
  1. Milliken y Johnson [Mill92, pp. 440-450] incluyeron los efectos de carryover en el modelo como covariables: C1 y C2.

$$C1 \text{ en } ROUND(r) = \begin{cases} 1 & \text{si APPLICATION} = 1 \text{ en ROUND } (r-1) \\ -1 & \text{si APPLICATION} = 3 \text{ en ROUND } (r-1) \\ 0 & \text{en caso contrario} \end{cases}$$

$$C2 \text{ en } ROUND(r) = \begin{cases} 1 & \text{si APPLICATION} = 2 \text{ en ROUND } (r-1) \\ -1 & \text{si APPLICATION} = 3 \text{ en ROUND } (r-1) \\ 0 & \text{en caso contrario} \end{cases}$$

  2. Ratkowsky, Evans y Alldredge [Ratk93, pp. 440-450] introdujeron los efectos de carryover en el modelo como otro factor: CARRY.

$$CARRY = \begin{cases} 0 & \text{si ROUND} = 1 \text{ (no hay efectos de carryover)} \\ \text{APPLICATION en ROUND} = 1 & \text{si ROUND} = 2 \\ \text{APPLICATION en ROUND} = 2 & \text{si ROUND} = 3 \end{cases}$$

El análisis del modelo ANOVA completo se realizó empleando ambas alternativas. Las conclusiones obtenidas fueron las mismas tanto para el tiempo total (TSEC) como para la puntuación global (NRESP).

6.5.2.1. Variable dependiente: TSEC

En la Tabla 6.7 se muestran los resultados para el tiempo total según la primera estrategia. El efecto de aprendizaje (ROUND) no es significativo estadísticamente para la variable TSEC ($F = 1{,}124$; $p = 0{,}333 > \alpha$). Lo mismo sucede con los efectos de persistencia para esta variable dependiente, dada la no relevancia estadística de las covariables C1 ($F = 0{,}270$; $p = 0{,}605 > \alpha$) y C2 ($F = 0{,}201$; $p = 0{,}656 > \alpha$).

Factor		SC Tipo III	GL	MC	F	p
Intersección	Hipótesis	31583308,400	1	31583308,400	346,299	0,000
	Error	273607,526	3	91202,509		
DIAGRAM	Hipótesis	125449,319	2	62724,660	0,688	0,568
	Error	273607,526	3	91202,509		
GROUP(DIAGRAM)	Hipótesis	273607,526	3	91202,509	2,120	0,124
	Error	1032652,753	24	43027,198		
SUBJECT(GROUP (DIAGRAM))	Hipótesis	1032652,753	24	43027,198	2,318	0,006
	Error	927950,878	50	18559,018		
APPLICATION	Hipótesis	918529,807	2	459264,903	24,746	0,000

	Error	927950,878	50	18559,018		
DIAGRAM x APPLICATION	Hipótesis	30890,869	4	7722,717	0,416	0,796
	Error	927950,878	50	18559,018		
ROUND	Hipótesis	41733,359	2	20866,680	1,124	0,333
	Error	927950,878	50	18559,018		
C1	Hipótesis	5015,831	1	5015,831	0,270	0,605
	Error	927950,878	50	18559,018		
C2	Hipótesis	3737,723	1	3737,723	0,201	0,656
	Error	927959,878	50	18559,018		

Tabla 6.7. Resultados SPSS del ANOVA completo para la variable TSEC según [Mill92]

6.5.2.2. Variable dependiente: NRESP

En la Tabla 6.8 se muestran los resultados para la puntuación total según el segundo método. El efecto de aprendizaje (ROUND) no es estadísticamente significativo para la variable NRESP (F = 0,365; p = 0,696 > α). Lo mismo sucede con los efectos de persistencia para esta variable, dada la no relevancia del factor CARRY (F = 1,959; p = 0,152 > α).

Factor		SC Tipo III	GL	MC	F	p
Intersección	Hipótesis	1095,511	1	1095,511	1493,879	0,000
	Error	2,200	3	0,733		
DIAGRAM	Hipótesis	1,622	2	0,811	1,106	0,437
	Error	2,200	3	0,733		
GROUP(DIAGRAM)	Hipótesis	2,200	3	0,733	0,614	0,613
	Error	28,667	24	1,194		
SUBJECT(GROUP (DIAGRAM))	Hipótesis	28,667	24	1,194	1,247	0,250
	Error	47,882	50	0,958		
APPLICATION	Hipótesis	32,822	2	16,411	17,137	0,00
	Error	47,882	50	0,958		
DIAGRAM x APPLICATION	Hipótesis	8,844	4	2,211	2,309	0,071
	Error	47,882	50	0,958		
ROUND	Hipótesis	0,700	2	0,350	0,365	0,696
	Error	47,882	50	0,958		
CARRY	Hipótesis	3,751	2	1,876	1,959	0,152
	Error	47,882	50	0,958		

Tabla 6.8. Resultados SPSS del ANOVA completo para la variable NRESP según [Ratk93]

Otro detalle importante es que el término CARRY del segundo método, en lugar de tener tres grados de libertad, uno menos que sus cuatro niveles 0, 1, 2 y 3, sólo tiene dos. Esta circunstancia también se repite en el caso de las covariables C1 y C2 de la primera estrategia, donde sus grados de libertad suman dos. Esto es debido a que los efectos del primer período y los de carryover en el primer período están confundidos, es decir, no se pueden estimar independientemente.

6.5.3. Análisis de varianza reducido

La exploración de los datos ha demostrado que los efectos de aprendizaje y de persistencia no son estadísticamente significativos para TSEC ni para NRESP. Así pues, el contraste de las hipótesis nulas concretas planteadas en el apartado 6.3.2 se ha realizado mediante el análisis de varianza univariado split-plot para las dos variables respuesta (véase los ejemplos en [JMP00, pp. 271-277] y en [SPSS97, pp. 127-130]), fijado previamente un nivel $\alpha = 0,1$. Asimismo dichos datos se han analizado usando los paquetes estadísticos y de diseño experimental: JMP v.4 de SAS, NCSS 2000 y SPSS v.11, obteniéndose los mismos resultados.

6.5.3.1. Variable dependiente: TSEC

Teniendo en cuenta los resultados de la Tabla 6.9, el análisis revela que debemos aceptar las hipótesis nulas $H_{0\text{-DIAGxAPPLIC-tsec}}$ ($F = 0,4262$; Prob. $= 0,7890 > \alpha$), $H_{0\text{-DIAGRAM-tsec}}$ ($F = 0,6677$; Prob. $= 0,5677 > \alpha$) y $H_{0\text{-GROUP-tsec}}$ ($F = 2,1196$; Prob. $= 0,1242 > \alpha$).

Variable dependiente TSEC					
Factor		MC Den	GL Den		Síntesis MC Den
Diagram		91202,5	3		Group[diagram]
Application		18119,2	54		Residual
Diagram x Application		18119,2	54		Residual
Group[diagram]		43027,2	24		Subject[group,diagram]
Subject[group,diagram]		18119,2	54		Residual
Factor	SC	MC Num	GL Num	F Ratio	Prob > F
Diagram	125449,0	62724,7	2	0,6677	0,5677
Application	918530,0	459265,0	2	25,3468	0,0000
Diagram x Application	30890,9	7722,7	4	0,4262	0,7890
Group[diagram]	273608,0	91202,5	3	2,1196	0,1242
Subject[group,diagram]	1032653,0	43027,2	24	2,3747	0,0043

Tabla 6.9. Resultados del modelo ANOVA reducido con respecto a TSEC utilizando JMP

Es decir, la interacción DIAGRAM x APPLICATION, el tipo de diagrama y la secuencia del documento no influyen en el tiempo total que se tarda en comprender los diseños UML. Por el contrario, la prueba sobre el efecto del dominio de aplicación correspondiente a los diseños UML sí es estadísticamente significativa ($F = 25,3468$; Prob. $= 0,0000 < \alpha$), lo que implica que el efecto debido a la instrumentación ha estado presente. Pero este resultado es lógico dada la diferente complejidad y dominio de aplicación de los diseños, explicado en el apartado 6.3.4.1.

6.5.3.2. Variable dependiente: NRESP

Observando los resultados obtenidos de la Tabla 6.10, rechazamos la hipótesis nula $H_{0\text{-DIAGxAPPLIC-nresp}}$. La interacción entre el tipo de diagrama dinámico y el tipo de dominio de aplicación del documento de diseño OO influye en el número de respuestas correctas, pues el análisis estadístico revela que la interacción DIAGRAM x APPLICATION sí es significativa ($F = 2,28$; Prob. = 0,072357 < α) con una potencia de 0,746268. Es decir, la probabilidad de rechazar esta hipótesis siendo realmente falsa es del 74.6268%. Por otra parte, cuando una interacción es significativa, no se analizan los resultados de las hipótesis nulas: $H_{0\text{-DIAGRAM-nresp}}$ y $H_{0\text{-APPLICATION-nresp}}$, correspondientes a los efectos principales de las variables que participan en dicha interacción. Por último, aceptamos la hipótesis nula $H_{0\text{-GROUP-nresp}}$ ($F = 0,61$; Prob. = 0,612605 > α) sobre la no influencia de la secuencia del factor APPLICATION en la variable dependiente NRESP.

Análisis de Varianza: NRESP

Término Fuente	GL	Término Fijo?	Term Den	Media Cuadrática (MC)
A:diagram	2	Yes	B(A)	S+dsC+cdsB+bcdsA
B(A):group	3	No	C(AB)	S+dsC+cdsB
C(AB):subject	24	No	S(ABCD)	S+dsC
D:application	2	Yes	S(ABCD)	S+abcsD
AD:diagram x applicaction	4	Yes	S(ABCD)	S+bcsAD
S(ABCD)	54	No		

Término Fuente	GL	SC	MC	F Ratio	Nivel Prob	Potencia (α = 0,1)
A:diagram	2	1,6222	0,8111	1,11	0,436677	0,227355
B(A):group	3	2,2000	0,7333	0,61	0,612605	
C(AB):subject	24	28,6667	1,1944	1,23	0,257641	
D:application	2	32,8222	16,4111	16,93	0,000002	0,999887
AD:diagram x applicaction	4	8,8444	2,2111	2,28	0,072357	0,746268
S	54	52,3333	0,9691			

Tabla 6.10. Resultados del modelo ANOVA reducido con respecto a NRESP utilizando NCSS

6.6. Interpretación de los resultados

Para lograr una interpretación correcta de los resultados de la réplica es importante compararlos con los datos obtenidos en el experimento original. También hay que tener en cuenta que en el experimento original (diseño *latin square*) los sujetos se enfrentaron en cada sesión a un tipo de diagrama y a un tipo de documento de diseño siempre diferentes. De esta forma se evitó que hubiese un efecto de aprendizaje, tanto del tipo de diagrama como del tipo de diseño UML, de una sesión a otra. Estadísticamente se demostró que esta amenaza no estuvo presente a lo largo del experimento original, con respecto a ninguna de las dos

variables dependientes. En la réplica (diseño *split plot*) los sujetos participaron en tres sesiones, con igual diagrama pero con diferente documento de diseño. En el análisis de varianza de las Tablas 6.7 y 6.8 se puede comprobar que el efecto de aprendizaje, debido a repetir el mismo tipo de diagrama en cada sesión y medido en la variable ROUND, no fue significativo para el tiempo (TSEC) ni para la puntuación (NRESP).

6.6.1. Interpretación del tiempo total

En la Tabla 6.11 se muestran las medias y las desviaciones típicas del tiempo total (TSEC) para cada tipo de diagrama y para cada dominio de aplicación en el formato: $\overline{X}_{t\sec}(S_{t\sec})$. Estos tiempos totales corresponden tanto al diseño original (*latin square*) como al diseño de la réplica (*split plot*). Dicho resumen referido al estadístico de la media de TSEC también se presenta en la Figura 6.3 a) para el experimento original y en la Figura 6.3 b) para la réplica. En estos gráficos la longitud de las barras representa el valor medio del tiempo total en responder a las preguntas para cada tipo de diagrama y para cada dominio de aplicación.

	Diseño de cuadrado latino (*latin square*)			Diseño de parcelas divididas (*split plot*)		
	Secuencia	Colaboración	Estado	Secuencia	Colaboración	Estado
Teléfono Móvil	16:31 (05:39)	09:46 (03:55)	14:14 (07:26)	06:55 (02:06)	08:26 (03:13)	07:55 (01:55)
Sistema Biblioteca	15:56 (05:55)	22:02 (06:20)	14:27 (03:42)	09:26 (02:49)	09:55 (02:06)	10:31 (02:55)
Dictáfono Digital	17:34 (04:06)	16:29 (06:35)	22:11 (08:18)	10:39 (01:55)	12:07 (03:12)	12:52 (04:07)

Tabla 6.11. Resumen estadístico del tiempo total (mm:ss) para ambos experimentos

En un primer vistazo de los datos de la Tabla 6.11 y de los gráficos de la Figura 6.3 se puede apreciar que se invirtió más tiempo en comprender los documentos UML del diseño original que esos mismos documentos utilizados en la réplica.

a) b)

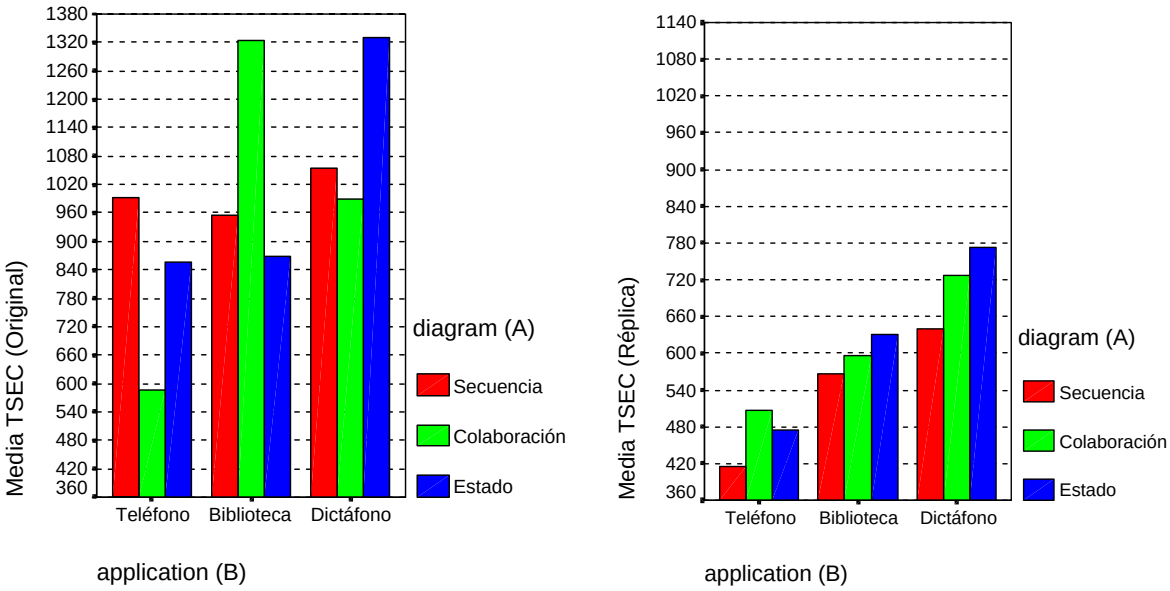


Figura 6.3. Diagramas de barras de la variable TSEC: a) experimento original b) réplica

Los contrastes estadísticos revelaron que la interacción DIAGRAM x APPLICATION era significativa en el experimento original, mientras que no lo era (significativa) en la réplica. Es decir, la comprensión del modelado dinámico en UML depende del tipo de diagrama y del dominio de aplicación de los diseños. Aunque esta conclusión no ha sido ratificada estadísticamente en el estudio empírico replicado, una observación más detallada de los gráficos de la Figura 6.3 nos indica que:

a) Teniendo en cuenta la complejidad de los escenarios de menor a mayor de la Tabla 6.3: Teléfono < Biblioteca < Dictáfono, en el gráfico de la Figura 6.3 b) se aprecia que el escenario especificado en un diseño UML a medida que es más complejo, requiere más tiempo de lectura para su interpretación. En el gráfico de la Figura 6.3 a) aquellos diseños que tienen los tiempos totales mayores también corresponden a los diseños con escenarios más complejos, pero además están modelados mediante un diagrama de colaboración (Biblioteca) y un diagrama de estado (Dictáfono).

b) La exploración del gráfico de la Figura 6.3 b) verifica la no relevancia de la interacción DIAGRAM x APPLICATION en la réplica, puesto que si se considera el tiempo de comprensión de menor a mayor, la relación Secuencia < Colaboración < Estado es independiente del dominio de aplicación y de la complejidad del escenario descrito en el diseño. Es decir, esta relación se verifica para los tres documentos de diseño, sean éstos de menor o mayor complejidad o de distinto dominio de aplicación.

c) De la relación anterior: Secuencia < Colaboración < Estado, que se observa en el gráfico de la Figura 6.3 b), se deduce que cuando el comportamiento dinámico se modela mediante un diagrama de secuencia, se comprende más rápidamente que si se trata de un diagrama de colaboración o de estado. Esta conclusión con respecto al diagrama de secuencia no es la misma en el gráfico de la Figura 6.3 a), donde el tiempo medio en responder se distribuye uniformemente a lo largo de los tres diseños, lo que implica que la comprensión del documento sea independiente del dominio de aplicación y de la complejidad del escenario.

d) Por lo tanto, con respecto a la métrica TSEC (tiempo total), la comprensión semántica del modelado dinámico:

- En la réplica es independiente del tipo de diagrama y del dominio de aplicación y complejidad de los diseños UML.
- En el experimento original esta conclusión sólo se puede aplicar al diagrama de secuencia. Precisamente los otros dos tipos de diagramas son los que originan que el tiempo de comprensión dependa del factor APPLICATION.

6.6.2. Interpretación de la puntuación total

En la Tabla 6.12 se muestran las medias y las desviaciones típicas de la puntuación total (NRESP) para cada tipo de diagrama y para cada dominio de aplicación en el formato: $\overline{X}_{nresp}(S_{nresp})$. Estas puntuaciones pertenecen tanto al diseño original (*latin square*) como al diseño de la réplica (*split plot*). Para una mejor interpretación de los resultados, se muestra gráficamente la puntuación media de respuestas correctas en la Figura 6.4 a) para el experimento original y en la Figura 6.4 b) para la réplica.

	Diseño de cuadrado latino (*latin square*)			Diseño de parcelas divididas (*split plot*)		
	Secuencia	Colaboración	Estado	Secuencia	Colaboración	Estado
Teléfono Móvil	4,5 (0,55)	4,5 (0,84)	3,33 (1,21)	4,6 (0,52)	3,9 (0,99)	3,80 (1,03)
Sistema Biblioteca	3 (1,26)	3,5 (1,38)	3,33 (1,37)	3,6 (1,17)	3,8 (1,32)	3,70 (0,95)
Dictáfono Digital	3,33 (1,21)	2,67 (1,63)	4,67 (0,52)	2,5 (1,08)	2,2 (1,03)	3,30 (0,82)

Tabla 6.12. Resumen estadístico de la puntuación total (5 puntos) para ambos experimentos

El análisis estadístico con respecto a la puntuación total (NRESP) reveló que la interacción DIAGRAM x APPLICATION era significativa, tanto en el diseño de cuadrado latino como en el diseño de parcelas divididas. Este resultado implica que es imposible separar el efecto del tipo de diagrama (DIAGRAM) del efecto del factor APPLICATION. En los gráficos de la Figura 6.4 se puede observar que, en general, a medida que el documento de diseño es más complejo, menor es el número de respuestas correctas, a excepción del diagrama de estado en el Dictáfono Digital.

