UNIVERSIDAD POLITÉCNICA DE CARTAGENA
Departamento de Tecnologías de la
Información y Comunicaciones

Desarrollo Integral de Sistemas de Procesamiento de Información Visual: Un Enfoque Multiparadigma basado en Líneas de Producto, Componentes y Generación Automática de Software

Tesis Doctoral

Dña. Cristina Vicente Chicote
Ingeniero Informático

Directores
Dr. D. José Carlos Fernández Andrés
Dr. D. Pedro Sánchez Palma

2005

Desarrollo Integral de Sistemas de Procesamiento de Información Visual: Un Enfoque Multiparadigma basado en Líneas de Producto, Componentes y Generación Automática de Software.

Esta memoria se presenta en cumplimiento de los requisitos exigidos para la obtención del grado de Doctor en Telecomunicación y la acreditación de Doctorado Europeo, en el Departamento de Tecnologías de la Información y Comunicaciones de la Universidad Politécnica de Cartagena. El trabajo que aquí se recoge se ha desarrollado durante el periodo comprendido entre octubre de 2001 y abril de 2005 bajo la supervisión de los Doctores D. José Carlos Fernández Andrés y D. Pedro Sánchez Palma.

A mis padres y a Carlos

Agradecimientos

Gracias a mis directores de Tesis, Dr. D. José Carlos Fernández Andrés y Dr. D. Pedro Sánchez Palma por su apoyo y su inestimable ayuda en la supervisión de este trabajo.

Gracias al Grupo de Investigación DSIE (División de Sistemas e Ingeniería Electrónica) por haberme acogido y haberme permitido contar con los mejores medios posibles para llevar a cabo mi investigación.

Gracias al Departamento de Tecnologías de la Información y Comunicaciones por facilitarme la compaginación de mis labores docentes e investigadoras.

Gracias al Centro de Visión por Computador de la Universidad Autónoma de Barcelona, y especialmente al Dr. D. Jordi Vitriá y a la Dra. Dña. Petia Radeva, por la fabulosa experiencia profesional y personal que supuso para mí la estancia de tres meses con ellos durante el 2002.

Gracias al Grupo de Ingeniería del Software Aplicada (ASE) de la Universidad de Reading (Reino Unido), y especialmente a la Dra. Rachel Harrison y al Dr. Daniel Rodríguez, por su cariñosa acogida durante mis cuatro meses de estancia durante el 2004 en los que se terminó de fraguar esta Tesis.

Gracias a todos los que han colaborado en este trabajo de uno u otro modo y en especial a Ana Toledo Moreo, Miguel Pinzolas Prado, Ginés García Mateos, Adolfo J. Martínez Lamberto y Jorge A. Martínez Navarro.

Gracias a mis padres por su ejemplo de buen hacer y su constante apoyo, por enseñarme lo verdaderamente importante, por alentarme siempre y no dejarme desfallecer. Gracias a Carlos por su generosidad y paciencia infinitas y por sacar siempre lo mejor de mí.

Resumen

La amplia variedad de sistemas gestionados actualmente mediante software ha disparado la complejidad de las aplicaciones que es necesario desarrollar. Por otra parte, dada la velocidad con la que se suceden los avances tecnológicos, sobre todo en áreas como la electrónica o las comunicaciones, resulta imprescindible dotar a las aplicaciones de la flexibilidad que precisan para poder asumir, con la suficiente agilidad, los continuos cambios impuestos por el mercado. Además, resulta igualmente esencial que el logro de esta mayor flexibilidad no sea a costa de perjudicar otros requisitos como los relacionados con la eficiencia, el coste o el tiempo de desarrollo de las aplicaciones.

En respuesta a estas nuevas demandas, en las últimas décadas han surgido nuevos paradigmas de desarrollo de software entre los que cabría destacar el paradigma Orientado a Objetos, el Orientado a Aspectos, el Dirigido por Modelos, el Basado en Componentes, el Basado en Líneas de Productos, o los paradigmas de Programación Generativa y Programación Visual. Sin embargo, por diversas razones, estas propuestas no han encontrado una excesiva aceptación en el ámbito del desarrollo de sistemas mixtos hardware/software (Hw/Sw). A ello ha contribuido la falta de consenso respecto de las metodologías y notaciones que se deben emplear, así como la ausencia de herramientas que soporten, de manera integral, todo el ciclo de vida de estos productos. De hecho, en la actualidad, estos sistemas suelen desarrollarse desde una perspectiva muy centrada en el hardware, quedando el software relegado a un segundo plano lo que conduce a la obtención de diseños muy eficientes pero por lo general poco o nada flexibles ni reutilizables.

Este trabajo de Tesis propone una nueva aproximación para el desarrollo de sistemas mixtos Hw/Sw desde una perspectiva más centrada en el software y desde la que se pretende mejorar la flexibilidad y el grado de reutilización de estas aplicaciones sin dejar de lado otros aspectos como la eficiencia, la fiabilidad, o el tiempo de desarrollo. Para ello, dado que en un sistema mixto el software es el componente más maleable, se pretende analizar cómo la incorporación de las nuevas tendencias surgidas en el área de la Ingeniería del Software puede ayudar a mejorar tanto el proceso de construcción de estos sistemas como los resultados que de él se deriven.

Tratando de aprovechar la experiencia acumulada por el Grupo de Investigación en el que se ha llevado a cabo esta Tesis Doctoral en cuanto al desarrollo de varios tipos de sistemas mixtos Hw/Sw y, en particular, de Sistemas de Procesamiento de Información Visual (VIPS), el trabajo que aquí se presenta se centrará en este dominio de aplicaciones, si bien los resultados obtenidos resultan, en cierta medida, directamente extrapolables a otros dominios como el de los sistemas de control en tiempo real o el de los sistemas de telecomunicación.

En la memoria se describen algunos de los trabajos previos realizados en el campo de los VIPS y que han servido de soporte para el desarrollo de esta Tesis. Asimismo, se describe el procedimiento usual para la construcción de estos sistemas y las mejoras que los paradigmas actuales de desarrollo de software pueden aportar a su diseño e implementación. Sobre esta base se propone una nueva metodología de desarrollo de VIPS en la que se incorporan varios de los paradigmas previamente descritos y con la que se persigue resolver, en buena medida, los problemas y limitaciones del procedimiento actual. El trabajo se completa con la descripción de la herramienta IP-CoDER que se ha diseñado para dar soporte a la nueva metodología propuesta y que permite, entre otras cosas, integrar varias de las herramientas y librerías de procesamiento de imágenes actualmente existentes en el mercado.

Summary

The wide variety of today software systems has significantly increased the size and complexity of applications. This, together with the rapid technological advances in some fields such as electronics or communications, makes it necessary to develop software flexible enough to handle the continuous market changes in a time-effective manner. It is equally essential not to achieve this application flexibility improvement by reducing their efficiency or increasing their cost or time to market.

During the last few decades new software development paradigms have arisen in response to these new demands. Among them, Object and Aspect Oriented, Component and Product Line Based Software Development paradigms, together with Generative and Visual Programming are some outstanding examples. However, these proposals have not found a significant acceptance among the community of mixed hardware/software (HW/SW) system developers. This can be partially attributed to the lack of agreement with respect to the methodologies and notations to be adopted as well as to the absence of tools which fully support the whole development life cycle of these applications. In fact, this type of systems are currently developed in a rather hardware-centric fashion while software is attributed a less important role. This approach usually leads to developing highly efficient systems which, on the other hand, are scarcely (if at all) flexible and reusable.