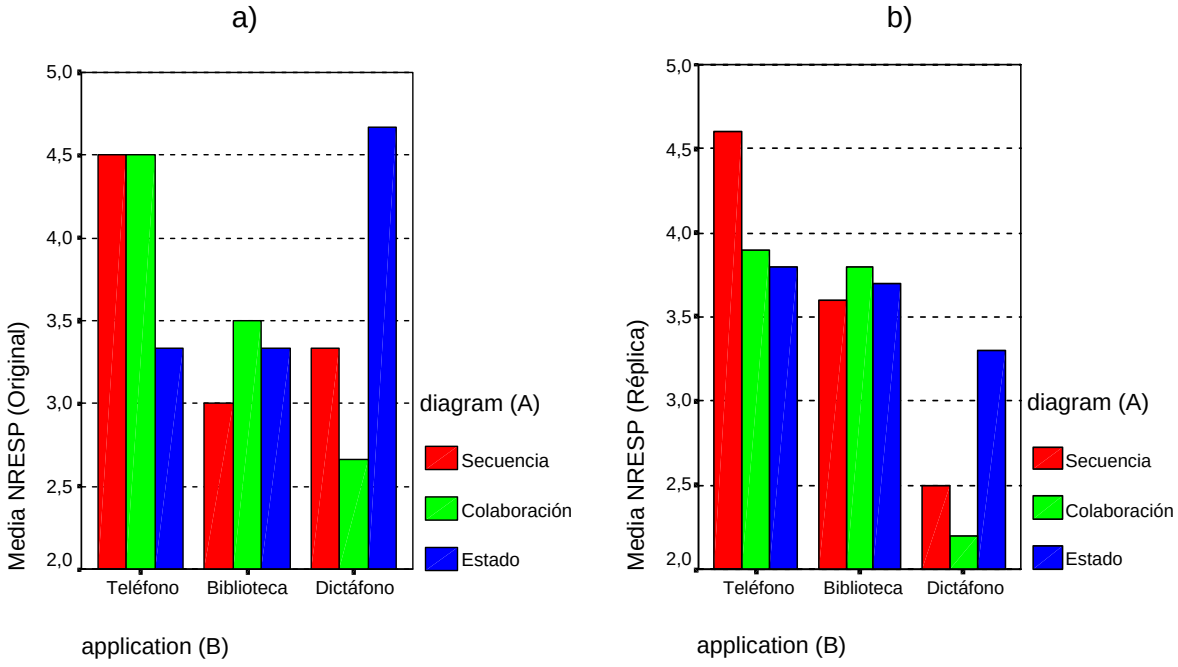


Figura 6.4. Diagramas de barras de la variable NRESP: a) experimento original b) réplica

Como la comprensión semántica del modelado dinámico en UML depende de los factores: DIAGRAM y APPLICATION, con respecto a la puntuación total, vamos a detallar las conclusiones para cada documento de diseño, considerando su dominio de aplicación y la complejidad (de menor a mayor) del escenario especificado.

a) Teléfono Móvil: sistema en tiempo real, cuyo escenario sólo incluye mensajes síncronos (uno de ellos con condición de repetición que anida tres mensajes), con el objetivo de describir el procedimiento de efectuar una llamada telefónica.

- En el experimento original no hay diferencias entre representar las interacciones entre objetos en el tiempo (diagrama de secuencia) o en el espacio (diagrama de colaboración). En la réplica este diseño es más fácil de interpretar semánticamente

cuando el diagrama dinámico utilizado es el de secuencia, pero no el de colaboración o el de estado.

- Lo que sí se comprueba es que en los dos experimentos el diagrama de secuencia ayuda a entender las especificaciones en tiempo real, conclusión que coincide con la primera consideración del apartado 3.3 ([OMG00, p. 91] y [OMG01, p. 100]). Pero se debería añadir que los diagramas de secuencia son mejores para comprender el modelado dinámico de especificaciones en tiempo real, siempre que los mensajes de los escenarios sean síncronos.
- Los resultados anteriores no apoyan empíricamente la segunda recomendación del capítulo 3 ([OMG00, p. 103] y [OMG01, p. 115]) acerca de que los diagramas de colaboración ayudan a entender mejor los procedimientos.
- En conclusión, los diagramas de secuencia mejoran la comprensión semántica del modelado dinámico en UML, cuando el dominio de aplicación es no reactivo en tiempo real y el escenario especifica sólo mensajes síncronos y no es complejo. En este caso representa un proceso secuencial con un único comportamiento repetitivo.

b) Sistema de Biblioteca: sistema de gestión de la información, cuyo escenario sólo incluye mensajes síncronos (uno de ellos con condición de bifurcación), con el objetivo de describir el procedimiento de pedir prestado un ejemplar de un libro o de una revista.

- Tanto en el experimento original como en la réplica la comprensión del comportamiento descrito en el diseño del Sistema de Biblioteca no depende del tipo de diagrama dinámico utilizado.
- Este resultado verifica, en parte, la quinta recomendación realizada por Fowler [Fowl00], que prefiere utilizar los diagramas de interacción (secuencia y colaboración) frente a los diagramas de estado y de actividad, cuando en el escenario no hay muchos comportamientos complejos.
- Sin embargo, la recomendación anterior se puede extender a los diagramas de estado. Gráficamente no hay diferencias significativas entre representar el escenario, que no es muy complejo, mediante un diagrama de interacción o varios de estado. Se indican varios de estado, pues en nuestro caso fueron necesarios dos diagramas de estado. El comportamiento especificado implicó que las instancias de dos clases cambiasen de estado.

- En conclusión, ninguno de los tres tipos de diagramas dinámicos aporta mayor seguridad en la interpretación de la información que transmiten los modelos UML, si el dominio de aplicación es de gestión y el escenario no es muy complejo.

c) Dictáfono Digital: sistema reactivo en tiempo real, cuyo escenario incluye mensajes síncronos y asíncronos (dos de ellos con condiciones de repetición y, a su vez, cada uno anida un mensaje iterativo), con el objetivo de describir el procedimiento de efectuar la reproducción de un mensaje que puede ser interrumpida.

- En el experimento original y en la réplica, es evidente que el mayor número de respuestas correctas se alcanza cuando el modelo dinámico es un diagrama de estado. Esta conclusión confirma la tercera recomendación del apartado 3.3 ([OMG00, p. 146] y [OMG01, p. 157]), que dice que es mejor utilizar los diagramas de estado en aquellos escenarios en que ocurren eventos asíncronos.
- Después del diagrama de estado, se observa gráficamente que la mayor puntuación se obtiene, en segundo lugar, con el diagrama de secuencia y, en tercer lugar, con el de colaboración. Por lo tanto, la cuarta recomendación del capítulo 3 [Rose99, p. 109], que indica que los diagramas de colaboración y de estado son más útiles en el diseño de sistemas en tiempo real, no se demuestra empíricamente.
- La conclusión general sobre la comprensión del modelado dinámico en UML cuando el dominio de aplicación es en tiempo real se resume en que:
  - Los diagramas de estado aportan una mayor estabilidad semántica cuando en los escenarios hay representados mensajes síncronos y algunos asíncronos. Además, se trata de escenarios complejos (4 repetitivas).
  - Los diagramas de secuencia aportan una mayor seguridad sobre la interpretación semántica de la información cuando en los escenarios sólo hay representados mensajes síncronos. Además, se trata de escenarios poco complejos (1 repetitiva).

6.7. Presentación

Como ya comentamos en el apartado 5.7, es importante que tanto los materiales como los datos recopilados estén disponibles. Toda esta información se puede encontrar:

- En este documento

  Material: Apéndice A y B.1

  Datos: Apéndice E
- En el CD dentro de la carpeta Réplica

- En la página WEB http://www.vc.ehu.es/jiwotvim/ecdmUML/ecdmUML.htm (en inglés).

Capítulo 6

CAPÍTULO 6 87

RÉPLICA DEL MODELADO DINÁMICO EN UML 87

6.1. INTRODUCCIÓN 87
6.2. DEFINICIÓN DE LA RÉPLICA Y DEL TERCER EXPERIMENTO 88
6.2.1. Hipótesis de alto nivel 88
6.3. PLANIFICACIÓN DE LA RÉPLICA 88
6.3.1. Variables experimentales 89
6.3.2. Hipótesis nulas concretas derivadas de la Hipótesis nula 1 89
6.3.3. Sujetos experimentales 90

Capítulo 7

Evaluación complementaria del modelado dinámico en UML

7.1. Introducción

Con el fin de planificar correctamente esta tercera práctica experimental, en primer lugar se examinaron los resultados estadísticos del primer experimento, y sucintamente los obtenidos en la réplica. Tras este análisis, observamos que para un sistema reactivo en tiempo real, la comprensión de la especificación dinámica en un diseño UML fue mejor cuando el escenario se implementó mediante un diagrama de estado. Por lo tanto, en el tercer estudio empírico no se incluyó como material a investigar un documento de diseño correspondiente a este dominio de aplicación.

En los apartados 7.2 a 7.7 se detallan todas las fases del proceso experimental para ejecutar esta tercera práctica empírica sobre el modelado dinámico.

7.2. Definición del tercer experimento

7.2.1. Hipótesis nula 2 de alto nivel

Antes de empezar a planificar el tercer experimento, vamos a recordar cuál era la Hipótesis nula 2 de alto nivel formulada en el apartado 6.2.1. Esta hipótesis trata de investigar si los modelos dinámicos de los diseños UML expresados mediante combinaciones de pares de diagramas (colaboración-estado, secuencia-estado o secuencia-colaboración) difieren en cuanto al número de respuestas correctas y al tiempo total en contestar a todas las cuestiones.

7.3. Planificación del tercer experimento

Para responder a esta Hipótesis nula 2 se planificó un tercer estudio empírico, en el que participaron los mismos sujetos del segundo experimento (réplica). Esta limitación de recursos humanos implicó la necesidad de elaborar nuevo material experimental, pero teniendo en cuenta los resultados obtenidos tras una primera exploración estadística de los datos de la réplica.

7.3.1. Variables experimentales

Este tercer experimento investiga dos variables independientes o factores:

- DIAGRAM_PAR: Es la combinación par de diagramas dinámicos UML. Se estudiaron: colaboración-estado, secuencia-estado y secuencia-colaboración.
- SOFT_APPLIC: Es el tipo de dominio correspondiente a la aplicación o sistema software. Se utilizaron dos diseños diferentes en su dominio y en su complejidad: Sistema de Gestión de Seguros (SGS) y Sistema de Control de Parking (SCP).

En cada sesión se midieron dos variables dependientes o variables respuesta: TSEC y NRESP, ya definidas anteriormente en el apartado 5.3.1.

7.3.2. Hipótesis nulas concretas derivadas de la Hipótesis nula 2

Al igual que en el apartado 6.3.2, la Hipótesis nula 2 de alto nivel se refina en las hipótesis nulas concretas de la Tabla 7.1.

	TSEC	NRESP
DIAGRAM_PAR x SOFT_APPLIC No hay diferencias entre las 3x2 condiciones experimentales con respecto a ...	$H_{0\text{-DIAG\_PARxSOFT\_APPLIC-tsec}}$	$H_{0\text{-DIAG\_PARxSOFT\_APPLIC-nresp}}$
DIAGRAM_PAR No hay diferencias entre los sujetos que utilizan las tres combinaciones de pares de diagramas dinámicos UML con respecto a ...	$H_{0\text{-DIAGRAM\_PAR-tsec}}$	$H_{0\text{-DIAGRAM\_PAR-nresp}}$
SOFT_APPLIC No hay diferencias entre los sujetos que leen los dos documentos de diseño UML con respecto a ...	$H_{0\text{-SOFT\_APPLIC-tsec}}$	$H_{0\text{-SOFT\_APPLIC-nresp}}$

Tabla 7.1. Hipótesis nulas concretas del tercer experimento

7.3.3. Sujetos experimentales

En este tercer estudio empírico participaron los 30 alumnos de la réplica. Sin embargo, tres de ellos no realizaron las tareas experimentales en los mismos días que el resto, lo que implicó que sus datos no fuesen válidos (ver apartado 7.3.6.1).

7.3.4. Material experimental

Para la confección de este nuevo material se tuvieron en cuenta los resultados del primer experimento y también los de la réplica en un primer análisis exploratorio de sus datos. Si es un sistema reactivo en tiempo real y el escenario es complejo, el diagrama más adecuado para comprender el documento UML es el de estado. Si es un sistema no reactivo en tiempo real y el escenario es poco complejo, la comprensión del diseño es más segura cuando el escenario se implementa mediante un diagrama de secuencia. Si se trata de una aplicación de gestión, no hay un modelo dinámico que haga mas fácil de interpretar la información del documento. En estos dos últimos dominios de aplicación, ¿cómo sería esta comprensión si los escenarios fuesen más complejos? Así pues, elegimos dos sistemas software de diferentes dominios: una aplicación de gestión de información y otra de control. Con respecto a la complejidad de los escenarios, decidimos que los diseños UML de estos sistemas especificasen todos los comportamientos dinámicos que aparecen en el diagrama de casos de uso, pero sin incluir eventos asíncronos.

En definitiva, el nuevo material elaborado para esta práctica empírica complementaria se detalla en el Apéndice B:

- Dos documentos de diseño en notación UML.

  Los diseños que empleamos en este estudio experimental describen el funcionamiento de dos aplicaciones: un Sistema de Control de Parking (SCP) [Trav00] y un Sistema de Gestión de Seguros (SGS) [Lagu00]. Cada documento contiene los modelos estáticos y dinámicos para cada sistema. La parte estática viene representada por un diagrama de paquetes y un diagrama de clases. El comportamiento dinámico se especifica mediante un diagrama de casos de uso, varios diagramas de secuencia (uno por cada caso de uso), varios de colaboración (uno por cada caso de uso) y varios diagramas de estado (uno por cada clase que cambia de estado). Todos los mensajes que aparecen en los escenarios son síncronos.

  - La aplicación Parking, o SCP, controla y supervisa las entradas y salidas al/del aparcamiento dependiendo del número de plazas libres disponibles. Su diseño consta

de 15 páginas, una para cada diagrama. Entre los diagramas figuran cuatro de secuencia, cuatro de colaboración y tres de estado. En la Tabla 7.2 se indica el tamaño de este diseño contabilizando para cada clase [Chid94]: acoplamiento entre objetos (*Coupling Between Objects: CBO*), profundidad en el árbol de herencia (*Depth of Inheritance Tree: DIT*), número de hijos (*Number Of Children: NOC*) y métodos ponderados por clase (*Weighted Methods per Class: WMC*).

Sistema PARKING	Billete	Billete Mensual	Billete Sin Reserva	Garaje Parking	Lectora Billete	Máquina Expendedora Billete	Puerta
CBO	1	1	1	4	1	1	1
DIT	0	1	1	0	0	0	0
NOC	2	0	0	0	0	0	0
WMC	3	2	3	6	1	2	2

Tabla 7.2. Métricas de tamaño correspondientes al diseño de PARKING

- La aplicación Seguros, o SGS, gestiona las pólizas de seguros y proporciona presupuestos a los clientes que lo solicitan. Su diseño consta de 19 páginas, en cada página viene descrito un diagrama. Los diagramas dinámicos que se incluyen son seis de secuencia, seis de colaboración y tres de estado. En la Tabla 7.3 se muestra el resultado de aplicar las métricas de diseño OO de [Chid94] a cada una de las clases.

Sistema SEGUROS	Póliza	Recibo	Recibo Provisional	Cliente	Presupuesto
CBO	3	1	1	2	2
DIT	0	0	1	0	0
NOC	0	1	0	0	0
WMC	8	4	0	3	4

Tabla 7.3. Métricas de tamaño correspondientes al diseño de SEGUROS

Teniendo en cuenta el número total de páginas y los datos de las Tablas 7.2 y 7.3, los dos diseños UML son de diferente complejidad. Podríamos suponer que la lectura del documento de la aplicación de Seguros llevará más tiempo que la del Parking. Las razones son dos: el diseño de Seguros tiene más diagramas de interacción que el del Parking; y además los escenarios de Seguros incluyen mensajes simples, con condiciones de bifurcación y de iteración, mientras que en Parking los mensajes son simples y sólo de bifurcación. En consecuencia, el diseño de Seguros es más complejo que el diseño de Parking.

- Dos cuestionarios.

  El formulario correspondiente a cada aplicación consta de 10 preguntas y cada pregunta de 4 posibles respuestas, siendo solamente una de ellas la correcta.

7.3.5. Diseño experimental

Teniendo en cuenta que el número de sujetos participantes fue 30 y que el número de tratamientos es 6 (3 niveles del factor A: DIAGRAM_PAR x 2 niveles del factor B: SOFT_APPLIC), se podía haber planteado un diseño factorial aleatorio, donde se obtendrían 30/6 = 5 resultados para cada condición experimental. En general, la randomización es efectiva cuando el tamaño de grupo es grande [Wine91, p. 504]. Como 5 no es lo suficientemente grande, optamos por un diseño factorial de parcelas divididas 3x2 con medidas repetidas en uno de los factores, en concreto en el factor B [Wine91, pp. 509-531], tal como muestra la Tabla 7.4. Este diseño también se denomina diseño factorial split-plot SPF 3.2 [Kirk95, pp. 515-524].

	Sesión I	Sesión II
Colaboración-Estado (G1 y G4)	Seguros	Parking
Secuencia-Estado (G2 y G5)	Seguros	Parking
Secuencia-Colaboración (G3 y G6)	Seguros	Parking

Tabla 7.4. Diseño factorial split-plot 3.2

Debido a la restricción descrita en la Tabla 6.1, no se pudo manipular el orden de presentación del factor DIAGRAM en el segundo experimento (réplica) ni tampoco se puede manipular el del factor DIAGRAM_PAR en este tercer experimento. Este hecho implica la posibilidad de que se produzca un efecto de aprendizaje debido a esta variable (DIAGRAM en la réplica o DIAGRAM_PAR en este tercer experimento) de una sesión a otra.

En la réplica se controló el efecto de aprendizaje debido a DIAGRAM. Para controlar esta amenaza se manipuló la secuencia del otro factor APPLICATION, es decir, se manipuló el orden de presentación de los tres diseños UML. Esta manipulación permitió no sólo controlar, sino también medir el efecto de dos amenazas a la validez interna del segundo experimento (réplica). El factor GROUP se utilizó para comprobar si el efecto debido a la secuencia en que se presentaron los documentos estaba presente en el estudio empírico de la réplica. Por otra

parte, se midió el efecto de aprendizaje en la variable independiente ROUND, no confundida con el factor APPLICATION.

Sin embargo, en este tercer estudio empírico aplicamos los controles inversos a los utilizados en el anterior, en la réplica. No se controló el efecto de aprendizaje debido a DIAGRAM_PAR, sino que se controló el efecto debido al plagio. Para evitar la posibilidad de que los sujetos compartiesen información sobre las aplicaciones de software se utilizó un orden fijo de presentación del factor SOFT_APPLIC, tal como muestra la Tabla 7.4. Este orden fijo repercute en que la variable GROUP, existente en el estudio de la réplica, no aparezca en este diseño split-plot. En consecuencia, la variable SUBJECT está anidada en los tres niveles del factor DIAGRAM_PAR, pues los grupos de sujetos (bloques) coinciden o se confunden con los niveles del factor DIAGRAM_PAR, tal como se explicó en el apartado 4.5.2.2.

7.3.6. Validez experimental

Considerando los cuatros tipos de validez experimental expuestos en el apartado 6.3.6, las amenazas a la validez constructiva y a la validez externa coinciden con las ya comentadas en el primer experimento. Por otra parte, al no disponer de datos previos relativos a este tercer experimento, para la validez de la conclusión no podemos estimar el tamaño muestral, pero sí podemos elegir un nivel de significación $\alpha = 0{,}1$ y una potencia $1 - \beta = 0{,}8$. Entonces, en el apartado 7.3.6.1 sólo vamos a abordar aquellas amenazas a la validez interna, cuyo control haya sido diferente al empleado en la réplica.

7.3.6.1. Validez interna

Para minimizar las amenazas a la validez interna de este tercer experimento se han manipulado los factores extraños de la siguiente forma:

- Efectos de instrumentación: medición

  Con respecto a los dos diseños UML, éstos fueron manipulados por los investigadores en cuanto a su complejidad y al dominio de aplicación. Por ello es uno de los factores a estudiar en este experimento, variable independiente SOFT_APPLIC. Pero con respecto a la variación de los dos test de 10 preguntas, se pudo evitar haciendo que todos los sujetos contestasen a ambos cuestionarios.
- Efectos de mortalidad: control

Aunque en un principio recopilamos datos correspondientes a 30 sujetos, algunos no se incluyeron en el análisis estadístico puesto que todos ellos no soportaron las mismas condiciones experimentales. En concreto, sólo 27 alumnos realizaron las tareas experimentales en dos días diferentes con una semana de intervalo entre ellas, mientras que los 3 restantes las hicieron en un mismo día y transcurrido un mes con respecto a los otros sujetos. Esta diferencia en las condiciones implicó que estos datos no se considerasen fiables. En concreto, el tiempo en que los 3 sujetos contestaron a los dos cuestionarios es el mismo que la mayoría de los 27 tardaron en responder a uno en un único día, lo que repercutió en que el número de respuestas correctas no fuese superior a 2.

- Efectos de aprendizaje: confusión

  Cada sujeto utilizó la misma combinación par de diagramas dinámicos UML a lo largo de todo el experimento y el orden de presentación de las aplicaciones software fue fija. En este caso, cualquier efecto de aprendizaje estaría directamente relacionado con la variable tratamiento DIAGRAM_PAR (es decir, implicaría una mayor habilidad con la combinación par de diagramas dinámicos), pero este aprendizaje sería un aprendizaje uniforme, por igual en todos los sujetos. Debido a esta confusión, el factor SOFT_APPLIC no se refiere sólo a la aplicación sino que también está enmascarando un efecto de aprendizaje, lo que implica que en el análisis estadístico este factor sea la suma de dos (patrón de confusión: SOFT_APPLIC + ROUND).

7.4. Ejecución del tercer experimento

Evidentemente la ejecución de este tercer estudio sólamente comprende la fase experimental, que se llevó a cabo en condiciones de examen.

Todos los sujetos participaron en dos sesiones, entre las que transcurrieron 7 días. En cada una de ellas cada alumno realizó la tarea experimental de contestar a las 10 preguntas del cuestionario correspondiente bien al diseño de Seguros o al diseño de Parking. Pero antes de empezar a responder se les recordó las normas que debían seguir, tanto de forma oral como escrita.

Como en la primera sesión únicamente se utilizó el cuestionario y el diseño de Seguros, los participantes sólo pudieron compartir información sobre esta aplicación. De esta forma nos aseguramos que la amenaza debido al plagio no estuviese presente en la segunda sesión de este experimento.

7.5. Análisis de los datos

7.5.1. Estrategia del análisis de varianza

En el tercer experimento a cada sujeto sólo se le tomaron dos medidas repetidas en el factor SOFT_APPLIC, una por cada nivel. Por lo tanto, no es necesario realizar ningún contraste sobre el supuesto de esfericidad de la matriz de varianzas-covarianzas, en el caso de que se quiera utilizar el enfoque univariado para la configuración y/o el análisis de los datos.

El Modelo Lineal General (*GLM: General Lineal Model*) para este tercer experimento, descrito en el apartado 4.5.2.2, es el siguiente ([Wine91, p. 510] y [Kueh00, p. 473]):

$$Y_{ijk} = \mu + A_j + C(A)_{i(j)} + B_k + (AB)_{jk} + \varepsilon_{ijk}$$

(i = 1, ..., 27; j = 1, ..., 3; k = 1, 2)

donde Y_{ijk} es la respuesta observada (con respecto a una de las dos variables dependientes), μ es la media general, A_j es el efecto principal de la combinación par de diagramas, B_k es el efecto principal del dominio de aplicación (que además incluye el efecto de aprendizaje), $(AB)_{jk}$ es el efecto correspondiente a la interacción DIAGRAM_PAR x SOFT_APPLIC, $C(A)_{i(j)}$ es el efecto asociado al factor correspondiente a los sujetos anidado dentro de la combinación par de diagramas dinámicos y ε_{ijk} es el error experimental.

El contraste de las hipótesis nulas concretas planteadas en el apartado 7.3.2 se ha realizado mediante el análisis de varianza de medidas repetidas para las dos variables dependientes, fijado un nivel $\alpha = 0{,}1$. Dado que el número de medidas repetidas es dos para cada sujeto, no es necesario verificar el supuesto de esfericidad. Por lo tanto, en los contrastes estadísticos tanto univariados como multivariados se consiguen los mismos datos. Los resultados que se detallan en la Tabla 7.5 corresponden al análisis de varianza univariado realizado con el paquete SPSS, pero que también coinciden con la salida obtenida por JMP y NCSS.

7.5.2. Análisis de varianza para TSEC y NRESP

Observando los datos de la Tabla 7.5, el análisis de varianza revela la no influencia de la interacción DIAGRAM_PAR x SOFT_APPLIC en la comprensión de los diseños UML que se mide en las variables respuesta: TSEC (Tiempo total) y NRESP (Puntuación global). Esta no influencia se debe a que se aceptan las hipótesis nulas $H_{0\text{-DIAG\_PARxSOFT\_APPLIC-tsec}}$ ($F = 1{,}186$; $p = 0{,}323 > \alpha$) y $H_{0\text{-DIAG\_PARxSOFT\_APPLIC-nresp}}$ ($F = 0{,}080$; $p = 0{,}924 > \alpha$). Asimismo, dado que los

dos documentos de diseño utilizados en el tercer experimento son de diferente complejidad, es lógico que los contrastes sobre las hipótesis nulas del factor SOFT_APPLIC sean significativos $H_{0\text{-SOFT\_APPLIC-tsec}}$ ($F = 84{,}629$; $p = 0{,}000 < \alpha$) y $H_{0\text{-SOFT\_APPLIC-nresp}}$ ($F = 21{,}071$; $p = 0{,}000 < \alpha$). Además, este factor no sólo tiene en cuenta el tipo de aplicación, sino que también mide un efecto de aprendizaje (SOFT_APPLIC + ROUND). No es posible separar estos dos efectos en dos variables independientes.

Factor	Medida	GL	MC	F	p	TE	Potencia
Entre-sujetos							
DIAGRAM_PAR	TSEC	2	45002,446	0,252	0,779	0,021	0,154
	Error	24	178380,424				
	NRESP	2	8,907	4,156	0,028	0,257	0,792
	Error	24	2,144				
Intra-sujetos							
SOFT_APPLIC + ROUND	TSEC	1	2312268,929	84,629	0,000	0,779	1,000
	Error	24	27322,422				
	NRESP	1	34,241	21,071	0,000	0,468	0,998
	Error	24	1,625				
DIAGRAM_PAR x SOFT_APPLIC	TSEC	2	32398,979	1,186	0,323	0,090	0,352
	Error	24	27322,422				
	NRESP	2	0,130	0,080	0,924	0,007	0,117
	Error	24	1,625				

Tabla 7.5. Resultados SPSS para las variables TSEC y NRESP

El único resultado estadístico a destacar de este tercer experimento es que la combinación par de diagramas dinámicos UML, representado en el factor DIAGRAM_PAR, sí es estadísticamente significativo en la variable NRESP $H_{0\text{-DIAGRAM\_PAR-nresp}}$ ($F = 4{,}156$; $p = 0{,}028 < \alpha$), con una potencia ($1 - \beta$) del 0,792 (79,20%). Es decir, si este experimento se repitiese 100 veces, en el aproximadamente 80 de esas veces se obtendría la misma conclusión: rechazar la hipótesis nula $H_{0\text{-DIAGRAM\_PAR-nresp}}$. El error de tipo II ($\beta$) que se comete es del 0,198 (~ 20%). Sin embargo, con respecto a la otra variable dependiente TSEC, el contraste estadístico no es significativo y por ello aceptamos la hipótesis $H_{0\text{-DIAGRAM\_PAR-tsec}}$ ($F = 0{,}252$; $p = 0{,}779 > \alpha$).

7.6. Interpretación de los resultados

En la Figura 7.1 se representan dos gráficos de perfil correspondientes a las variables del tiempo total (TSEC) y a la puntuación global (NRESP). Estos dos gráficos de perfil son gráficos de línea de las medias marginales estimadas de TSEC y NRESP a través de los niveles de cada factor. Mientras que un gráfico de medias observadas muestra el efecto de

TSEC o NRESP y el error, un gráfico de medias marginales estimadas muestra sólo el efecto de la variable dependiente, sin incluir el error. En ambos gráficos de la Figura 7.1 las líneas son paralelas, lo que indica que no hay interacción entre los factores estudiados con respecto a TSEC y a NRESP. Este resultado gráfico confirma el resultado del análisis estadístico de aceptación de las hipótesis referidas a la interacción DIAGRAM_PAR x SOFT_APPLIC de la Tabla 7.5.

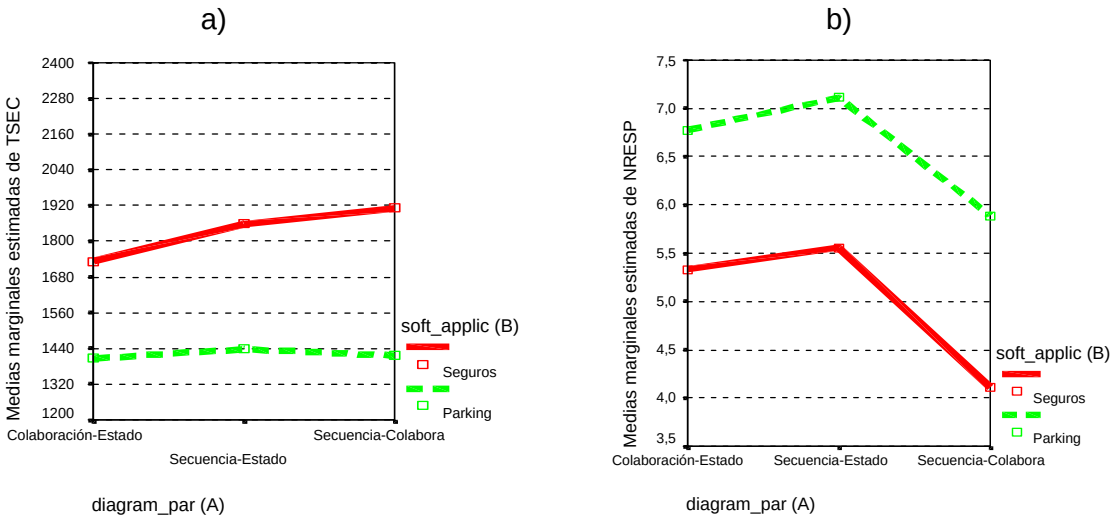


Figura 7.1. Gráficos de perfil correspondientes al tiempo (TSEC) y a la puntuación (NRESP)

Al observar los dos gráficos de perfil se aprecia que en la comprensión semántica del documento de Seguros, utilizado en la primera sesión, se consumió más tiempo y se obtuvieron menos respuestas correctas que en comprender el diseño del Parking, usado en la segunda sesión. En un principio, el hecho de que en la segunda sesión se invierta menos tiempo y se logren más respuestas se debe a que el diseño de la aplicación Parking es más sencillo que el de Seguros. Sin embargo, también influye el aprendizaje de la misma combinación par de diagramas de una sesión a otra, lo que viene a confirmar que el factor relativo al tipo de aplicación esconde un efecto de aprendizaje (SOFT_APPLIC + ROUND).

En el gráfico de perfil de la Figura 7.1 a) con respecto al tiempo total, en la primera sesión (aplicación de gestión de Seguros) se observa que hay diferencias entre las tres combinaciones de pares de diagramas. Estas diferencias no resultaron significativas estadísticamente en un análisis post-hoc. La comprensión semántica del comportamiento dinámico en el diseño de una aplicación de gestión necesita más tiempo, si el modelado incorpora la combinación secuencia-colaboración y requiere menos tiempo si dicha

composición es colaboración-estado. Por el contrario, en la segunda sesión (aplicación de control de Parking) el tiempo total invertido en la comprensión semántica de la parte dinámica de este diseño UML es similar en las tres combinaciones de diagramas, lo que demuestra que ha habido un efecto de aprendizaje de una sesión a otra (ya explicado en el apartado 7.3.6.1). Por lo tanto, es lógico la aceptación de la hipótesis $H_{0\text{-DIAGRAM\_PAR-tsec}}$.

En el gráfico de perfil de la Figura 7.1 b) con respecto a la puntuación global, tanto en la primera sesión como en la segunda se aprecian diferencias gráficas entre las tres combinaciones de pares de diagramas. Estas diferencias se confirmaron estadísticamente en un análisis a posteriori, resultando significativos los contrastes del número de respuestas en la combinación secuencia-colaboración con respecto a las otras dos combinaciones de pares de diagramas. Asimismo se detecta un aprendizaje de la segunda sesión (aplicación de control de Parking) con respecto a la primera sesión (aplicación de gestión de Seguros), pues hay un mayor número de aciertos en las respuestas. Pero este aumento debido a un efecto de aprendizaje se mantiene (es proporcional) para los tres pares de combinaciones tanto en el primer día como en el segundo. Independientemente de que el dominio de aplicación sea de gestión o de control, se logra una mayor estabilidad semántica del comportamiento dinámico si los modelos especificados en los diseños UML es el par secuencia-estado o el par colaboración-estado (en segundo lugar), con la condición de que los escenarios sólo incluyan mensajes síncronos. En cambio, se consigue una interpretación menos segura de la información cuando el diseño de la parte dinámica se representa mediante el par secuencia-colaboración. En consecuencia, es lógico el rechazo de la hipótesis $H_{0\text{-DIAGRAM\_PAR-nresp}}$.

7.7. Presentación

Como ya comentamos en el apartado 6.7, es importante que tanto los materiales como los datos recopilados estén disponibles. Toda esta información se puede encontrar:

- En este documento

  Material: Apéndice B.2 y B.3

  Datos: Apéndice F
- En el CD dentro de la carpeta Experimento3
- En la página WEB http://www.vc.ehu.es/jiwotvim/ecdmUML/ecdmUML.htm (en inglés).

Capítulo 8

OML y UML

8.1. Introducción a OML

Aparte del Proceso Unificado (RUP) de desarrollo de software, surge otra metodología denominada OPEN (*Object-oriented Process, Environment and Notation*). Ambos son enfoques orientados a objetos de tercera generación que incluyen lenguajes estándares de modelado. Aunque hay similitudes, algunos estudios comparativos tales como [Barb00] y [Hend99a] exponen las diferencias existentes entre los dos métodos. Sin embargo, existen pocos estudios empíricos que evalúen dichas diferencias en el modelado de aplicaciones software. Por ejemplo, en [Zend01] se comparan experimentalmente los modelos de estructura estática (los diagramas de clases) de UML y OPEN con respecto al concepto OO de granulado grueso. En cambio, este capítulo se centra en evaluar de forma empírica los diferentes modelos para la especificación del comportamiento dinámico existentes tanto en UML como en OPEN, pero no con respecto a un concepto OO sino referente a la comprensión de dichos modelos.

8.1.1. Diagramas de modelado en los lenguajes OML y UML

La metodología de desarrollo OPEN proporciona un lenguaje de modelado, denominado OML, y un proceso de desarrollo [Hend98]. A su vez el lenguaje OML se compone de una notación COMN (*Common Object Modelling Language*) y un metamodelo [Fire98a]. Por el contrario, UML es sólo un lenguaje de modelado sin estructura de proceso/ciclo de desarrollo, aunque posteriormente Rational haya definido su particular Proceso Unificado [Jaco00]. También UML consta de una notación UML y de un metamodelo [Rumb00]. Las características de ambos lenguajes, así como los aspectos de los tipos de diagramas que fueron objeto de investigación, se encuentran explicados en el Capítulo 1.

La notación COMN, al igual que UML, ofrece un conjunto de tipos de diagramas que se utilizan para modelar sistemas software. Unos diagramas documentan la parte estática, otros especifican el comportamiento dinámico de una aplicación. En general, hay cuatro tipos principales de diagramas tanto en COMN como en UML: redes semánticas, diagramas de clases escenario, diagramas de interacción y diagramas de transición de estado (ver Tabla 8.1). Aparte, UML incluye diagramas de implementación (vistas como un tipo de red semántica en COMN) y diagramas de actividad (no se utilizan en la notación COMN).

Notación UML	Notación COMN
Diagramas de estructura estática • Diagramas de clases • Diagramas de paquetes	Redes semánticas • Diagramas de contexto • Diagramas de configuración • Diagramas de paquetes • Diagramas de herencia • Diagrama de capas • Diagramas de despliegue
Diagramas de casos de uso	Diagramas de clases escenario
Diagramas de interacción • Diagramas de secuencia • Diagramas de colaboración	Diagramas de interacción • Diagramas de secuencia de caja negra • Diagramas de secuencia de caja blanca • Diagramas de colaboración de paquete • Diagramas de colaboración interna
Diagramas de estado	Diagramas de transición de estado
Diagramas de actividad	(no se usan diagramas de actividad)
Diagramas de implementación	(tipo de red semántica)

Tabla 8.1. Comparación de tipos principales de diagramas (UML y COMN)

Teniendo en cuenta que nuestra investigación se centra en la comprensión del modelado dinámico en UML y en COMN, la correspondencia entre los diagramas dinámicos de UML y los de OML que vamos a estudiar se establece en la Tabla 8.2.

Lenguaje UML	Lenguaje OML
Diagrama de secuencia Diagrama de colaboración	Diagrama de secuencia de caja blanca
Diagrama de estado	Diagrama de colaboración interna Diagrama de transición de estado

Tabla 8.2. Equivalencia entre los modelos dinámicos de UML y OML a estudiar

La utilidad de los dos diagramas de interacción de UML se corresponde sólo con el diagrama de secuencia de caja blanca de OML. Por otra parte, el diagrama de estado de UML tiene su equivalente en el diagrama de transición de estado de OML. También el diagrama de

colaboración interna de OML se ajusta al diagrama de estado de UML, pues es un diagrama especializado en documentar la colaboración de las operaciones contenidas dentro de un objeto (ver apartado 1.2.2.3.3).

Para comparar la influencia que el tipo de lenguaje puede tener o no en la comprensión de los modelos que especifican el comportamiento dinámico de una aplicación decidimos llevar a cabo un cuarto estudio empírico. En los apartados 8.2 a 8.7 se detallan las principales actividades de este experimento: definición del objetivo, planificación, ejecución, análisis, interpretación y presentación tanto de los materiales como de los resultados experimentales.

8.2. Definición del objetivo del cuarto experimento

8.2.1. Objetivo

Tal como se sugiere en [Wohl00], el primer paso es definir formalmente nuestro objetivo. Por ello, si aplicamos la plantilla del paradigma GQM a este cuarto experimento nuestro objetivo queda definido así:

"Analizar *los modelos dinámicos de los diseños en lenguaje UML y en OML*
para *evaluar* su *comprensión semántica* desde el punto de vista del
diseñador de aplicaciones software en el contexto de *un experimento*
de laboratorio con estudiantes de Informática."

8.3. Planificación del cuarto experimento

8.3.1. Hipótesis nula de alto nivel

Una vez especificado el objetivo de este experimento, planteamos la siguiente hipótesis nula de alto nivel:

$H_{0\text{-NOTATION}}$: El lenguaje de modelado orientado a objetos (UML versus OML) no influye en la comprensión de los modelos dinámicos que se encuentran especificados en los documentos de diseño.
$H_{A\text{-NOTATION}}$: Sí influye.

Asimismo, teniendo en cuenta las equivalencias entre los modelos dinámicos de la Tabla 8.2, pensamos en establecer hipótesis nulas más específicas, relacionadas con el tipo de

diagrama dinámico en las notaciones de ambos lenguajes. Por ello, planteamos contrastar estadísticamente si hay o no diferencias significativas entre:

$H_{0\text{-STATE}}$: UML-Estado versus OML-Colaboración Interna y OML-Transición de Estado.
$H_{0\text{-SEQUENCE}}$: OML-Secuencia Caja Blanca versus UML-Secuencia y UML-Colaboración.

8.3.2. Variables experimentales

Considerando tan solo la primera hipótesis nula $H_{0\text{-NOTATION}}$ en el experimento se manipularon dos variables independientes:

- NOTATION: Es el tipo de notación estándar empleado en los diseños OO. Se estudiaron las dos notaciones correspondientes a los lenguajes UML y OML.
- GROUP: Es la secuencia o el orden de presentación de la notación. Como el factor NOTATION tiene 2 niveles, se obtienen 2! = 2 secuencias diferentes. Si al grupo A le corresponde la secuencia UML-OML, al grupo B la secuencia OML-UML (ver Figura 8.1). De esta forma se pudo detectar y medir la presencia del efecto debido a la práctica.

Para medir la comprensión en cada sesión se calcularon dos variables dependientes: TSEC y NRESP, que ya se describieron en el apartado 5.3.1 del primer experimento.

8.3.3. Sujetos experimentales

Es evidente que la situación ideal sería experimentar con profesionales. Pero hay otras circunstancias que respaldan la utilización de estudiantes como sujetos experimentales. En este caso, los profesionales podrían estar más familiarizados con la notación UML que con la notación COMN, debido a su mayor divulgación en el mercado como lenguaje de modelado OO. Es decir, podrían tener más conocimientos de UML que de COMN y, en consecuencia, la comprensión del lenguaje OML estaría en desventaja.

Considerando las razones anteriores, los sujetos que participaron en el experimento fueron 30 alumnos del segundo ciclo de Ingeniería Informática de la Universidad del País Vasco. Previa a la fase experimental, los estudiantes asistieron a una etapa de formación, donde se impartieron clases sobre el modelado estático y dinámico tanto en UML como en OML. Por lo tanto, su conocimiento previo con respecto a los dos lenguajes era equiparable.

8.3.4. Material experimental

Para este experimento se utilizó nuevo material empírico, que se encuentra desarrollado en el Apéndice C:

- Dos documentos de diseño.

  Los dos diseños OO describen la misma aplicación sobre el funcionamiento de una máquina expendedora de bebidas [Fire98b], uno especificado en UML y el otro en OML. La documentación elaborada para los dos períodos fue igual con respecto a la estructura estática, pero diferente en cuanto al modelado dinámico. En el primer período se emplearon diagramas de estado (UML versus OML) desde la perspectiva del responsable de mantenimiento, mientras que en el segundo período se utilizaron diagramas de interacción (UML versus OML) desde la perspectiva del cliente, tal como se muestra en la Figura 8.1.
- Dos cuestionarios de preguntas y respuestas de elección múltiple.