This Thesis presents a novel software-centric approach to mixed HW/SW system development which is aimed at improving this kind of product flexibility and reusability while bearing in mind other requirements regarding efficiency, reliability, low cost, or short time to market. This new approach is built on the foundation of Software Product Lines while leveraging the capabilities offered by Component-Based Software Development and Generative Programming. In order to show the synergy arising from the integration of these three current Software Engineering hot topics, this Thesis present a practical study case on building Visual Information Processing Systems (VIPS), which can be easily extended to cover many other application domains, e.g. real-time control systems or networking applications. In order to support the multi-paradigm approach presented here, the IP-CoDER visual programming tool has been developed which covers the whole VIPS development life-cycle allowing the integration of several image processing libraries and tools currently available.

Índice de contenidos

1. Introducción 1

1.1. Motivación 1
1.2. Descripción del Marco de Trabajo 3
1.2.1. Líneas de trabajo relacionadas con la Tesis 4
1.2.1.1. Desarrollo de Sistemas de Procesamiento de Información Visual 4
1.2.1.2. Líneas de Productos y Arquitecturas Software 6
1.3. Objetivos 7
1.4. Estructura de la Tesis Doctoral 8

2. Sistemas de Procesamiento de Información Visual (VIPS) 11

2.1. Introducción 11
2.1.1. ¿Qué es un VIPS? 13
2.1.2. Disciplinas relacionadas 17
2.1.2.1. Visión Artificial 18
2.1.2.2. Ingeniería de Sistemas y Automática 20
2.1.2.3. Ingeniería de Telecomunicación 21
2.1.2.4. Ingeniería del Software 22
2.2. Clasificación de los VIPS 23
2.2.1. Inspección visual automática 23
2.2.2. Seguridad basada en biométricas 26
2.2.3. Aplicaciones de imagen médica 27
2.2.4. Interfaces preceptuales de usuario 29
2.2.5. Sistemas de ayuda a la navegación 31
2.2.6. Sistemas de información geográfica 32
2.3. Herramientas para procesamiento de imágenes 33
2.3.1. Herramientas de programación SW 33
2.3.1.1. Entornos de programación para el desarrollo de VIPS 34
2.3.1.2. Librerías de procesamiento de imágenes 36
2.3.2. Herramientas para la especificación/programación del HW 37

3. Paradigmas de desarrollo de software 39

3.1. Introducción 39

3.1.1. Requisitos típicamente impuestos a los VIPS 40

3.2. Procedimiento actual de desarrollo de VIPS: problemas asociados 43

3.2.1. Posibles soluciones a las deficiencias detectadas 46

3.3. Paradigmas actuales de desarrollo de software 49

3.3.1. Desarrollo de software basado en componentes (DSBC) 50

3.3.1.1. Definición de Componente Software 51
3.3.1.2. Desarrollo de Software Basado en Componentes: Etapas 53
3.3.1.3. Modelos de Componentes Software comerciales 56
3.3.1.4. Componentes COTS 58

3.3.2. Desarrollo basado en Líneas de Productos Software (LPS) 59

3.3.2.1. Definición del alcance 62
3.3.2.2. Desarrollo del núcleo de recursos reutilizables (core assets) 62
3.3.2.3. Plan de producción de una LPS 64

3.3.3. Programación generativa 64
3.3.4. Programación visual 66
3.3.5. Aportaciones de estos paradigmas a la construcción de VIPS 67

3.4. Una nueva metodología multi-paradigma para el desarrollo de VIPS 69

4. Definición de la Línea LPS-VIPS 73

4.1. Definición de LPS: Trabajos Previos 73

4.1.1. Metodologías seleccionadas para definir la línea LPS-VIPS 74
4.1.2. Esquema del proceso de definición de la línea LPS-VIPS 74

4.2. Definición del alcance de la LPS-VIPS (PuLSE™-Eco) 76

4.2.1. Descripción de las distintas familias de productos VIPS 77
4.2.2. Descripción de los principales dominios o áreas funcionales 85

4.2.2.1. Tabla de características (agrupadas por dominios) 86
4.2.2.2. Descripción de los dominios identificados 88
4.2.2.3. Estructura funcional de los VIPS 91
4.2.2.4. Mapa inicial de productos 92