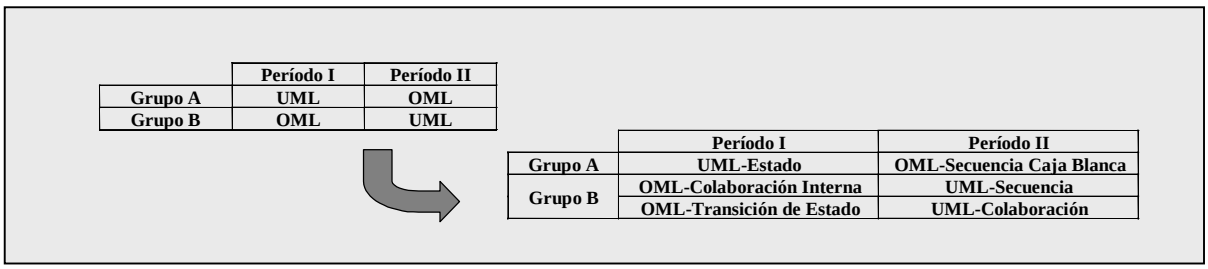
  En cada período se empleó un formulario diferente, compuesto de 5 preguntas con 1 única respuesta correcta.
- Cronómetros digitales.

  Cada sujeto dispuso de un cronómetro digital para medir el tiempo invertido en cada respuesta.

8.3.5. Diseño experimental

Como el factor, objeto de estudio, NOTATION tiene 2 niveles, se pensó en un diseño de dos grupos aleatorios, donde 15 sujetos fuesen asignados al azar a UML y los 15 restantes a OML. Además, para contrastar las dos hipótesis nulas $H_{0\text{-STATE}}$ y $H_{0\text{-SEQUENCE}}$ era necesario dividir cada grupo en tres subgrupos, uno por cada tipo de diagrama dinámico a investigar dentro de cada notación. Dado que disponer de tan solo 5 sujetos para evaluar cada condición experimental específica nos parecía escaso, pensamos en un diseño de medidas repetidas donde cada sujeto fuese medido 2 veces. Aunque ahora disponemos de 10 valores para aceptar o rechazar las hipótesis específicas, en un diseño intra-sujeto aparecen otros efectos distorsionantes. Uno de ellos es el efecto debido a la práctica. Para controlarlo, en nuestro diseño aplicamos la técnica del reequilibrado (*counterbalancing*) a la variable experimental NOTATION. Con esta estrategia, el efecto de la práctica también se puede medir, pues se puede considerar como otro factor en nuestro experimento: GROUP.

Teniendo en cuenta todas estas particularidades, planificamos un diseño de dos-tratamientos, dos-secuencias y dos-períodos, muy utilizado en el campo de la farmacología (por ejemplo, para evaluar el efecto de un nuevo fármaco versus un placebo). Según la terminología de [Kirk95, pp. 349-355] se designa como diseño factorial CO-2 por CrossOver de 2 tratamientos, que se explica en el apartado 4.5.1.1. Asimismo se puede asimilar como un diseño de cuadrado latino 2x2 o cross-over 2x2, pues las 2 filas y las 2 columnas de un cuadrado latino corresponden respectivamente a las 2 secuencias y a los 2 períodos de un cross-over [Kueh00, pp. 536-540]. Aplicando la técnica del reequilibrado al factor NOTATION, la secuencia UML-OML se asigna al grupo A, mientras que el grupo B, recibe la secuencia opuesta OML-UML, tal como muestra la tabla de la izquierda de la Figura 8.1. De esta forma se controló el efecto del orden de presentación o secuencia, y además se evitó que hubiese un efecto de aprendizaje del tipo de lenguaje de una sesión a otra.



	Período I	Período II
Grupo A	UML	OML
Grupo B	OML	UML

	Período I	Período II
Grupo A	UML-Estado	OML-Secuencia Caja Blanca
Grupo B	OML-Colaboración Interna	UML-Secuencia
	OML-Transición de Estado	UML-Colaboración

Figura 8.1. Diseño factorial cross-over 2x2

Por otra parte, el plan experimental de nuestro diseño cross-over 2x2 no sólo estuvo arreglado de acuerdo al tipo de notación, sino que también se adaptó al tipo de diagrama dinámico de UML y de OML. En esta adaptación tuvimos en cuenta las equivalencias entre los diagramas de ambas notaciones y las hipótesis nulas concretas $H_{0\text{-STATE}}$ y $H_{0\text{-SEQUENCE}}$. Por ello en el primer período se compararon el diagrama de estado de UML y sus correspondientes de OML, mientras que en el segundo período se evaluaron el diagrama de secuencia de caja blanca de OML y sus correspondientes de UML (ver la tabla de la derecha de Figura 8.1).

Con esta planificación no sólo logramos dar respuesta a todas las hipótesis nulas del apartado 8.3.1, sino que también conseguimos controlar el efecto de aprendizaje debido tanto al tipo de lenguaje como al tipo de diagrama dinámico. Para ello 10 estudiantes se asignaron al Grupo A y 20 al Grupo B, 10 a cada par de tratamientos a evaluar de la secuencia OML-UML. Así cada sujeto se midió dos veces, una en cada tipo de notación y en un tipo diferente de modelo dinámico.

8.3.6. Validez experimental

Para asegurar la fiabilidad de los resultados, en todo estudio empírico debemos controlar los factores que amenazan la validez del experimento. A continuación, explicamos cómo han sido minimizadas estas amenazas.

8.3.6.1. Validez interna

Mediante nuestro diseño hemos controlado las amenazas a la validez interna de la siguiente forma:

- Efectos de selección: control mediante la técnica de aleatorización

  Los estudiantes se asignaron al azar bien al grupo A, bien al grupo B. Además, el diseño fue de medidas repetidas, lo que redujo la amenaza debida a las diferencias individuales.
- Efectos de instrumentación: medición

  Los efectos de instrumentación no constituyeron amenaza alguna, pues la única variación en el material experimental se debió a la variable independiente NOTATION manipulada por nosotros. Se usaron dos versiones del diseño de un proyecto: una en notación UML y otra en notación COMN.
- Efectos de madurez: control y medición

  Los efectos de madurez debidos al cansancio y al aprendizaje de los sujetos según avanzaba el experimento se controlaron y se midieron en el factor PERIOD (también denominado ROUND). En cada período los sujetos realizaron tareas experimentales diferentes en cuanto al tipo de notación y al tipo de diagrama dinámico, tal como muestra la Figura 8.1. Esta decisión en el diseño evitó que hubiese un efecto de aprendizaje del tipo de lenguaje de una sesión a otra. De esta forma el factor NOTATION no estaba enmascarando un efecto de madurez. Es decir, PERIOD no dependía del factor NOTATION (Período 1 no estaba sólo limitado a la versión UML) y se pudo tratar como otro factor.
- Efectos de la práctica o del orden de presentación: control y medición

  Para evitar los efectos de la práctica o de la secuencia con respecto al factor NOTATION se controló su orden de presentación. La estrategia utilizada fue el *counterbalancing*, que consistió en repetir los dos tratamientos experimentales, UML y OML, de tal forma que primero se presentaron en un orden y, después, en el inverso. Por ello fue posible medir su influencia en el factor GROUP.

- Efectos de persistencia: confusión

  Se controlaron los efectos de persistencia o de *carryover*, debidos a que los efectos de aplicar un tratamiento en el período 1 podrían continuar en el tratamiento del período 2, planificando entre ambas sesiones un intervalo de varios días. Aún así, este efecto residual se tuvo en cuenta en el posterior análisis de los datos, aunque confundido con alguno de los otros factores, tal como se explica en el apartado 8.5.

8.3.6.2. Validez externa

Las amenazas a la validez externa condicionan que los resultados de un estudio empírico se puedan generalizar a un entorno más amplio bajo condiciones diferentes. En nuestro caso hemos identificado dos amenazas. La primera es la selección de los sujetos, ya tratada en el apartado 8.3.3. La segunda objeción está relacionada con el tipo de aplicación al que pertenecen las versiones de diseño empleadas. Al tratarse de un sistema empotrado, las conclusiones de este estudio están restringidas a este dominio de aplicación. Así pues, para lograr que las conclusiones sean ampliables, es necesario replicar este experimento bajo otras condiciones.

8.4. Ejecución del cuarto experimento

Las tres etapas fundamentales que comprende la ejecución de una práctica empírica son:

- Formación

  En primer lugar, los sujetos asistieron a una etapa de formación para instruirse en los conceptos sobre modelado en UML y en OML.

- Entrenamiento

  Luego, en la etapa de entrenamiento tuvieron que familiarizarse con los instrumentos y las tareas experimentales. En esta fase se utilizó el diseño de un Termostato Digital [Fire98b] en las dos versiones y un test. Al final del test se incluía además una tarea diferente: realizar la verificación de la traza de los mensajes, que aparecían en un diagrama de interacción, en los diagramas de clases, y también anotar el tiempo. Por ello se les explicó cómo debían realizar dicha verificación según se tratase de un diagrama de secuencia o de colaboración en UML, o de un diagrama de secuencia de caja blanca en OML.

- Experimental

La fase experimental se dividió en dos períodos, entre los que transcurrieron 6 días para evitar efectos de cansancio y de aprendizaje. Pensando en los posibles interrogantes que los sujetos pudiesen plantear, se les explicó oralmente algunas condiciones que están redactadas en el apartado C.1 y, también se les entregó una hoja de instrucciones (ver apartado A.1) para que realizasen correctamente su trabajo experimental.

En el primer período se les repartió un paquete que contenía un documento de diseño, un cuestionario y un cronómetro. Su tarea experimental consistió en responder a la preguntas del test a la vista del tipo de diagramas y del tipo de notación que aparecían en el diseño. Cada estudiante tuvo que anotar no sólo la respuesta sino también el tiempo (en min:seg:cseg) que tardó en contestar cada pregunta.

En el segundo período se les distribuyó el mismo material que en el primero. En esta sesión tuvieron que desempeñar dos tareas experimentales. La primera fue idéntica a la anterior. Sin embargo, la segunda tarea consistió en verificar la secuencia de interacciones entre objetos y clases implicados en dos escenarios diferentes. Para ello tuvieron que enumerar dichas interacciones, por ejemplo, de un diagrama de secuencia y, luego identificarlas a lo largo de los cinco paquetes, en concreto, a lo largo de aquellos paquetes que contenían las clases que implementaban dichas interacciones. Los alumnos numeraron cada traza con un color diferente.

8.5. Análisis de los datos

8.5.1. Estrategia del análisis de varianza

El principal objetivo es averiguar en qué medida el tipo de notación estándar influye en las dos variables dependientes, que en este estudio empírico explican la comprensión de los diseños OO. Para establecer dicha influencia diseñamos un experimento crossover 2x2. En este tipo de diseño no se debe ignorar la amenaza de un efecto residual o de carryover [Kueh00]. Por ello, a la hora de contrastar la hipótesis nula $H_{0\text{-NOTATION}}$ debemos considerar el modelo lineal completo de un crossover 2x2 [Kueh00, pp. 536-540]:

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_k + \tau_t + \lambda_c + \varepsilon_{ijk}$$

(i, k, t, c = 1, 2; j = 1, 2, ... , 30)

donde Y_{ijk} es la respuesta observada, μ es la media general, α_i es el efecto de la práctica o de la secuencia (GROUP), β_j es el efecto random asociado al sujeto (SUBJECT), γ_k es el

efecto de la madurez o del período (PERIOD), τ_t es el efecto directo del factor tratamiento (NOTATION), λ_c es el efecto de carryover y ε_{ijk} es el error experimental.

Aunque lo lógico sería que se aplicara el modelo completo, ello no es posible, pues los efectos de carryover en un crossover 2x2 están confundidos con otros efectos. Es decir, no se puede separar la estimación de los efectos de carryover sin implicar a los efectos de los otros factores. El motivo es que en la formulación matemática de este modelo hay que estimar cinco parámetros a partir de cuatro ecuaciones. Por ello se debe eliminar un parámetro del modelo completo, lo que representa considerar modelos alternativos para el análisis de un crossover 2x2, tales como [Mill92, pp. 435-440], [Kirk95, pp. 349-355], [Armi97, pp. 231-235], cuya terminología se refleja en la Tabla 8.3.

Referencia	Parámetros del modelo		
[Mill92]	Tratamiento	Tratamiento x Secuencia	Secuencia
[Kirk95]	Período x Secuencia	Período	Secuencia
[Armi97]	Tratamiento	Período	Tratamiento x Período

Tabla 8.3. Equivalencias en la terminología de tres modelos para el diseño crossover 2x2

En la Tabla 8.3 se puede comprobar que los efectos principales están confundidos con los efectos de interacción de los otros dos factores. En un experimento crossover, al no tratarse de un factorial completo, no se pueden estimar todas las interacciones de los factores. Por ello, en nuestro diseño PERIOD y NOTATION x GROUP están confundidos y se refieren al efecto de madurez (aprendizaje). Del mismo modo el efecto de secuencia representado en el factor GROUP tiene como alias NOTATION x PERIOD, y el factor NOTATION como alias PERIOD x GROUP [Dall00]. Por otra parte, en estos modelos el parámetro que se ha eliminado es el correspondiente al del efecto de carryover. Sin embargo, como un efecto de carryover se produce cuando el efecto de un tratamiento en el período 1 persiste en el tratamiento del período 2, equivale a que la interacción Tratamiento x Período sea significativa. Por lo tanto, NOTATION x PERIOD no sólo es un alias para el efecto de secuencia (GROUP), sino que también se refiere al efecto de carryover. Es decir, el efecto de persistencia está confundido con el efecto de la práctica.

En definitiva, vamos a adoptar el modelo de [Kirk95] que nos permite calcular el valor de Y_{ijk} para un sujeto o bloque *i*, en el grupo *j*, y en el período *k*, especificado en el apartado 4.5.2.2:

$$Y_{ijk} = \mu + \alpha_j + \beta_k + \pi_i + \alpha\beta_{jk} + \varepsilon_{ijk}$$

(i = 1, …, 30; j = A, B; k = 1, 2)

donde $\alpha\beta_{jk}$ es la interacción Período x Secuencia que está confundida y se corresponde con el factor tratamiento NOTATION, tal como se puede comprobar en la Tabla 8.5.

Para contrastar las hipótesis nulas de este modelo, que incluye la hipótesis $H_{0\text{-NOTATION}}$, fijamos un nivel de $\alpha = 0{,}1$. Asimismo aplicamos la técnica del análisis de varianza, utilizando el paquete SPSS v.11. Estos resultados se confirmaron usando el estadístico de contraste *t* para diseños crossover ahora disponible en el NCSS [NCSS00].

8.5.2. Análisis de varianza para contrastar la hipótesis $H_{0\text{-NOTATION}}$

En la Tabla 8.4 se plantean las hipótesis nulas que se derivan a partir del modelo según el enfoque [Kirk95], teniendo en cuenta las métricas: tiempo total (TSEC) y puntuación obtenida (NRESP).

	TSEC	NRESP
NOTATION No hay diferencias entre las medias de los lenguajes UML y OML con respecto a ...	$H_{0\text{-NOTATION-tsec}}$	$H_{0\text{-NOTATION-nresp}}$
PERIOD No hay diferencias entre las medias del Período 1 y el Período 2 con respecto a ...	$H_{0\text{-PERIOD-tsec}}$	$H_{0\text{-PERIOD-nresp}}$
GROUP No hay diferencias entre las medias de la secuencia UML-OML y la secuencia OML-UML con respecto a ...	$H_{0\text{-GROUP-tsec}}$	$H_{0\text{-GROUP-nresp}}$

Tabla 8.4. Hipótesis nulas concretas del cuarto experimento

Aplicando a los datos recogidos el análisis de varianza de medidas repetidas y utilizando el estadístico de contraste F, se obtuvo el informe estadístico que se resume en la Tabla 8.5.

Según los resultados de la Tabla 8.5 se aceptan las hipótesis nulas referidas a la amenaza de un efecto de aprendizaje del tipo de notación de una sesión a otra: $H_{0\text{-PERIOD-tsec}}$ ($F = 2{,}855$; $p = 0{,}102 > \alpha$) y $H_{0\text{-PERIOD-nresp}}$ ($F = 0{,}056$, $p = 0{,}814 > \alpha$). Asimismo el efecto de secuencia, medido en el factor GROUP y referido también al efecto de carryover, no estuvo presente en el experimento, pues las hipótesis no han sido significativas estadísticamente para ninguna de las variables dependientes: $H_{0\text{-GROUP-tsec}}$ ($F = 0{,}008$; $p = 0{,}931 > \alpha$) y $H_{0\text{-GROUP-nresp}}$ ($F = 0{,}405$; $p = 0{,}530 > \alpha$). Por lo tanto, si los efectos de secuencia, de período y de carryover no son relevantes, nuestro diseño crossover es apropiado para lograr el objetivo planteado en el apartado 8.2.1 [Jone89].

Factor	Métrica	GL	MC	F	p
Entre-sujetos					
GROUP (Secuencia)	TSEC	1	1473,432	0,008	0,931
	Error	28	0,424		
	NRESP	1	194043,125	0,405	0,530
	Error	28	1,667		
Intra-sujetos					
PERIOD	TSEC	1	3275,854	2,855	0,102
	Error (Period)	28	6741,190		
	NRESP	1	7,5E-02	0,056	0,814
	Error (Period)	28	1,331		
PERIOD x GROUP (NOTATION)	TSEC	1	400050,2	63,389	0,000
	Error (Period)	28	6741,190		
	NRESP	1	16,875	12,676	0,001
	Error (Period)	28	1,331		

Tabla 8.5. Análisis de varianza para las variables TSEC y NRESP

Por último, las hipótesis nulas relacionadas con el efecto directo del tipo de notación se rechazan con respecto a las dos variables respuesta: $H_{0\text{-NOTATION-tsec}}$ ($F = 63,389$; $p = 0,000 < \alpha$) y $H_{0\text{-NOTATION-nresp}}$ ($F = 12,676$; $p = 0,001 < \alpha$). Este rechazo implica la aceptación de la hipótesis alternativa $H_{A\text{-NOTATION}}$. Así pues, la conclusión es que el tipo de lenguaje sí influye en la comprensión semántica de los diseños.

8.5.3. Análisis de varianza para contrastar la hipótesis $H_{0\text{-STATE}}$

Teniendo en cuenta sólo los datos del Período 1, podemos contrastar la hipótesis nula referida al tipo de diagramas que se utilizan en UML y en COMN para modelar los estados y las acciones a lo largo del tiempo de vida de una instancia de una clase. Para verificar esta hipótesis con respecto al tiempo total: $H_{0\text{-STATE-tsec}}$ y a la puntuación obtenida: $H_{0\text{-STATE-nresp}}$, se utilizó el análisis de varianza entre-sujetos y el estadístico F.

Los resultados del análisis revelan el rechazo de la hipótesis $H_{0\text{-STATE-tsec}}$ ($F = 14,288$; $p = 0,000 < \alpha$) y la aceptación de la otra hipótesis $H_{0\text{-STATE-nresp}}$ ($F = 2,342$; $p = 0,115 > \alpha$). Como el factor tipo de diagrama tiene tres niveles, cuando se encuentra un efecto significativo quiere decir que hay diferencia entre ellos pero no se indica entre cuáles. Para saberlo hay que realizar contrastes personalizados. Por ello se ejecutaron comparaciones múltiples por parejas para determinar qué medias difieren solamente con respecto al tiempo (ver Tabla 8.6). Un asterisco indica que las medias de grupo son estadísticamente significativas a un nivel de significación $\alpha = 0,1$.

(I) Tipo de diagrama	(J) Tipo de diagrama	(I-J)	Error	p

UML-Estado	OML-Colaboración Interna	0:10:12*	0:02:51	0,005
UML-Estado	OML-Transición de Estado	0:14:58*	0:02:51	0,000
OML-Colaboración Interna	OML-Transición de Estado	0:04:45	0:02:51	0,267

Tabla 8.6. Comparaciones múltiples para contrastar la hipótesis $H_{0\text{-STATE}}$

En la Tabla 8.6 se observan diferencias significativas entre el diagrama de estado de UML y cualquiera de los diagramas de OML, pero no entre los dos diagramas propios de OML. De aquí se deduce que estas diferencias en la variable tiempo son a nivel de notación. Por lo tanto, si la modelización de los estados de una instancia se realiza utilizando la notación COMN, su comprensión semántica es más rápida que usando la notación UML.

8.5.4. Análisis de varianza para contrastar la hipótesis $H_{0\text{-SEQUENCE}}$

Considerando sólo los datos del Período 2, podemos contrastar la hipótesis nula referida al tipo de diagramas que se utilizan en UML y en COMN para especificar los escenarios correspondientes a un caso de uso. Puesto que se definieron dos métricas, estas hipótesis para el tiempo y para la puntuación se denotan $H_{0\text{-SEQUENCE-tsec}}$ y $H_{0\text{-SEQUENCE-nresp}}$ respectivamente.