4.2.3. Valoración y selección de dominios 94

4.2.3.1. Relevancia y aportación de cada dominio a los distintos VIPS 94
4.2.3.2. Potencial de reutilización de cada uno de los dominio 95
4.2.3.3. Selección de dominios y definición final del alcance 96

4.3. Análisis del dominio (PuLSE™-CDA) 99
4.3.1. Modelado de Casos de Uso (CU) 99
4.3.2. Modelado de la variabilidad 104
4.3.3. Modelado de la toma de decisiones 105
4.4. Arquitectura genérica de la línea LPS-VIPS (KobrA) 106
4.5. ¿Cómo crear nuevos VIPS a partir de la definición de la línea LPS-VIPS? 110

5. IP-CoDER: herramienta para desarrollo de prototipos ejecutables de VIPS a partir de componentes software 111

5.1. Introducción 112
5.2. Definición de un nuevo modelo de componentes software para procesamiento de información visual 114
5.2.1. ¿Por qué un modelo de componentes propio? 114
5.2.2. Características que debe ofrecer el modelo de componentes 115
5.2.3. Elementos del modelo de componentes 116
5.2.3.1. Definición del conjunto de tipos 116
5.2.3.2. Componentes atómicos 119
5.2.3.3. Componentes compuestos 122
5.3. IP-CoDER: descripción de la funcionalidad implementada 123
5.3.1. Creación de componentes atómicos 124
5.3.2. Creación de Componentes Compuestos 127
5.3.3. Definición de Constantes 128
5.3.4. Almacenamiento y recuperación de componentes 128
5.3.5. Instanciación de la línea LPS-VIPS 130
5.3.6. Generación de prototipos ejecutables para su validación 130
5.4. Validación de la herramienta mediante un caso de estudio sencillo 131

6. Conclusiones, Aportaciones y Líneas de Trabajo Futuras 137

6.1. Conclusiones 137
6.2. Principales aportaciones 138
6.3. Publicación de resultados y financiación de la investigación 139
6.4. Líneas de trabajo futuras 141

Apéndice I: Descripción de las metodologías PuLSE™ y KobrA..... 143

I.1. PuLSE™ (Product Line Software Engineering™) 144

I.1.1. Fases del ciclo de vida PuLSE™ 144
I.1.2. Componentes técnicos 146
I.1.2.1. PuLSE™ - BC (Baselining and Customization) 146
I.1.2.2. PuLSE™-Eco (Economic Scoping) 146
I.1.2.3. PuLSE™-CDA (Customizable Domain Analysis) 147
I.1.2.4. PuLSE™-DSSA (Domain Specific Software Architecture) 149
I.1.2.5. PuLSE™-I (Instantiation) 149
I.1.2.6. PuLSE™-EM (Evolution and Management) 149
I.1.3. Componentes de soporte 150

I.2. La metodología KobrA 151

I.2.1. Proceso de ingeniería del dominio 152
I.2.1.1. Definición del contexto 153
I.2.1.2. Especificación de los Komponentes 153
I.2.1.3. Definición de los Komponentes 154
I.2.2. Proceso de ingeniería de las aplicaciones 155
I.2.2.1. Instanciación del contexto de la LPS 155
I.2.2.2. Instanciación del framework de la LPS 156

Apéndice II: Definición de la Línea LPS-VIPS: resultados adicionales.... 157

II.1. Descripción de los dominios relevantes (PuLSE™-Eco) 157
II.2. Descripción de Casos de Uso (PuLSE™-CDA) 174
II.3. Diagramas de variabilidad (PuLSE™-CDA) 181
II.4. Modelos de decisión (PuLSE™-CDA) 184