Los resultados del análisis de varianza, una vez aplicado a los datos del Período 2, revelaron que el tipo de diagrama en que se especifica un caso de uso son estadísticamente significativos con respecto al tiempo y al número de respuestas correctas. Por ello, las hipótesis nulas $H_{0\text{-SEQUENCE-tsec}}$ ($F = 11{,}274$; $p = 0{,}000 < \alpha$) y $H_{0\text{-SEQUENCE-nresp}}$ ($F = 4{,}231$; $p = 0{,}025 < \alpha$) son rechazadas. Una vez que se ha determinado que existen diferencias entre las medias, realizamos comparaciones múltiples post-hoc para contrastar qué parejas de medias son las que difieren con respecto a las dos variables dependientes (ver Tabla 8.7).

Métrica	(I) Tipo de diagrama	(J) Tipo de diagrama	(I-J)	Error	p
TSEC	OML-Secuencia Caja Blanca	UML-Secuencia	-0:10:04*	0:02:58	0,002
	OML-Secuencia Caja Blanca	UML-Colaboración	-0:12:24*	0:02:58	0,001
	UML-Secuencia	UML-Colaboración	0:00:20	0:02:58	0,993
NRESP	OML-Secuencia Caja Blanca	UML-Secuencia	1,20*	0,55	0,098
	OML-Secuencia Caja Blanca	UML-Colaboración	1,50*	0,55	0,036
	UML-Secuencia	UML-Colaboración	0,30	0,55	0,860

Tabla 8.7. Comparaciones múltiples para contrastar la hipótesis $H_{0\text{-SEQUENCE}}$

En la Tabla 8.7 se aprecian diferencias significativas entre el diagrama de secuencia de caja blanca de OML y cualquiera de los diagramas de UML, pero no entre los dos diagramas de interacción de UML. Estas diferencias se refieren tanto al tiempo como a la puntuación. Es decir, las diferencias son a nivel de notación no a nivel de diagrama. Así pues, las

interacciones entre objetos en un escenario son más rápidas y más fáciles de interpretar semánticamente, cuando el diagrama dinámico está implementado en notación COMN.

8.6. Interpretación de los resultados

A la hora de interpretar los resultados, es importante combinar el análisis con la visualización de los datos. Una forma de describir un conjunto de datos es utilizando un gráfico de caja y bigotes (*box and whisker plot*). En general, la interpretación de este tipo de gráfico es la siguiente. La amplitud de la caja recoge el 50% de la muestra, desde el cuartil inferior (25%) hasta el cuartil superior (75%). La línea que atraviesa la caja representa a la mediana.

La Figura 8.2 muestra dos gráficos de caja con respecto a la variable tiempo total (TSEC). El gráfico de la izquierda corresponde al Período 1, mientras que el de la derecha al Período 2. En dicha figura se observa que, independientemente del tipo de diagrama, la comprensión del modelado dinámico en OML requiere menos tiempo que en UML.

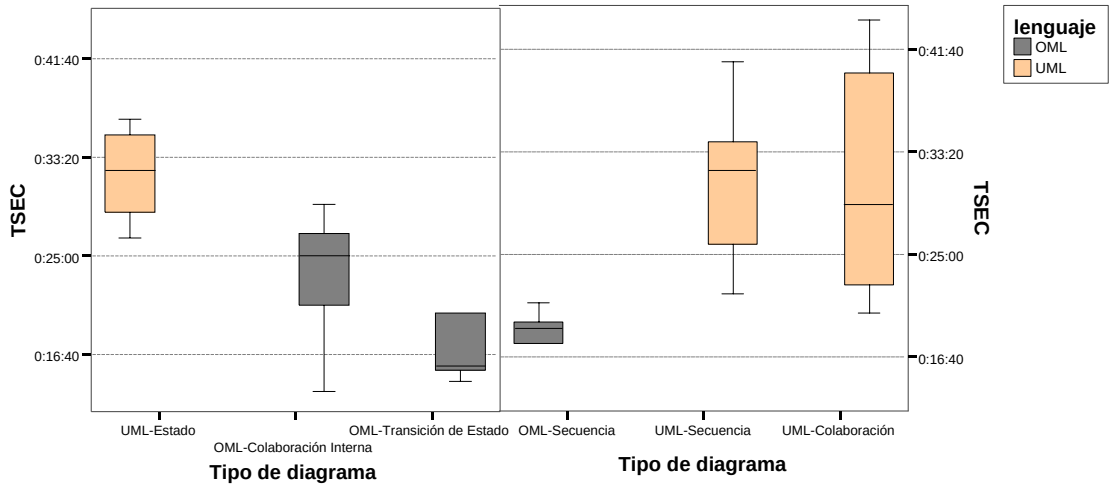


Figura 8.2. Gráficos de caja para cada tipo de diagrama UML y OML con respecto al tiempo

La Figura 8.3 muestra dos gráficos de caja con respecto a la variable puntuación total (NRESP). El gráfico de la izquierda corresponde al Período 1 y el de la derecha al Período 2. Al igual que antes se observa que, independientemente del tipo de diagrama, la comprensión del modelado dinámico en OML obtiene un mayor número de respuestas correctas que si se representa en UML.

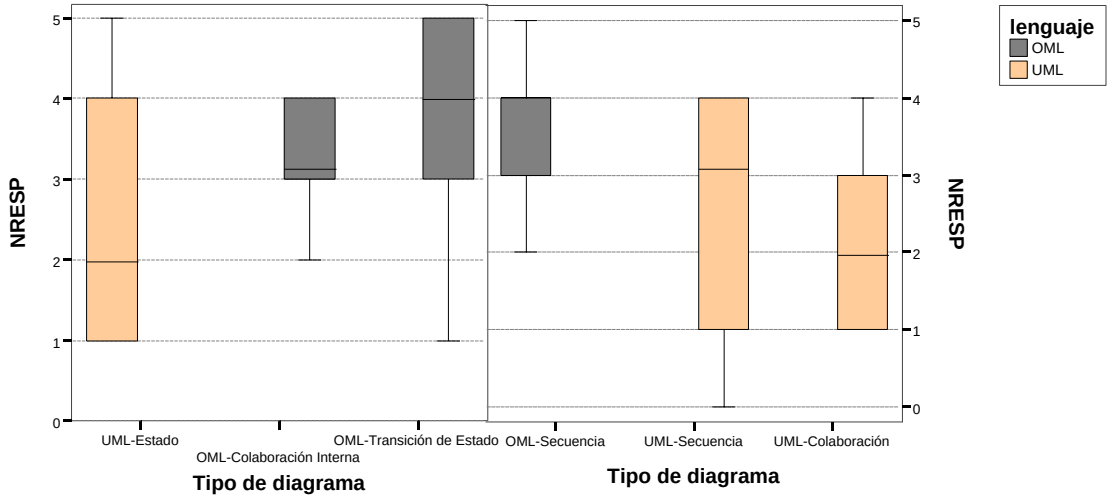


Figura 8.3. Gráficos de caja para cada tipo de diagrama con respecto a la puntuación

Por último, no debemos olvidar que en el Período 2 se realizó una segunda tarea experimental. Los tiempos empleados en realizarla aparecen resumidos en la Tabla 8.8. Esta tarea, que consistió en comprobar la consistencia de la traza entre los mensajes de los diagramas de interacción y los métodos de los diagramas de clases, estaría englobada dentro de la actividad de V&V (Verificación y Validación) de los modelos de una aplicación software. Tal como se puede ver en la Tabla 8.8 dicha tarea se efectuó para dos escenarios diferentes, pero se llevó a cabo por los mismos sujetos. Por ello, de un escenario a otro se aprecia un efecto de aprendizaje en cuanto al tipo de lenguaje, pues en el segundo par de casos de uso los tiempos invertidos fueron menores. Sin embargo, la conclusión es la misma. La verificación de la traza de los mensajes de los escenarios especificados en OML consume menos tiempo que representados en UML.

		Escenarios	
		Cliente Hace un Pago – Cliente Inserta un Billete Válido	Dispensar Cambios – Dispensar 20 centavos
Lenguaje	OML	0:08:39 - 0:02:37	0:07:45 - 0:03:53
	UML	0:09:18 - 0:04:00	0:08:07 - 0:03:09

Tabla 8.8. Estadísticos descriptivos del tiempo (media - desviación típica) de verificación

8.7. Presentación

Es importante que los materiales elaborados y los datos recopilados estén accesibles. Toda esta información se puede hallar:

- En este documento

  Material: Apéndice C

  Datos: Apéndice G
- En el CD dentro de la carpeta Experimento4
- En la página WEB http://www.vc.ehu.es/jiwotvim/ecdmOMLUML/ecdmOMLUML.htm (en inglés).

Capítulo 9

Resumen de los resultados

9.1. Introducción

El concepto clave especificado en el objetivo de los cuatro experimentos que engloba esta investigación es la comprensión. Dentro del mundo psicoeducativo la comprensión es un término complejo a la hora de definirlo. Pero existe acuerdo en considerar que la comprensión de un texto nos remite a la información contenida en él y a la representación que el sujeto construye sobre esa información. Actualmente, la comprensión se concibe como un proceso a través del cual el sujeto lector interactúa con el texto, elaborando un significado por la vía de aprender las ideas relevantes del texto y relacionarlas con las ideas que ya posee [Pear84]. En el área de la enseñanza la comprensión forma parte de un proceso cíclico de aprendizaje: aprender a leer para aprender a comprender y para, por último, aprender a escribir. Además, la comprensión es un proceso asociado al lenguaje y es parte integral de las técnicas del lenguaje: la lectura y la escritura.

Ubicándonos en el contexto de la ingeniería del software, el tema de la comprensión se abordó primeramente en la etapa de programación. En [Upch02] se encuentra recopilada una amplia bibliografía actualizada y comentada desde los años 80 sobre la lectura del código fuente y la comprensión del programa. Sin embargo, sólo algunos de los artículos de dicha bibliografía aportan datos empíricos sobre la comprensión de diferentes métodos o lenguajes de programación, debido tanto a la dificultad de conseguir sujetos válidos como a la inversión de tiempo requerida para experimentar.

Posteriormente la necesidad de evaluar empíricamente los productos obtenidos a lo largo del desarrollo de software se trasladó a la etapa de diseño. Ya en el año 1987 B. Boehm dejó constancia de que solventar errores después de la entrega del software era más caro que en las primeras etapas de desarrollo en una escala de 5:1 para aplicaciones no críticas y de tamaño

pequeño [Boeh87]. Este problema sigue siendo el primero en orden de prioridad de los 10 de una lista para reducir defectos, pero ahora en una escala de 100:1, dada la complejidad de los sistemas de información actuales [Boeh01]. Las afirmaciones publicadas en dicha lista se encuentran respaldadas experimentalmente por el centro CeBASE (C*enter for Empirically Based Software Enginnering*) [CeBA02]. En estos estudios empíricos la comprensión no era el factor clave a investigar, sino que era una labor implícita en la evaluación de nuevas técnicas de revisión, verificación y validación con el objetivo de detectar defectos.

Del mismo modo que se examina la comprensión de los lenguajes de programación [Prec00], es lógico plantear el estudio de la comprensión de los lenguajes de modelado. Precisamente el foco de investigación de esta tesis es la comprensión semántica del modelado dinámico. Sin embargo, ¿cuál es el fin último de esta investigación? La respuesta es la misma que en el mundo de la educación, pero referida al área de la ingeniería del software. Así pues, el objetivo prioritario consiste en evaluar la comprensión como parte del proceso de aprendizaje para modelar software, con el propósito final de orientar al diseñador de aplicaciones en la construcción de modelos dinámicos. Para ello es necesario sintetizar los resultados obtenidos en esta investigación empírica. En los apartados 9.2 y 9.3 se resumen los datos obtenidos en la experimentación con UML respecto al tiempo total y a la puntuación total respectivamente. Luego, en el apartado 9.4 se reflexiona sobre la repercusión de la capacidad semántica del modelado dinámico en UML. Para finalizar, se examina la comprensión de los modelos dinámicos de OML versus los correspondientes de UML.

9.2. Compendio de los datos de UML para el tiempo total

Los resultados obtenidos a partir de los tres primeros experimentos con respecto al tiempo total (TSEC) se visualizan en la Figura 9.1. Este gráfico de 3 dimensiones muestra en el plano XY los dos factores objeto de estudio, en el eje X el diagrama dinámico y la combinación de pares de diagramas, y en el eje Y el dominio de aplicación, clasificado con respecto a su complejidad partiendo de los sistemas de gestión de información a los sistemas en tiempo real (desde Biblioteca hasta Dictáfono). Por último, en la tercera dimensión, el eje Z, se representa la métrica del tiempo según el formato hh:mm:ss.

En la Figura 9.1 se puede observar que hay una gran diferencia entre los tiempos totales del primer experimento y aquellos de los dos estudios posteriores. La principal causa reside en el tipo de diseño experimental. En la primera práctica empírica se utilizó un diseño de cuadrado latino, lo que implicó que tanto el diagrama dinámico como el domino de aplicación

del diseño especificado en UML fuesen diferentes en cada sesión. En cambio, el tipo de diseño que se planificó para los dos siguientes experimentos, que estaban condicionados entre sí, fue uno denominado diseño de parcelas divididas. En este caso el tipo de diagrama dinámico se repitió en cada sesión de la réplica (segundo experimento), y luego la combinación de pares de diagramas en cada período del tercer experimento. El único factor cambiante en las diferentes sesiones de estos dos estudios fue el tipo de dominio de aplicación.

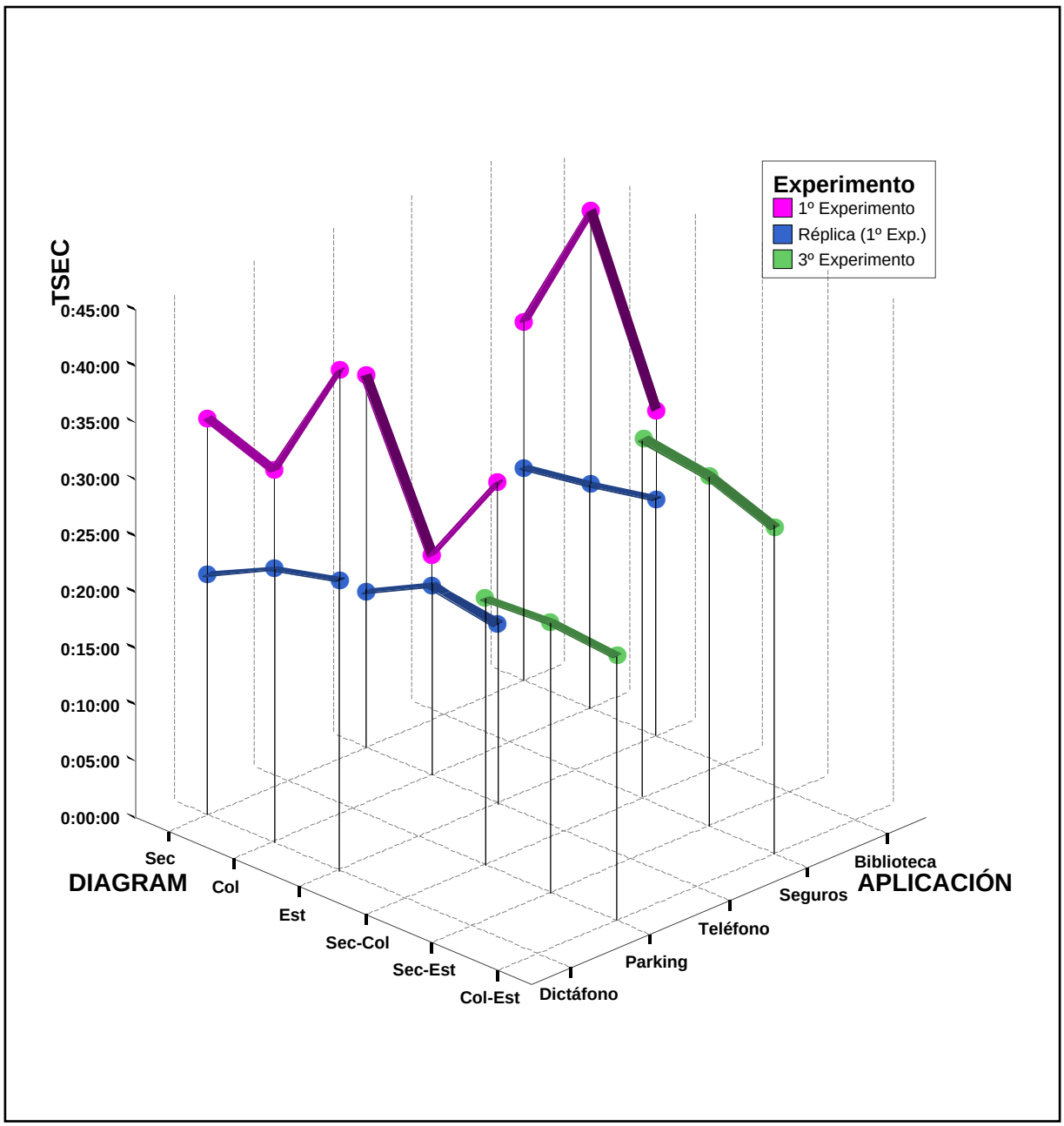


Figura 9.1. Gráfico de líneas para el tipo de diagrama y el dominio de aplicación en cada experimento con respecto al tiempo total

En el gráfico de la Figura 9.1 se puede comprobar que, para la réplica y para el tercer experimento, las líneas que representan al tiempo total (TSEC) son prácticamente rectas, sin oscilaciones con respecto al tipo de diagrama y el dominio de aplicación de los diseños. Esto confirma que esta métrica TSEC no fuese significativa estadísticamente. Por el contrario, las fluctuaciones que se visualizan en dicho gráfico para el primer experimento varían con respecto al diagrama dinámico y al dominio de aplicación, lo que verifica la relevancia estadística de la interacción de estos dos factores.

Por último, en la Figura 9.1 se observa que los diseños que tienen valores mayores en la variable TSEC cumplen en general dos condiciones: pertenecen a los dominios de aplicación más complejos y están modelados bien con diagramas de colaboración o con diagramas de estado. En definitiva, la comprensión del modelado dinámico en UML es más rápida cuando el comportamiento se implementa mediante diagramas de Secuencia y, además, es independiente del dominio de aplicación. A la hora de integrar diagramas, el par Secuencia-Estado es la combinación que ofrece una mayor velocidad de comprensión del modelado dinámico en notación UML.

9.3. Compendio de los datos de UML para la puntuación total

En el gráfico tridimensional de la Figura 9.2 se muestran las puntuaciones obtenidas en los tres experimentos sobre UML. En los ejes X e Y de la Figura 9.2 se representan los mismos factores que en la Figura 9.1, mientras que el eje Z corresponde a la métrica NRESP que mide el número de respuestas correctas.

En la Figura 9.2 se observa que los resultados del primer experimento, que es el original, y de su réplica son parecidos con respecto a la puntuación total. Una conclusión evidente es que la semántica de un diagrama de Estado ayuda a comprender mejor el modelado dinámico si se trata del dominio de un sistema reactivo en tiempo real. Considerando la puntuación en el tercer experimento, los diseños de ambos dominios de aplicación alcanzan el mayor número de aciertos cuando el comportamiento dinámico se implementa mediante el par Secuencia-Estado.

Por otra parte, es importante examinar los datos de la tercera práctica empírica comparándolos con los valores obtenidos en el primer y segundo experimento (réplica). Esta comparación se va a realizar con respecto a la puntuación (NRESP), pues con respecto al tiempo total los resultados no resultaron estadísticamente significativos en el tercer estudio.