7. Listado de Acrónimos 187

8. Bibliografía 191

Índice de tablas

Tabla 1. Características del proceso visual humano. 12
Tabla 2. Características de los VIP..... 16
Tabla 3. Problemas asociados al procedimiento actual de desarrollo de los VIPS. 46
Tabla 4. Comparativa COM/DCOM, .NET, EJB y CORBA. 58
Tabla 5. Componentes de desarrollo propio frente a componentes COTS. 59
Tabla 6. Mapa inicial de características de las distintas familias de VIPS. 87
Tabla 7. Mapa inicial de productos 93
Tabla 8. Valoración de la relevancia y aportación los dominios a los distintos VIPS..... 94
Tabla 9. Valoración del potencial de reutilización de los distintos dominios. 96
Tabla 10. Resultados de la valoración y selección de dominios..... 98
Tabla 11. Modelo de decisiones asociado al CU *iniciar proceso*..... 106
Tabla 12. Datos especificados por el usuario para crear un componente atómico. ... 126

Índice de figuras

Figura 1. Algunos de los SIVA desarrollados por el DSIE. 5
Figura 2. Sistemas basados en biométricas faciales. 6
Figura 3. Sistema de detección de fecha y hora en imágenes CCTV. 6
Figura 4. Resolución (bits por píxel) vs. tamaño en memoria de las imágenes (KB). 14
Figura 5. Imágenes que ilustran la complejidad del proceso visual humano. 14
Figura 6. Ejemplo de VIPS basado en biométricas faciales. 16
Figura 7. Disciplinas relacionadas con el desarrollo de VIPS. 17
Figura 8. Sistemas de Inspección Visual Automatizados (SIVA) 25
Figura 9. Algunos ejemplos de medidas biométricas. 26
Figura 10. Dispositivos de seguridad biométricos. 27
Figura 11. Segmentación y caracterización de leucocitos en sangre [Angulo 03] 28
Figura 12. Reconstrucción 3D a partir de imágenes médicas [ANATQUEST]. 28
Figura 13. Sistema de cirugía guiado por computador. 29
Figura 14. Ejemplos de Interfaces preceptúales. 30
Figura 15. Ejemplos de sistema de navegación basados en visión *artificial*. 31
Figura 16. Ejemplos de GIS aplicados a cartografía (a) urbana (b) de cultivos. 32
Figura 17. Fases de desarrollo de un VIPS siguiendo el enfoque tradicional. 44
Figura 18. Ciclo de vida (a) en cascada y (b) en espiral. 48
Figura 19. Etapas de los ciclos de vida top-down y DSBC 54
Figura 20. Mecanismos de adaptación de componentes 55
Figura 21. Esquema de los procesos de ingeniería de dominio y de aplicaciones. 62
Figura 22. Distintos niveles de abstracción de una AS. 63
Figura 23. Elementos de la PG. 65
Figura 24. Núcleo de recursos reutilizables de la línea LPS-VIPS. 70
Figura 25. Plantilla para la descripción de las familias de productos. 77
Figura 26. Plantilla para la descripción de dominios. 88
Figura 27. Mapa de estructura de dominios. 91
Figura 28. Criterios de valoración y selección de dominios. 97
Figura 29. Plantilla para la descripción de CU extendida con variabilidad. 100
Figura 30. Extensión de UML para incluir variabilidad en actores y CU. 100
Figura 31. Diagramas de CU de los dominios seleccionados. 101
Figura 32. Diagramas de variabilidad de los CU *arrancar VIPS* e *iniciar proceso*. 105
Figura 33. Diagrama de entidades. 107

Figura 34. Diagramas de Procesos. 108
Figura 35. Diagrama del árbol de Komponentes. 109
Figura 36. Definición de un conjunto de tipos homogéneo. 117
Figura 37. Estructura de la clase gImage. 118
Figura 38. Estructura de la clase gComponent. 120
Figura 39. Estructura de la clase gComponent. 122
Figura 40. Aspecto de la herramienta IP-CoDER. 124
Figura 41. Formulario de definición de componentes atómicos. 125
Figura 42. Relación entre las clases utilizadas por IP-CoDER 125
Figura 43. Código del *wrapper* generado para el componente atómico anterior. 126
Figura 44. Formulario de creación de nueva constante. 128
Figura 45. Formato con el que se almacena un componente atómico. 129
Figura 46. Formato con el que se almacenan los componente compuestos. 130
Figura 47. Diseño del componente que se desea ejecutar. 131
Figura 48. Formulario para ejecutar el diseño mostrado en la Figura 47. 131
Figura 49. Componentes que se emplearán durante el caso de estudio 132
Figura 50. Especificación del prototipo del caso de estudio utilizando IP-CoDER. 133
Figura 51. Componentes internos del componente compuesto SegmentPieles 134
Figura 52. Resultados de la validación del prototipo con distintas imágenes. 135
Figura 53. Vista general de PuLSE 144
Figura 54. Plantilla para la descripción de dominios. 158
Figura 55. Plantilla para la descripción de CU extendida con variabilidad. 174

Capítulo 1

Introducción

Este capítulo recoge las motivaciones que han guiado el desarrollo de esta Tesis Doctoral así como los objetivos planteados al comienzo de la misma. Se realiza una breve descripción del marco de trabajo en el que se ha llevado a cabo esta investigación, haciendo especial hincapié en la experiencia que en el desarrollo de Sistemas de Procesamiento de Información Visual acumula el Grupo de Investigación DSIE (División de Sistemas e Ingeniería Electrónica) de la Universidad Politécnica de Cartagena, en cuyo seno se ha realizado este trabajo. Por último, se describe la estructura de esta memoria, indicando de manera concisa el contenido de cada uno de sus capítulos.

1.1. Motivación

La Ingeniería del Software es una disciplina relativamente reciente[1], sobre todo si se la compara con otras más tradicionales como la Ingeniería Civil, la Naval o incluso la Ingeniería Industrial. Sin embargo, a pesar de su relativa inmadurez, su evolución ha sido y sigue siendo vertiginosa. El desarrollo de software ha pasado de ser una práctica semiartesanal de codificación de programas línea a línea, implementados por lo general bajo demanda y a medida, a convertirse en un proceso mucho más sistemático, soportado por herramientas de alto nivel y metodologías más o menos formales que permiten explotar cualidades como la reutilización, la flexibilidad, o la portabilidad, entre otras muchas. Así, la tendencia actual es a desarrollar productos software estandarizados, fáciles de integrar con los ya existentes y que puedan configurarse de manera sencilla para adaptarse a las necesidades específicas de cada cliente.

[1] El término "Ingeniería del Software" se acuñó en 1968 durante la celebración de la Conferencia de la OTAN en la ciudad de Garmisch.

La amplia variedad de sistemas gestionados actualmente mediante software ha disparado la complejidad de las aplicaciones que es necesario desarrollar. Por otra parte, dada la velocidad con la que se suceden los avances tecnológicos, sobre todo en áreas como la electrónica o las comunicaciones, resulta imprescindible dotar a las aplicaciones de la flexibilidad suficiente para que puedan asumir de manera ágil los continuos cambios impuestos por el mercado (nuevas plataformas hardware cada vez más flexibles, potentes y económicas, entornos distribuidos, cooperativos y on-line, nuevos protocolos de comunicaciones, etc.). Además, resulta igualmente esencial que el logro de esta mayor flexibilidad no sea a costa de perjudicar otros parámetros como la eficiencia, el coste o el tiempo de desarrollo de las aplicaciones.

En respuesta a estas nuevas demandas, en las últimas décadas han surgido nuevos paradigmas de desarrollo de software promovidos, no sólo desde el ámbito científico-académico, sino también fuertemente respaldados por muchas de las empresas líderes del sector informático (Sun Microsystems®, Microsoft®, etc.). Entre estas nuevas propuestas cabe destacar las siguientes:

- Desarrollo de Software Basado en Componentes (CBSD)
- Líneas de Productos Software (SPL)
- Generación Automática de Software (GP)
- Programación Visual (VP)
- Arquitecturas Dirigidas por Modelos (MDA)
- Desarrollo de Software Orientado a Aspectos (AOSD), etc.