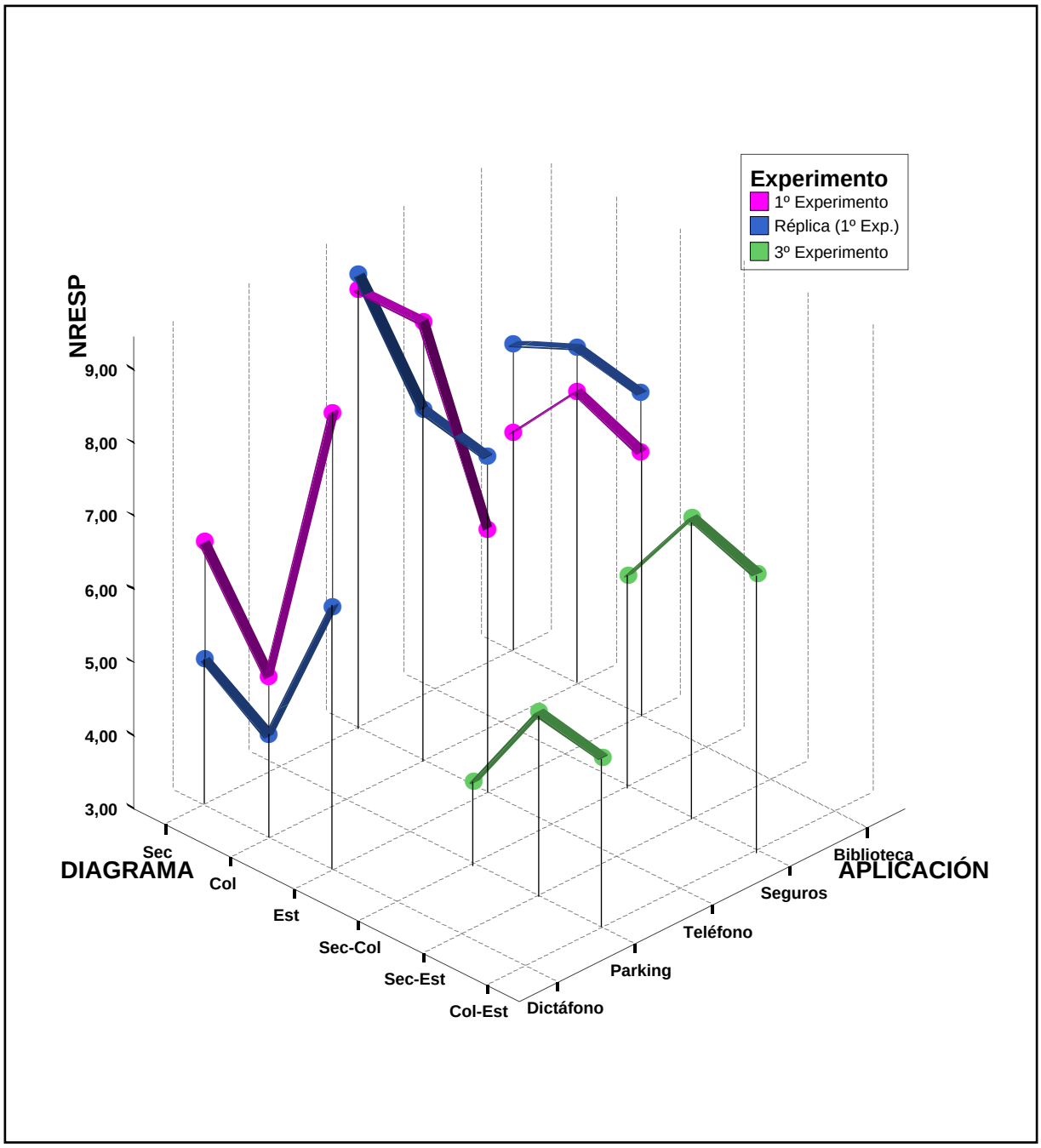


Figura 9.2. Gráfico de líneas para el tipo de diagrama y el dominio de aplicación en cada experimento con respecto a la puntuación total

Los dos documentos de diseño utilizados en el tercer experimento corresponden a un sistema de gestión de Seguros y a un sistema de control de un Parking.

El diseño de Seguros (sistema de gestión de información) comunica mejor la información cuando los modelos dinámicos implementados integran el par Secuencia-Estado, tal como se observa en la Figura 9.2. Comparado con el diseño de la Biblioteca (sistema de gestión de información), la relación de los tres tipos de diagramas de mayor a menor comprensión es:

Colaboración > Secuencia > Estado. Ahora bien, dicha relación podría ser la siguiente: Secuencia = Colaboración = Estado, dada la escasa oscilación que se puede apreciar entre los tres diagramas dinámicos en la Figura 9.2. Para valorar mejor esta pequeña diferencia se presenta en la Figura 9.3 un resumen de los porcentajes de acierto para cada diagrama y para cada combinación par de los mismos, considerando sólo el dominio de aplicación de gestión de información. Cabe destacar que el par Secuencia-Estado aporta una mayor comprensión de los modelos del comportamiento dinámico que si se hubiesen representado únicamente mediante los diagramas individuales Secuencia y Estado que integran el par, un 71% (S-E) frente a un 66% (Secuencia) y un 70% (Estado). En contrapartida, en un diseño UML dicho modelado es más comprensible si sólo incluye diagramas de Colaboración que si se combina con alguno de los otros dos diagramas, un 73% (Colaboración) frente a un 58% (S-C) y un 67% (C-E).

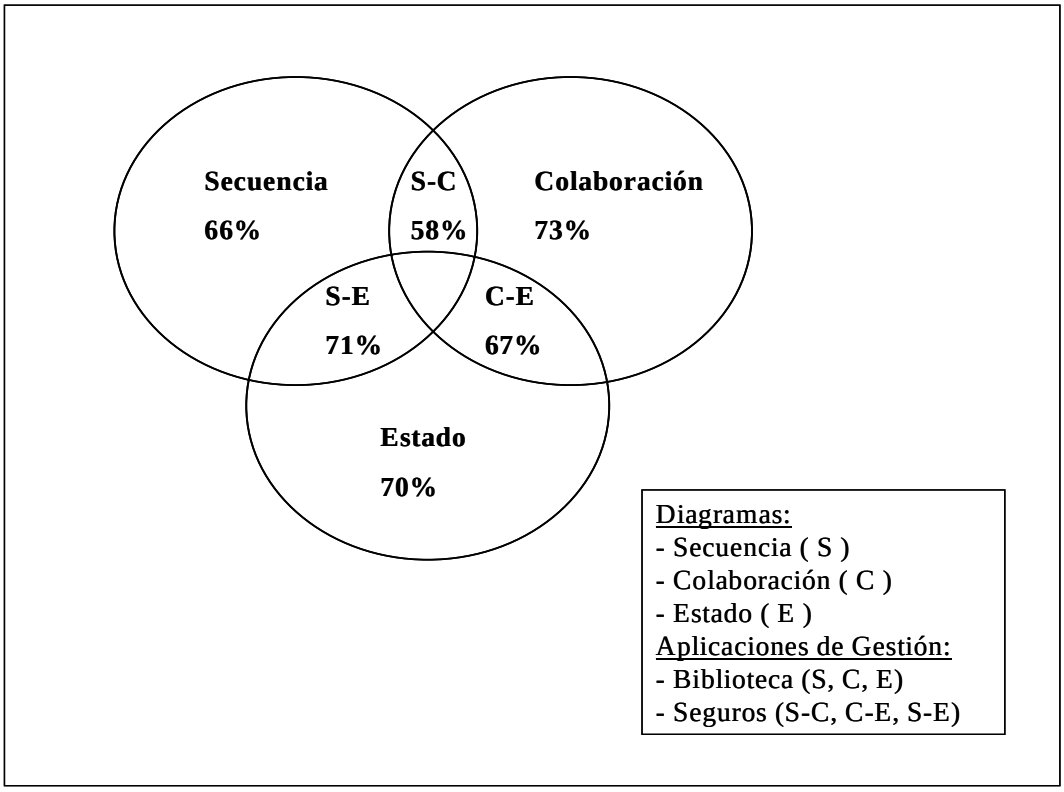


Figura 9.3. Porcentaje de acierto para aplicaciones de gestión

La interpretación del diseño del Parking (sistema no reactivo en tiempo real) es más segura, cuando el comportamiento dinámico se modela utilizando la composición Secuencia-Estado. Si el dominio de aplicación de este diseño se asocia con los diseños del Teléfono (sistema no reactivo en tiempo real y escenario de menor complejidad) y del Dictáfono

(sistema reactivo en tiempo real y escenario de mayor complejidad), la relación de los tipos de diagramas dinámicos clasificados en orden de mayor a menor comprensión es: Secuencia > Colaboración > Estado y Estado > Secuencia > Colaboración respectivamente (ver Figura 9.2). El primer diagrama en ambas relaciones es Secuencia y Estado, cuya combinación par es Secuencia-Estado.

9.4. Eficacia de la capacidad semántica del modelado dinámico en UML

Con el fin de proporcionar al diseñador de aplicaciones unas guías sobre la construcción de modelos dinámicos en función del dominio de aplicación, vamos a establecer qué tipo de diagrama o qué combinación de pares de diagramas es más eficaz. Curiosamente los términos eficaz, eficiente y efectivo no se suelen utilizar correctamente. Por ejemplo, en [Lait00] y [Trav99b] se estudia la eficacia (en inglés, *effectiveness*) de una serie de técnicas, que muchas veces los revisores traducen erróneamente como efectividad. Asimismo el adjetivo *effective* es otro falso amigo, que no significa efectivo, sino eficaz o eficiente. No debe confundirse efectivo con eficaz/eficiente, pues significa “real y verdadero, en oposición a lo quimérico”. Por otra parte, se emplea “eficaz” principalmente para seres inanimados y “eficiente” para seres animados, dado que la eficiencia es una virtud o facultad más propia de los seres vivos.

Así pues, vamos a examinar la eficacia de los tipos de diagramas dinámicos de UML, individualmente y combinados por parejas. Pero, ¿en qué métrica nos vamos a basar para determinar la eficacia? Cualquiera de las dos métricas definidas en los experimentos de esta investigación sería válida, al no estar correlacionadas (ver apartado 2.3.5). Aún así, el tiempo total (TSEC) no parece la más indicada, ya que mide el tiempo que se tarda tanto en responder bien como en contestar mal. Aunque el número de respuestas correctas (NRESP) podría ser una medida de dicha eficacia, hay otra que se aproxima más. Normalmente, se dice que una persona es eficiente o una máquina es eficaz, cuando desarrolla su trabajo de forma óptima en el menor tiempo. En nuestro caso, los diagramas más eficaces corresponderán a aquellos que consuman menos tiempo en cada respuesta correcta. Por lo tanto, la eficacia es el cociente de la división entre el tiempo total y el número de respuestas correctas, es decir, TSEC / NRESP. Esta métrica se va a denominar TSECRESP.

En la Figura 9.4 se muestra la eficacia del modelado dinámico en UML para cada diagrama o combinación par de diagramas y para cada dominio de aplicación. Por otra parte, en este gráfico no se visualiza la variable TSECRESP para cada uno de los tres experimentos. Con el objeto de obtener las medias para cada factor, en la Figura 9.2 se representa la media

conjunta de la eficacia de los dos primeros estudios, pues se refieren a los mismos tipos de diagramas y de dominios de aplicación (líneas con puntos de color rosa), así como la media de la eficacia del tercer experimento (líneas con puntos de color azul).

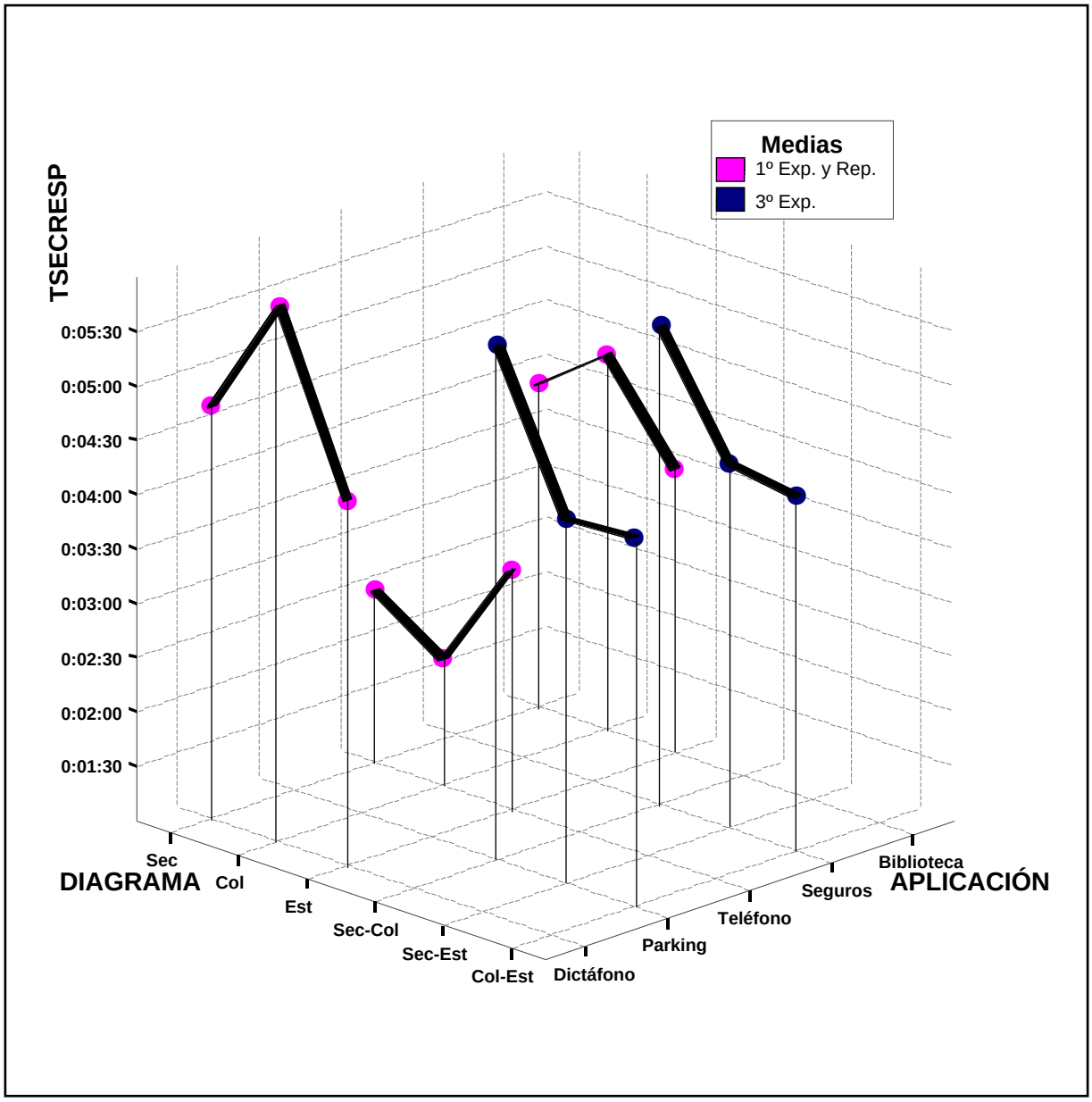


Figura 9.4. Gráfico de líneas para el tipo de diagrama y el dominio de aplicación respecto a la eficacia

El modelado dinámico es una tarea difícil de acometer dentro del proceso de desarrollo de aplicaciones software, en la que además es necesario invertir mucho tiempo. Para paliar el trabajo que conlleva dicha tarea podemos aplicar las guías de estilo de [Amb03] a la hora de dibujar diagramas de secuencia, de colaboración y de estado en notación UML, o incluso seguir los pasos propuestos en [Song01] para desarrollar diagramas de interacción de UML.

En ninguna de esas alternativas se tiene en cuenta el dominio de aplicación. Precisamente este factor junto con el tipo de diagrama dinámico se resumen en la Tabla 9.1 con respecto a su eficacia. En la primera y segunda columna las aplicaciones están ordenadas de mayor a menor complejidad en cuanto a su dominio, de gestión de información a tiempo real. Asimismo cada una de las filas se interpreta de mayor a menor eficacia en la comprensión. Por ejemplo, Secuencia < Estado < Colaboración indica que el diagrama de secuencia requiere menor inversión de tiempo por respuesta acertada, lo que ayuda a entender más eficazmente la semántica de los modelos dinámicos UML para una aplicación de gestión.

		Eficacia →		
		+		-
Gestión	Biblioteca	Secuencia <	Estado <	Colaboración
↓	Seguros	Secuencia-Estado =	Colaboración-Estado <	Secuencia-Colaboración
	Teléfono	Colaboración <	Secuencia <	Estado
	Parking	Colaboración-Estado =	Secuencia-Estado <	Secuencia-Colaboración
Real	Dictáfono	Estado <	Secuencia <	Colaboración

Tabla 9.1. Síntesis de la eficacia en la comprensión para cada diagrama y dominio de aplicación

En definitiva, la segunda y tercera columna son las que nos informan sobre qué condiciones deben concurrir para conseguir una comprensión eficaz de los modelos dinámicos. Si se trata de una aplicación de gestión (Biblioteca y Seguros), los diagramas son Secuencia o la composición Secuencia-Estado o Colaboración-Estado. Para un sistema no reactivo en tiempo real (Teléfono y Parking) los diagramas son Colaboración o el par Colaboración-Estado o Secuencia-Estado. Finalmente, cuando se trata del diseño de un sistema reactivo en tiempo real, el diagrama es el de Estado.

Aplicando el proceso cíclico de aprendizaje mencionado en el apartado 9.1 a nuestro caso, nos encontramos en la fase de aprender a comprender para aprender a construir modelos dinámicos. Por lo tanto, los resultados de la Tabla 9.1 pueden ser útiles a los desarrolladores de software, pues su contribución repercute no sólo en la comprensión sino también en la construcción de los diseños UML, por lo menos en lo que respecta al modelado dinámico.

9.5. Eficacia del modelado dinámico en OML versus UML

Para una aplicación de gestión la comprensión del comportamiento dinámico resultó más eficaz si se especificaba mediante diagramas individuales que mediante combinaciones de ellos (ver Figura 9.3). En este caso los diagramas pertenecían a UML. Pero, ¿cómo sería esa comprensión de los diagramas individuales cuando la notación OO utilizada fuese otra? Para

responderla, se decide adoptar la misma métrica que en el apartado 9.4, que se denomina TSECRESP y que representa la eficacia en la tarea de comprender. Se entiende que la comprensión de la parte dinámica de un diseño en un lenguaje de modelado, por ejemplo, en UML, es más eficaz que en otro lenguaje, por ejemplo, en OML, siempre que el tiempo invertido en interpretar correctamente la información sea en UML menor que en OML. Para establecer dicha comparativa en la Figura 9.5 se resume la eficacia para cada lenguaje de modelado estándar (UML y OML) y para sus correspondientes tipos de diagramas dinámicos, pero sólo respecto al dominio de un sistema empotrado.

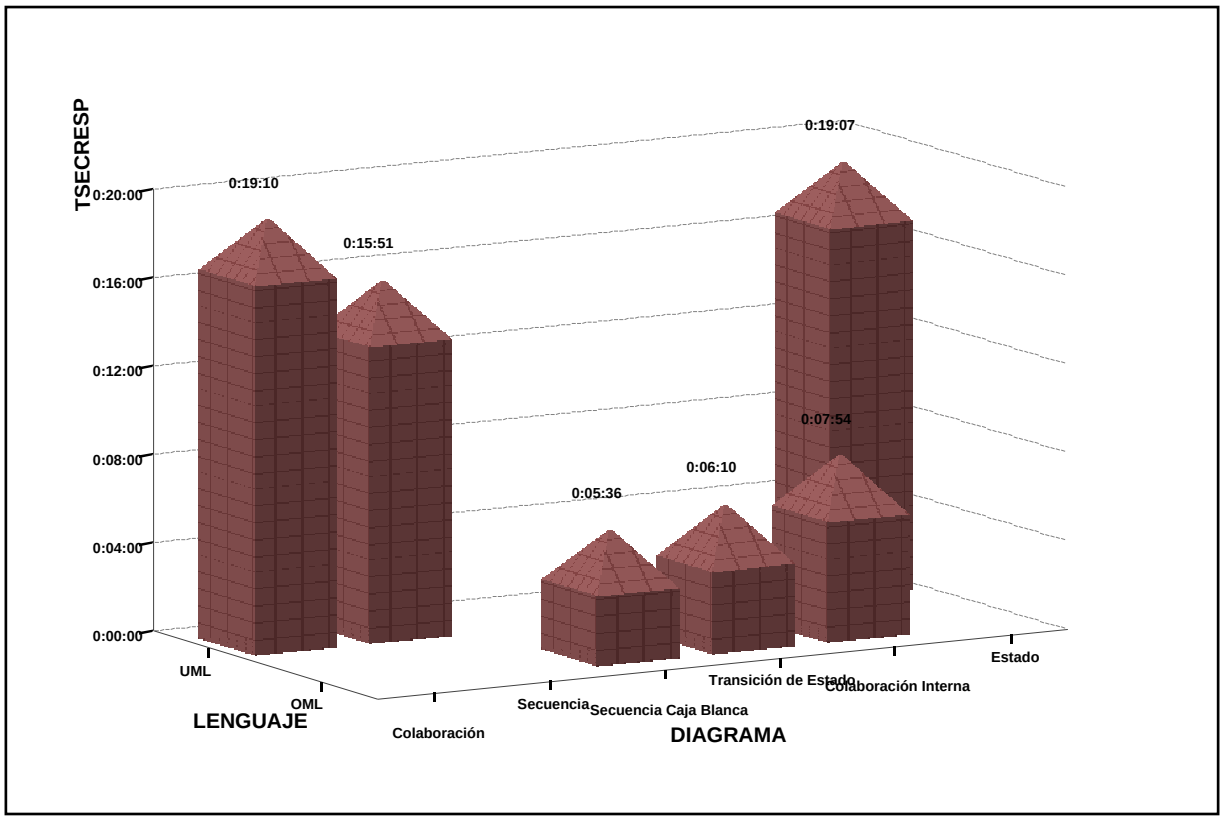


Figura 9.5. Gráfico de barras de la eficacia en la comprensión de OML y UML

En el apartado 8.6 se comprobó que la especificación del comportamiento dinámico mediante OML es más rápida de comprender y más fácil de interpretar que mediante UML. En consecuencia, es lógico que la semántica del modelado dinámico de un diseño en lenguaje OML se comprenda más eficazmente que en UML. Tal como muestra la Figura 9.5, el tiempo consumido en comprender dicho modelado en OML es menor que en UML, y además independientemente del tipo de diagrama dinámico implementado.

PARTE IV

CONCLUSIONES

Capítulo 10

Conclusiones

10.1. Introducción

Sea cual fuere el área de investigación, la importancia de la experimentación como validación o refutación de una teoría es innegable. En medicina, por ejemplo, nadie aceptaría una droga no experimentada previamente. En cambio, en ingeniería del software, disciplina que está en continuo cambio, se admiten nuevas tecnologías sin una comprobación fehaciente de sus beneficios. Existen ejemplos claros, tales como la programación orientada a objetos o las técnicas formales, que se asumen que mejoran la productividad o la calidad de los artefactos, pero sin ninguna evidencia empírica. Así pues, es preciso que la experimentación constituya un componente intrínseco de la ingeniería del software, para poder alcanzar la madurez experimental propia de otras disciplinas.

Esta tesis se enmarca dentro de una de las líneas de investigación experimental propuesta en el workshop Empirical Studies of Software Development and Evolution (*ESSDE*) celebrado en Los Ángeles en mayo de 1999 [Bria99, p. 396]. Esta línea propugnaba investigar el uso del lenguaje UML, sugiriendo que un subconjunto de la notación se podría utilizar de forma más eficiente en ciertos contextos.

Precisamente, el objetivo principal de esta tesis es evaluar la comprensión de los modelos dinámicos contenidos en un documento de diseño y especificados en un lenguaje de modelado orientado a objetos. Pero el objetivo no se centra en el uso del lenguaje, sino en la comprensión, pues en el proceso cíclico de aprendizaje de cualquier técnica o metodología, antes de usarla es condición necesaria comprenderla. También se explora la perspectiva dinámica y no la estática del diseño OO, debido a que para modelar el comportamiento dinámico los dos lenguajes estándares ofrecen una mayor variedad de diagramas. De esta forma, se respalda la sugerencia anterior sobre la posibilidad de que el uso de un subconjunto

de ellos sea más eficaz en función de una serie de factores relacionados con el documento de diseño.

Las aportaciones de esta investigación se exponen en el apartado 10.2, mientras que las conclusiones y los futuros trabajos que se derivan a partir de ella se detallan en los apartados 10.3 y 10.4 respectivamente.

10.2. Aportaciones

Entre las aportaciones para la consecución del objetivo de esta tesis cabe destacar:

- La aplicación del paradigma experimental, que incluye el método GQM, como mecanismo para verificar qué técnicas son más fáciles de comprender y, por tanto, resultan más apropiadas a la hora de modelar el comportamiento dinámico.
- La planificación, bajo un enfoque evolutivo, de una familia de experimentos controlados.
- La elección cuidadosa del tipo de diseño adecuado para acometer cada una de las prácticas empíricas, adoptando tipos de diseños experimentales propios de otras áreas, tales como la agricultura o la farmacología.
- La incorporación de los principios de diseño experimental, técnicas de control y de replicación, a los cuatro estudios empíricos, con el fin de reducir el error experimental y maximizar las propiedades de los experimentos.
- La estimación del tamaño muestral mínimo que se necesita para llevar a cabo la ejecución del segundo experimento (réplica del primero), al disponer de datos sobre las medias y las desviaciones típicas de un estudio previo.
- La rigurosidad en el análisis de los datos recopilados debido principalmente a dos razones, la aplicación del modelo lineal estadístico de acuerdo al tipo de diseño de cada experimento y la verificación de los resultados obtenidos mediante la utilización de tres paquetes estadísticos (JMP, NCSS y SPSS).
- El establecimiento tanto del error de tipo I como del error de tipo II con un convenio común para el contraste de hipótesis en los cuatro experimentos, donde las decisiones correctas se han tomado con una seguridad del 90% en el caso de aceptar la hipótesis nula y con una potencia del 80% en el supuesto de rechazar la hipótesis.
- La creación de paquetes de laboratorio correspondientes a los experimentos y disponibles vía web para su replicación, donde se puede encontrar, además del material experimental, los datos y los ficheros de sintaxis codificados en los tres paquetes de software estadístico mencionados antes.

10.3. Conclusiones

Las conclusiones derivadas a partir de la ejecución de los experimentos que engloban esta tesis para evaluar el modelado dinámico son las siguientes:

- La dificultad de los sujetos para comprender los modelos dinámicos, también denominada complejidad cognitiva [Fent97], se ha evaluado en función de dos métricas: el tiempo total invertido en leer y comprender y la puntuación obtenida tras responder correctamente.
- En primer lugar, se ha comparado la comprensión semántica de tres diagramas individuales (secuencia, colaboración y estado) para representar el comportamiento dinámico en UML. Tanto en el primer experimento como en la réplica se concluye que la comprensión del modelado dinámico depende del tipo de diagrama y del dominio de aplicación. Cuando se trata del diseño de un sistema reactivo en tiempo real, el diagrama de estado proporciona una interpretación más segura de la información del modelado dinámico, mientras que en el caso de un sistema no reactivo en tiempo real, el diagrama de secuencia es el que aporta mayor estabilidad semántica. En cambio, ninguno de los diagramas estudiados incrementa la capacidad de comunicación de la especificación dinámica de un diseño UML, si corresponde a una aplicación de gestión. Por otra parte, el diagrama de secuencia es el más rápido de leer y comprender, deducción que no depende del dominio de aplicación. Este último resultado se ajusta a la opinión de Fowler en [Smit02], donde dice que en la práctica hay tres técnicas importantes en UML que deberían formar parte del núcleo en su próxima versión: casos de uso, diagramas de clases y diagramas de secuencia. El resto de las técnicas se podrían definir por medio de algún tipo de mecanismo de extensión.
- El siguiente paso ha sido explorar las combinaciones de pares de diagramas (colaboración-estado, secuencia-estado y secuencia-colaboración). Independientemente de que el dominio de aplicación sea de gestión o de control, el modelado dinámico del diseño en UML se comprende más eficazmente cuando se implementa mediante el par secuencia-estado.
- Por último, teniendo en cuenta que la notación en que se especifica un diagrama debe ser fácil de comprender, de aprender y de usar, se ha investigado si el tipo de lenguaje (UML versus OML) usado para construir los modelos dinámicos influye o no en su comprensión. Los resultados obtenidos revelan que la especificación del comportamiento dinámico mediante OML es más rápida de comprender y más fácil de interpretar que mediante

UML. Esta conclusión también se puede extender al modelado de la estructura estática, pero está limitada al dominio de aplicación de un sistema empotrado. Además, esta conclusión corrobora las características de las notaciones de ambos lenguajes estándares. Los diseñadores de la notación tanto de UML como de OML han acomodado elementos heterogéneos ya existentes en las notaciones de sus creadores. Sin embargo, en la notación del lenguaje OML se integraron ideas semióticas, que estudian el significado de los signos y de los símbolos. Estas ideas han repercutido en que los iconos de los diagramas de OML sean más intuitivos que en UML, tanto en el modelado estático como dinámico.

- El objetivo final, al experimentar la comprensión del modelado dinámico en UML, es facilitar la utilización del lenguaje en la creación de modelos. Uno de los propósitos que persiguen los diseñadores del lenguaje UML es que los usuarios sean capaces de adaptar UML a dominios específicos. Para lograrlo, es necesario mejorar los mecanismos de extensión de esta notación utilizados en la creación de perfiles (*profiles*) que especializan UML. En [OMG01] se definen *profiles* de UML para procesos de negocio y para procesos de software. Asimismo en [Ambl02] se propone un *profile* de modelado de datos no oficial para los diagramas de clases de UML. También en algunas herramientas CASE, tales como Rational Rose y Objecteering/UML, los usuarios ya disponen de unos perfiles de UML específicos a los lenguajes de programación (por ejemplo, C++, Java y C#). Así pues, los *profiles*, también conocidos como extensiones ligeras (*lightweight extensions*), juegan un papel importante en la adaptación de UML a dominios de aplicación y a contextos de trabajo, tales como el análisis, el diseño y la codificación. En esta adaptación los perfiles se utilizan para expresar qué elementos y técnicas se deben incluir en el caso de un dominio concreto de aplicación software y de una fase del proceso de desarrollo, pues no todos los elementos son relevantes para todos los tipos de trabajo. Los resultados de esta evaluación empírica han demostrado que los elementos semánticos que incorporan los diagramas dinámicos se adaptan mejor a la semántica de un dominio de aplicación. Por tanto, estas conclusiones pueden servir de instrumento para el diseño y la creación de perfiles especializados en el marco de un dominio de aplicación software y en concreto para la etapa de diseño. Asimismo, estos resultados aportan una base empírica a la hora de decidir qué artefactos son adecuados para modelar en los métodos ágiles [Ambl01].

10.4. Futuras líneas de investigación

El propósito de esta tesis ha sido integrar la experimentación como vehículo para la evolución de la ingeniería del software, al menos en lo que se refiere a la adopción de lenguajes de modelado orientado a objetos. Continuando en esta línea, entre los futuros trabajos cabe señalar:

- Explorar la experimentación del modelado dinámico especificado en OML en otros dominios de aplicación, con el fin de validar si realmente las técnicas de diagramación de la notación de OML son mejores que las de UML y se pueden hacer extensivas a otros contextos de trabajo.
- Automatizar estos resultados empíricos, incorporándolos en las herramientas CASE como soporte a la construcción de *profiles* para dominios específicos.
- Aplicar técnicas de meta-análisis para combinar los resultados obtenidos y extrapolar dicho conocimiento.

  De hecho, el objetivo principal del meta-análisis es encontrar patrones regulares a partir de un conjunto de datos derivados de diferentes experimentos que aborden un mismo tema. Dentro de la comunidad de la ingeniería del software ya se ha empezado a investigar las técnicas del meta-análisis. En [Pick98] se aplicaron a estudios observacionales para explorar la relación entre el esfuerzo del proyecto y el tamaño del producto, mientras que en [Mill00] se aplicaron a tres experimentos controlados ([Basi87], [Kams96] y [Rope97]) para comparar sus resultados sobre la eficacia de tres técnicas de detección de defectos.
- Crear una base de datos común como fuente de información en la ingeniería del software, donde se puedan integrar tanto nuestros resultados experimentales sobre el modelado dinámico como los de futuros trabajos empíricos.

  Es importante mantener un repositorio que contenga registros de todos los experimentos acerca de los temas de interés en la ingeniería del software. Por ejemplo, el Instituto Nacional de la Salud de EE. UU. ha desarrollado el sitio web http://clinicaltrials.gov, donde se pueden consultar los resultados empíricos de pruebas clínicas realizadas en la investiga-ción médica. En cambio, en nuestra disciplina se pueden encontrar las siguientes iniciativas en forma de herramientas CASE [Tori99], portales [ESER02] o páginas web [OSLO02], que mantienen sus propios repositorios. Mediante estas iniciativas estamos en el buen camino de crear un cuerpo de conocimiento [Basi99] y así ir avanzando en el

largo proceso de madurez ya experimentado por otras disciplinas, tales como la medicina o la física.

Para afrontar cualquiera de estas nuevas líneas de investigación me gustaría mencionar dos pares de máximas de Edgar Dijkstra, cuya evocación me ha acompañado, y también ayudado, a lo largo de la ejecución de esta tesis:

"No podemos mejorar algo a menos que podamos evaluarlo"

"Un experimento puede demostrar la presencia de fallas en una teoría, nunca su ausencia".

BIBLIOGRAFÍA

Referencias bibliográficas

[Ambl01] Ambler, S. W., *The Official Agile Modeling (AM) Site*, 2001-2002. <www.agilemodeling.com/essays/whenIsAModelAgile.htm> [Consulta: noviembre de 2002].

[Ambl02] Ambler, S. W., "A UML profile for data modelling", *Agile Data*, 2002. <www.agiledata.org/essays/umlDataModelingProfile.html> [Consulta: diciembre de 2002].

[Ambl03] Ambler, S. W., *The Elements of UML Style*, Cambridge University Press, 2003. <www.ambysoft.com/elementsUMLStyle.html> [Consulta: diciembre de 2002].

[Armi97] Armitage, P. y Berry, G., *Estadística para la Investigación Biomédica*, third edition, Madrid (España): Harcourt Brace, 1997.

[Barb00] Barbier, F. y Henderson-Sellers, B., "Object modelling languages: An evaluation and some key expectations for the future", *Annals of Software Engineering*, vol. 10, 2000, pp. 67-101.

[Basi75] Basili, V. R. y Turner, A. J., "Iterative enhancement: A practical technique for software development", *IEEE Transactions on Software Engineering*, vol. 1, nº 4, 1975.

[Basi86] Basili, V. R., Selby, R. W. y Hutchens, D. H., "Experimentation in Software Engineering", *IEEE Transactions on Software Engineering*, vol. 12, nº 7, julio 1986, pp. 733-743.

[Basi87] Basili, V. R. y Selby, R. W., "Comparing the effectiveness of software testing strategies", *IEEE Transactions on Software Engineering*, vol. 13, nº 12, diciembre 1987, pp. 1278-1296.

[Basi93] Basili, V. R., "The experimental paradigm in software engineering", *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, Lecture Notes in Computer Science 706, Springer-Verlag, 1993, pp. 3-12.

[Basi94] Basili, V. R., Caldiera, G. y Rombach, H. D., "Goal Question Metric paradigm", *Encyclopedia of Software Engineering*, vol. 1, Wiley, 1994, pp. 528-532.

[Basi96a] Basili, V. R., "The role of experimentation: Past, present and future", *Proc. of the 18th International Conference on Software Engineering*, 1996, pp. 442-450.

[Basi96b] Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S. y Zelkowitz, M. V., "The empirical investigation of perspective-based reading", *Empirical Software Engineering*, vol. 1, nº 2, 1996, pp. 133-164.

[Basi99] Basili, V. R., Shull, F. y Lanubile, F., "Building knowledge through families of experiments", *IEEE Transactions on Software Engineering*, vol. 25, nº 4, julio/agosto 1999, pp. 456-473.

[Bela72] Belady, L. A. y Lehman, M. M., "An introduction to growth dynamics", *Statistical Computer Performance Evaluation*, Nueva York (EE. UU.): Academic Press, 1972.

[Bela76] Belady, L. A. y Lehman, M. M., "A model of large program development", *IBM Systems Journal*, vol. 13, nº 3, 1976, pp. 225-252.

[Björ00] Björkander, M., "Graphical programming using UML and SDL", *IEEE Computer*, diciembre 2000, pp. 30-35.

[Blah98] Blaha, M. y Premerlani, W. J., *Object-Oriented Modeling and Design for Database Applications*, Nueva York (EE. UU.): Prentice-Hall, 1998.

[Boeh87] Boehm, B., "Industrial metrics top-10 list", *IEEE Software*, septiembre 1987, pp. 84-85.

[Boeh01] Boehm, B. y Basili, V. R., "Software defect reduction top-10 list", *IEEE Computer*, vol. 34, nº 1, enero 2001, pp. 135-137.

[Booc94] Booch, G., *Object-Oriented Analysis and Design with Applications*, second edition, Addison Wesley, 1994.

[Box88] Box, G. E. P., Hunter, W. G. y Hunter, J. S., *Statistics for Experimenters. An Introduction to Design, Data Analysis and Model Building*, Nueva York (EE. UU.): John Wiley & Sons, 1978 (traducido por Arimany, L. y Tort-Martorell, J., *Estadística para Investigadores. Introducción al Diseño de Experimentos, Análisis de Datos y Construcción de Modelos*, Barcelona (España): Editorial Reverté, 1988).

[Brit96] Brito e Abreu, F. y Melo, W., "Evaluating the impact of object-oriented design on software quality", *Proc. of the 3rd International on Software Metrics*, 1996, pp. 90-99.

[Brit01] Brito e Abreu, F., Henderson-Sellers, B., Piattini, M., Poels, G. y Sahraoui, H. A., "Quantitative approaches in object-oriented software engineering", *ECOOP'01 Workshop Reader*, LCNS 2323, Springer-Verlag, 2001, pp. 174-183.

[Bria97] Briand, L. C., Bunse, C., Daly, J. W. y Differding, C., "An experimental comparison of the maintainability of object-oriented and structured design documents", *Empirical Software Engineering*, vol. 2, nº 3, 1997, pp. 291-312.

[Bria99] Briand, L., Arisholm, E., Counsell, S., Houdek, F. y Thévenod-Fosse, P., "Empirical studies of object-oriented artifacts, methods and processes: State of the art and future directions", *Empirical Software Engineering*, vol. 4, nº 4, 1999, pp. 387-404.

[Camp73] Campbell, D. T. y Stanley, J. C., *Experimental and Quasi-experimental Design for Research*, first edition, Chicago (EE. UU.): Rand McNally, 1966 (traducido por Kitaigorodzki, M., *Diseños Experimentales y Cuasi-Experimentales en la Investigación Social*, Buenos Aires (Argentina): Amorrortu, 1973).

[Cart98] Cartwright, M., "An empirical view of inheritance", *Information and Software Technology*, vol. 40, nº 14, diciembre 1998, pp. 795-799.

[CeBA02] CeBASE, *NSF Center for Empirically Based Software Engineering*, University of Maryland (V. Basili), Fraunhofer Center MD (F. Shull), University of Southern California (B. Boehm), University of Nebraska-Lincoln (S. Henninger) y Mississippi State University (R. Vaughn), 2002.
<www.cebase.org/www/home/index.htm>
[Consulta: enero de 2002].

[Chid94] Chidamber, S. R. y Kemerer, C. F., "A metric suite for object-oriented design", *IEEE Transactions on Software Engineering*, vol. 20, nº 6, junio 1994, pp. 476-493.

[Coch80] Cochran, W. G. y Cox, G. M., *Experimental Designs*, second edition, Nueva York (EE. UU.): John Wiley & Sons, 1957 (traducido por el Centro de Estadística de la Escuela Nacional de Agricultura, *Diseños Experimentales*, sexta reimpresión, México (Méjico): Editorial Trillas, 1980).

[Cohe88] Cohen, J., *Statistical Power Analysis for the Behavioral Sciences*, second edition, New Jersey (EE. UU.): Lawrence Erlbaum Associates, 1988.

[Cook79] Cook, T. D. y Campbell, D. T., *Quasi-Experimentation — Design and Analysis Issues for Field Settings*, Boston (EE. UU.): Houghton Mifflin Company, 1979.

[Cook94] Cook, S. y Daniels, J. D., *Designing Object Systems: Object-Oriented Modelling with Syntropy*, Prentice-Hall, 1994.

[Corr00] Corritore, C. y Wiedenbeck, S., "Direction an scope of comprehension-related activities by procedural and object-oriented programmers: An empirical study", *8th International Workshop on Program Comprehension (IWPC'00)*, 2000.

[Dall00] Dallal, G. E., *The Computer-Aided Analysis of Crossover Studies*, 2000. <www.tufts.edu/~gdallal/crossovr.htm>
[Consulta: febrero de 2001].

[Daly94] Daly, J., Brooks, A., Miller, J., Roper, M. y Wood, M., "Verification of results in software maintenance through external replication", *Proc. of the 1st International Conference on Software Maintenance*, 1994, pp. 50-57.

[Daly96] Daly, J., Brooks, A., Miller, J., Roper, M. y Wood, M., "Evaluation inheritance depth on the maintainability of object-oriented software", *Empirical Software Engineering*, vol. 1, nº 2, 1996, pp. 109-132.

[Davi96] Davis, A., "From the editor", *IEEE Software*, marzo 1996, pp. 4-7.

[Dean99] Dean, A. y Voss, D., *Design and Analysis of Experiments*, Nueva York (EE. UU.): Springer-Verlag, 1999.

[Deli02] Deligiannis, I., Shepperd, M., Webster, M. y Roumeliotis, M, "A review of experimental investigations into object-oriented technology", *Empirical Software Engineering*, vol. 7, nº 3, septiembre 2002, pp. 193-231.

[Erik98] Eriksson, H. y Penker, M., *UML Toolkit*, Nueva York (EE. UU.): John Wiley & Sons, 1998.

[ESER02] ESERNET, *Experimental Software EngineeRing NETwork*, 2002.
<www.esernet.org>
[Consulta: septiembre de 2002].

[Fent97] Fenton, N. E. y Pfleeger, S. L., *Software Metrics. A Rigorous & Practical Approach*, second edition, Londres (RU): International Thomson Computer Press, 1997.

[Figu98] Figueroa, P., *Elementos notacionales de UML*, 1998.
<www.cs.ualberta.ca/~pfiguero/soo/uml/maqcafe.html>
[Consulta: diciembre de 1998].

[Finn98] Finney, K., Rennolls, K. y Fedorec, A., "Measuring the comprehensibility of Z specifications", *Journal of Systems and Software*, vol. 42, nº 1, julio 1998, pp. 3-15.

[Fire93] Firesmith, D. G., *Object-Oriented Requirements Analysis and Logical Design: A Software Engineering Approach*, Nueva York (EE. UU.): Wiley & Sons, 1993.

[Fire98a] Firesmith, D. G., Henderson-Sellers, B., Graham, I. y Page-Jones, M., *Open Modeling Language (OML) Reference Manual*, SIGS Books & Multimedia, 1998.

[Fire98b] Firesmith, D. G., Krutsch, S. A., Stowe, M. y Hendley, G., *Documenting a Complete Java Application Using OPEN*, Nueva York (EE.UU.): Addison Wesley, 1998.

[Fost94] Foster, B. M. y Kirk, R. E., "Sample size determination for Split-plot Factorial Designs", Paper presented at the meeting of the *American Psychological Association*, Los Angeles (CA), agosto 1994.

[Fowl99] Fowler, M., "What´s a model for?", *Distributed Computing*, julio 1999, pp. 33-35.

[Fowl00] Fowler, M. y Scott, K., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, second edition, Massachusetts (EE.UU.): Addison Wesley, 2000.

[Gene02] Genero, M., Miranda, D. y Piattini, M., "Defining and validating metrics for UML statechart diagrams", *6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002)*, Málaga (España), junio 2002.

[Ghez91] Ghezzi, C., Jazayeri, M. y Mandroli, C., *Fundamentals of Software Engineering*, Prentice-Hall, 1991.

[Glas94] Glass, R., "The software research crisis", *IEEE Software*, noviembre 1994, pp. 42-47.

[Glas02] Glass, R., Vessey, I. y Ramesh, V., "Research in software engineering: an analysis of the literature", *Information and Software Technology*, vol. 44, nº 8, junio 2002, pp. 491-506.

[GQM02] GQM, "Aplicación de ayuda al método GQM", 2002. <ivs.cs.uni-magdeburg.de/sw-eng/us/java/GQM/link4.shtml> [Consulta: agosto de 2002].

[Grah95] Graham, I. M., *Migrating to Object Technology*, Wokingham (UK): Addison Wesley, 1995.

[Hahn99] Hahn, J. y Jinwoo, K., "Why are some diagrams easier to work with? Effects of diagrammatic representation on the cognitive integration process of systems analysis and design", *ACM Transactions on Computer-Human Interaction*, vol. 6, nº 3, septiembre 1999, pp. 181-213.

[Hare98] Harel, D. y Politi, M., *Modeling Reactive Systems with Statecharts: The STATEMATE Approach*, Nueva York (EE. UU.): MacGraw-Hill, 1998.

[Harr99] Harrison, R., Badoo, N., Barry, E., Biffl, S., Parra, A., Winter, B. y Wuest, J., "Directions and methodologies for empirical software engineering research", *Empirical Software Engineering*, vol. 4, nº 4, 1999, pp. 405-410.

[Harr00] Harrison, W., "N = 1: An alternative for software engineering research?", *Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research, Workshop at 22nd International Conference on Software Engineering (ICSE'00)*, Limerick (Irlanda), junio 2000.

[Hend94] Henderson-Sellers, B. y Edwards, J.M., *BOOKTWO of Object-Oriented*

Knowledge: The Working Object, Sidney (Australia): Prentice Hall, 1994.

[Hend98] Henderson-Sellers, B., Simons, A. y Younessi, H., *The OPEN Toolbox of Techniques*, Addison Wesley, 1998.

[Hend99a] Henderson-Sellers, B. y Firesmith, D. G., "Comparing OPEN and UML: the two third-generation OO development approaches", *Information and Software Technology*, vol. 41, 1999, pp. 139-156.

[Hend99b] Henderson-Sellers, B., Atkinson, C. y Firesmith, D. G, "Viewing the OML as a variant of the UML", *UML'99 - The Unified Modeling Language. Beyond the Standard*, LNCS 1723, 1999, pp. 49-66.

[Hitz98] Hitz, M. y Kappel, G., "Developing with UML – Some pitfalls and workarounds", *UML'98 – Beyond the notation*, Mulhouse (Francia), septiembre 1998.

[Höst00] Höst, M., Regnell, B. y Wohlin, C., "Using students as subjects — A comparative study of students and professionals in lead-time impact assessment", *Empirical Software Engineering*, vol. 5, nº 3, 2000, pp. 201-214.

[Jaco92] Jacobson, I., *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley, 1992.

[Jaco00] Jacobson, I., Booch, G. y Rumbaugh, J., *El Proceso Unificado de Desarrollo de Software*, Madrid (España): Addison Wesley, Pearson Education, 2000.

[JMP00] JMP, *JMP® Statistics and Graphics Guide*, SAS Institute, 2000.

[Jone89] Jones, B. y Kenward, M. G., *Design and Analysis of Cross-over Trials*, Londres (RU): Chapman and Hall, 1989.

[Judd91] Judd, C. M., Smith, E. R. y Kidder, L. H., *Research Methods in Social Relations*, sixth edition, Orlando (EE.UU): Harcourt Brace Jovanovich, 1991.

[Juri01] Juristo, N. y Moreno, A. M., *Basics of Software Engineering Experimentation*, Massachusetts (EE.UU.): Kluwer Academic Publishers, 2001.

[Kams96] Kamsties, E. y Lott, C. M., "An empirical evaluation of three defect detection techniques", *Proc. of the 5th European Software Engineering Conference*, 1996, pp. 362-383.

[Kim02] Kim, H. y Boldyreff, C., "Developing software metrics applicable to UML models", *6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002)*, Málaga (España), junio 2002.

[Kitc96] Kitchenham, B. A., "Evaluating software engineering methods and tools", parts 1 to 12, *ACM SIGSOFT Software Engineering Notes*, enero 1996 hasta septiembre 1998.

[Kitc02] Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El

Emam, K. y Rosenberg, J., "Preliminary guidelines for empirical research in software engineering", *IEEE Transactions on Software Engineering*, vol. 28, nº 8, agosto 2002, pp. 721-734.

[Kirk95] Kirk, R. E., *Experimental Design: Procedures for the Behavioral Sciences*, third edition, California (EE. UU.): Brooks/Cole Publishing Company, 1995.

[Kish95] Kish, L., *Statistical Design for Research*, Nueva York (EE. UU.): John Wiley & Sons, 1987 (traducido por González, B., *Diseño Estadístico para la Investigación*, Madrid (España): Centro de Investigaciones Sociológicas Siglo XXI, 1995).

[Kueh00] Kuehl, R. O., *Design of Experiments. Statistical Principles of Research Design and Analysis*, California (EE. UU.): Duxbury Thomson Learning, 2000.

[Lagu00] Laguna, M. A., *Sistema de Seguros*, 2000.
<www.infor.uva.es/~mlaguna/is2/is2.html>
[Consulta: enero de 2000].

[Lait97] Laitenberger, O. y DeBaud, J., "Perspective-based reading of code documents at Robert Bosch GmbH", *Information and Software Technology*, vol. 39, nº 11, octubre 1997, pp. 781-791.

[Lait00] Laitenberger, O., Atkinson, C., Schlich, M. y Emam, K. E., "An experimental comparison of reading techniques for defect detection in UML design documents", *Journal of Systems and Software*, vol. 53, nº 2, agosto 2000, pp. 183-204.

[Lind53] Lindquist, E. F., *Design and Analysis of Experiments in Psychology and Education*, Boston (EE. UU.): , Houghton Mifflin, 1953.

[Lore94] Lorenz, M. y Kidd, J., *Object-Oriented Software Metrics: A Practical Guide*, New Jersey (EE. UU.): Prentice-Hall, 1994.

[Marc98] Marchesi, M., "OOA metrics for the Unified Modeling Language", *Proc. of the CSRM'98*, marzo 1998, pp. 67-73.

[Mart98] Martin, R. C., *UML Sequence Diagrams*, Engineering Notebook Column, C++ Report, abril 1998.
<www.objectmentor.com/base.asp?id=40>
[Consulta: diciembre de 1998].

[McGi78] McGill, R., Tukey, J. W. y Larsen, W., "Variations of boxplots", *American Statistics*, 1978, pp. 12-16.

[MÉTR00] MÉTRICA Versión 3: Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información, *Guías de Estructura Principal, de Interfaces y de Técnicas*, Ministerio de Administraciones Públicas, enero 2000.
<www.map.es/csi/metrica3/>
[Consulta: agosto de 2002].

[Mill97] Miller, J., Daly, J., Wood, M., Roper, M. y Brooks, A., "Statistical power and its

subcomponents: Missing and misunderstood concepts in software engineering empirical research", *Information and Software Technology*, vol. 39, nº 4, abril 1997, pp. 285-295.

[Mill00] Miller, J., "Applying meta-analytical procedures to software engineering experiments", *Journal of Systems and Software*, vol. 54, nº 1, septiembre 2000, pp. 29-39.

[Mill92] Milliken, G. A. y Johnson, D. E., *Analysis of Messy Data. Volume I: Designed Experiments*, Nueva York (EE.UU.): Chapman & Hall, 1992.

[NCSS00] NCSS 2000, *NCSS 2000 – User's Guide III*, Utah (EE. UU.): Number Cruncher Statistical Systems, 2000.

[OMG00] OMG, "UML Notation Guide", *UML 1.3 Specification*, marzo 2000.
<www.omg.org/cgi-bin/doc?formal/00-03-01>
[Consulta: marzo de 2000].

[OMG01] OMG, "UML Notation Guide", *UML 1.4 Specification*, septiembre 2001.
<www.omg.org/cgi-bin/doc?formal/01-09-67>
[Consulta: octubre de 2001].

[OSLO02] Universitetet I Oslo, *Experiments at Department of Software Engineering*, Institutt for Informatikk, 2002.
<www.ifi.uio.no/forskning/grupper/isu/forskerbasen/>
[Consulta: agosto de 2002].

[Oter00] Otero, M. C. y Dolado, J. J., "Diseño experimental en ingeniería del software", *Medición para la Gestión en la Ingeniería del Software*, vol. capítulo 3, RA-MA, 2000, pp. 51-72.

[Oter02] Otero, M. C. y Dolado, J. J., "An initial experimental assessment of the dynamic modelling in UML", *Empirical Software Engineering*, vol. 7, nº 1, marzo 2002, pp. 27-47.

[PASS02] PASS 2002, *Power Analysis and Sample Size Software from NCSS*, 2002.
<www.ncss.com/pass.html>
[Consulta: agosto de 2002].

[Pear84] Pearson, P. D. y Tierney, R. J., "On becoming a thoughtful readers: Learning to read like a write", *Eighty- third yearbook of the National Society for the Study of Education: Becoming readers in a complex society*, Chicago: University of Chicago Press, Purves y Niles (eds.), 1984, pp. 114-174.

[Pfle95] Pfleeger, S. L., "Experimental design and analysis in software engineering", *Annals of Software Engineering*, vol. 1, 1995, pp. 219-253.

[Pfle99] Pfleeger, S. L., "Albert Einstein and empirical software engineering", *IEEE Computer*, octubre 1999, pp. 32-37.

[Pfle01] Pfleeger, S. L., *Software Engineering: Theory and Practice*, Prentice-Hall, 2001.

[Pick98] Pickard, L. M., Kitchenham, B. A. y Jones, P. W., "Combining empirical results in software engineering", *Information and Software Technology*, vol. 40, nº 14, diciembre 1998, pp. 811-821.

[Porr99] Porres, I. y Lilius, J., "Digital Sound Record: A case study on designing embedded systems using the UML notation", *TUCS Technical Report 234*, Turku Centre for Computer Science, enero 1999.

[Port95] Porter, A. A., Votta, L. G. y Basili, V. R., "Comparing detection methods for software engineering: A replicated experiment", *IEEE Transactions on Software Engineering*, vol. 21, nº 6, junio 1995, pp. 563-575.

[Prec00] Prechelt, L., "An empirical comparison of seven programming languages", *IEEE Computer*, octubre 2000, pp. 23-29.

[Prec02] Prechelt, L., Unger-Lamprecht, B., Philippsen, M. y Tichy, W. F., "Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance", *IEEE Transactions on Software Engineering*, vol. 28, nº 6, junio 2002, pp. 595-606.

[Purc01] Purchase, H. C., Allder, J. y Carrington, D., "User preference of graph layout aesthetics: A UML study", *Proc. of Graph Drawing: 8th International Symposium GD 2000*, LNCS 1984, 2001, pp. 5-18.

[Ratk93] Ratkowsky, D. A., Evans, M. A. y Alldredge, J. R., *Cross-over Experiments: Design, Analysis and Application*, Nueva York (EE.UU.): Marcel Dekker, 1993.

[Reen95]] Reenskaug, T., Wold, P. y Lehne, O. A., *Working with objects* , Prentice-Hall, 1995.

[Rope97] Roper, M., Wood, M. y Miller, J., "An empirical evaluation of defect detection techniques", *Information and Software Technology*, vol. 39, nº 11, octubre 1997, pp. 763-775.

[Rose99] Rosenberg, D. y Scott, K., *Use Case Driven Object Modeling with UML: A Practical Approach*, Massachusetts (EE.UU): Addison Wesley, 1999.

[Rubi92] Rubin, K. S. y Goldberg, A., "Object behavior analysis", *Communications of the ACM*, vol. 35, nº 9, septiembre 1992.

[Rumb91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. y Lorenson, W., *Object-Oriented Modeling and Design*, Englewood Cliffs, Nueva York (EE. UU.): Prentice-Hall, 1991.

[Rumb00] Rumbaugh, J., Jacobson, I. y Booch, G., *El Lenguaje Unificado de Modelado. Manual de Referencia*, Nueva York (EE.UU.): Addison Wesley, Pearson Education, 2000.

[Scan89] Scanlan, D. A., "Structured flowcharts outperform pseudocode: An experimental comparison", *IEEE Software*, septiembre 1989, pp. 28-36.

[Seli94] Selic, B., Gullekson, G. y Ward, P., *Real-Time Object-Oriented Modelling*, Nueva York (EE. UU.): John Wiley & Sons, 1994.

[Shau97] Shaughnessy, J. J. y Zechneister, E. B., *Research Methods in Psychology*, fourth edition, Nueva York (EE. UU.): McGraw-Hill, 1997.

[Shne77] Shneiderman R. J. et al., "Experimental investigations of the utility of detailed flowcharts in programming", *Communications of the ACM*, junio 1977, pp. 373-381.

[Simo98] Simons, A. J. H. y Graham, I., "37 things that don't work in object modelling with UML", *Proc. 2nd ECOOP Workshop on Precise Behavioural Semantics*, Bruselas (RFA): TU Munich, 1998, pp. 209-232.

[Simo99] Simons, A. J. H. y Graham, I., "30 things that go wrong in object modelling with UML 1.3", capítulo 7 en *Behavioural Specifications of Businesses and Systems*, Kluwer Academic Publishers, 1999, pp. 237-257.
<www.oopsnl.ukc.ac.uk/Issuse35Autumn1998/>
[Consulta: junio de 2000].

[Slon94] Slonim, J., "OO in the real world – Success or Latest Fashion", *Proc. of International Conference of Software Maintenance (ICSM'94)*, septiembre 1994, pp. 440-441.

[Smit02] Smith, R., "Defining the UML kernel", *Software Development Online*, 2002.

[Song01] Song, I. Y., "A heuristic for developing object interaction diagrams", *2001 Informing Science Conference*, Craców (Poland), junio 2001.

[SPSS90] SPSS Inc., *SPSS ™ Reference Guide*, Chicago: SPSS Inc., 1990.

[SPSS97] SPSS Inc, *SPSS ™ Advanced Statistics 7.5*, Chicago: SPSS Inc., 1997.

[SSAD95] SSADM Versión 4.2: Structured Systems Analysis and Systems Design, *SSADM 4+ Reference Manual*, Oxford (RU): NCC Blackwell, 1995.

[Tich98] Tichy, W. F., "Should computer scientists experiment more?", *IEEE Computer*, mayo 1998, pp. 32-40.

[Tori99] Torii, K., Matsumoto, K., Nakakoji, K., Takada, Y., Takada, S., y Shima, K., "Ginger2: An environment for computer-aided empirical software engineering", *IEEE Transactions on Software Engineering*, vol. 25, nº 4, julio/agosto 1999, pp. 474-492.

[Trav99a] Travassos, G. H., *Models and Measures (Product) Part II*, 1999.
<www.cs.umd.edu/class/fall1999/cmsc735/>
[Consulta: septiembre de 1999].

[Trav99b] Travassos, G. H., Shull, F., Fredericks, M. y Basili, V. R., "Detecting defects in object-oriented designs: Using reading techniques to increase software quality", *ACM Sigplan Notices*, vol. 34, nº 10, 1999, pp. 47-56.

[Trav00] Travassos, G. H., *Requirements Document for a Parking Garage Control System*, 2000.
<www.cs.umd.edu/projects/SoftEng/ESEG/manual/OORT_package/OORT/>
[Consulta: marzo de 2000].

[Trav01] Travassos, G. H., Shull, F. y Carver, J., "Working with UML: A software design process based on inspections for the Unified Modeling Language", *Advances in Computers*, vol. 54, 2001.

[Tryg97] Tryggeseth, E., "Report from an experiment: Impact of documentation on maintenance", *Empirical Software Engineering*, vol. 2, nº 2, 1997, pp. 201-207.

[Upch02] Upchurch, R., *Code Reading and Program Comprehension*, Computer and Information Science Department, University of Massachusetts Dartmouth, 2002.
<www2.umassd.edu/SWPI/ProcessBibliography/bib-codereading.html>
[Consulta: agosto de 2002].

[VanS99] Van Solingen, R. y Berghout, E., *The Goal/Question/Metric Method. A Practical Guide for Quality Improvement of Software Development*, Londres (RU): McGraw-Hill, 1999.

[VonM93] Von Mayrhauser, A., "The role of simulation in software engineering experimentation", *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, Lecture Notes in Computer Science 706, Springer-Verlag, 1993, pp. 177-179.

[Wald95] Walden, K. y Nerson, J. M., *Seamless Object-Oriented Software Architecture*, Prentice-Hall, 1995.

[Wirf90] Wirfs-Brock, R., Wilkerson, B. y Weiner, L., *Designing Object Oriented Software*,

Prentice-Hall, 1990.

[Wine91] Winer, B. J., Brown, D. R. y Michels, K. M., *Statistical Principles in Experimental Design*, third edition, Nueva York (EE.UU): McGraw-Hill, 1991.

[Wohl00] Wohlin, C., Runeson, P, Höst, M., Ohlsson, M. C., Regenell, B. y Wesslén, A., E*xperimentation in Software Engineering: An Introduction*, Massachusetts (EE.UU.): Kluwer Academic Publishers, 2000.

[Wohl01] Wohlin, C. y Höst, M., "Special section: Controlled experiments in software engineering", *Information and Software Technology*, vol. 43, nº 15, diciembre 2001, pp. 921-924.

[Yaco99] Yacoub, S., Ammar, H. y Robinson, T., "Dynamic metrics for object-oriented designs", *Proc. of the 6th International Symposium on Software Metrics*, 1999, pp. 50-61.

[Zelk97] Zelkowitz, M. V. y Wallace, D. R., "Experimental validation in software engineering", *Information and Software Technology*, vol. 39, nº 11, octubre 1997, pp. 735-743.

[Zelk98] Zelkowitz, M. V. y Wallace, D. R., "Experimental models for validating technology", *IEEE Computer*, mayo 1998, pp. 23-31.

[Zend01] Zendler, A., Pfeiffer, T., Eicks, M. y Lehner, F., "Experimental comparison of coarse-grained concepts in UML, OML and TOS", *Journal of Systems and Software*, vol. 57, nº 1, abril 2001, pp. 21-30.

ACRÓNIMOS

Glosario de acrónimos

α ..... nivel de significación o error de tipo I

ANOVA ..... *ANalysis Of VAriance*

β ..... error de tipo II

CASE ..... herramientas *Computer-Aided Software Engineering*

CIRT ..... *Class, Instance, Role and Type*

CO ..... diseño *Cross-Over*

COMN ..... *Common Object Modeling Notation*

F ..... estadístico F según la ley de Fisher-Snedecor

GL ..... Grados de Libertad

GLM ..... *General Linear Model*

GQM ..... método *Goal/Question/Metric*

H_A ..... Hipótesis alternativa

H_0 ..... Hipótesis nula

JMP ..... *Statistical Discovery Software*

LSCF ..... diseño *Latin Square Confounded Factorial*

MC ..... Media Cuadrática

NCSS ..... *Number Cruncher Statistical System*

OMG ..... *Object Management Group*

OML ..... *OPEN Modeling Language*

OMT ..... *Object Modeling Technique*

OO ..... Orientado a Objetos

OOAD ..... *Object-Oriented Analysis and Design*

OOSE ..... *Object-Oriented Software Engineering*

OPEN ..... *Object-oriented Process, Environment and Notation*

p ..... probabilidad

PASS ..... programa *Power Analysis and Sample Size*

RUP ..... *Rational Unified Process*

SC ..... Suma de Cuadrados

SPF ..... diseño *Split-Plot Factorial*

SPSS ..... *Statistical Package for Social Sciences*

UML ..... *Unified Modeling Language*