A pesar de la revolución que estos nuevos paradigmas han supuesto para la Ingeniería del Software, la falta de consenso en cuanto a las metodologías y notaciones que se deben emplear, así como la ausencia de herramientas que den un soporte integral al desarrollo de aplicaciones siguiendo una o más de estas nuevas propuestas hace que, a fecha de hoy, éstas no resulten tan efectivas en la práctica como cabría esperar.

Además, a la vista de los trabajos publicados sobre casos de estudio desarrollados con éxito siguiendo algunas de estas nuevas tendencias, se observa que son muy pocas las experiencias descritas hasta el momento en el área de informática de sistemas en comparación con las llevadas a cabo en el ámbito de la informática de gestión. Esto puede deberse, entre otras, a las siguientes razones:

- Las aplicaciones de sistemas suelen implementarse como rutinas optimizadas para resolver tareas muy específicas. Su enorme especificidad hace difícil su reutilización, del mismo modo que la necesidad de optimizar su rendimiento limita sensiblemente su grado de flexibilidad.
- Por lo general, el software de sistemas resulta difícil de abstraer de la plataforma hardware para la que se construye y a la que se suele supeditar su desarrollo. Esto explica la actual concepción centrada en el hardware de este tipo de aplicaciones que de hecho, en muchos casos, suelen implementarse como sistemas mixtos con una parte hardware (código que se ejecuta sobre una

plataforma de propósito específico) y otra software (código que se ejecuta sobre un procesador convencional).

- Además, el software que gestiona estos sistemas suele ser de pequeña o mediana envergadura, por lo que en la mayoría de los casos la aplicación de metodologías formales para su desarrollo suele considerarse un proceso más engorroso que útil.
- Por último, y directamente relacionado con lo expuesto en el punto anterior, este tipo de aplicaciones las suelen implementar ingenieros de sistemas, expertos en programación a bajo nivel de dispositivos hardware, pero poco familiarizados con el uso de notaciones y metodologías semi-formales de desarrollo de software, y por lo general, bastante reticentes a incorporarlas.

De acuerdo con todo lo anterior cabe preguntarse cómo y en qué medida puede resultar útil abordar la construcción de este tipo de sistemas mixtos Hardware/Software (HW/SW) desde una nueva óptica que permita mejorar su grado de flexibilidad y reutilización, teniendo en cuenta las restricciones que suelen guiar su construcción (eficiencia, bajo coste, alta fiabilidad, espacio y peso reducidos, corto tiempo de desarrollo, etc.).

Aunque cada vez más se tiende a desarrollar hardware flexible y de propósito general, es sin duda el componente software el más maleable en este tipo de productos. Por ello, este trabajo de tesis pretende analizar cómo la incorporación de las nuevas tendencias surgidas en el área de la Ingeniería del Software puede ayudar a mejorar el proceso y los resultados obtenidos a la hora de construir este tipo de aplicaciones mixtas HW/SW.

En particular, la amplia experiencia del grupo de investigación DSIE (División de Sistemas e Ingeniería Electrónica) en el desarrollo de Sistemas de Procesamiento de Información Visual (en adelante VIPS), ha motivado que este trabajo se centre en la mejora de esta familia de productos mixtos HW/SW. Sin embargo, y como se explicará en el apartado de conclusiones, los resultados obtenidos resultan perfectamente extrapolables a otros dominios de aplicación como los de los sistemas de control o los de telecomunicación.

1.2. Descripción del Marco de Trabajo

Esta Tesis Doctoral se enmarca dentro de las líneas de trabajo del Grupo de Investigación DSIE (División de Sistemas e Ingeniería Electrónica) y del Departamento de Tecnologías de la Información y Comunicaciones de la Universidad Politécnica de Cartagena.

En el DSIE participan investigadores procedentes de distintos Departamentos y Áreas de Conocimiento, lo que confiere al grupo un carácter multidisciplinar que le permite afrontar los nuevos retos de investigación que se le plantean, de manera cooperativa y desde perspectivas distintas y complementarias. Gracias a ello, el DSIE tiene abiertas varias líneas de investigación en áreas tan diversas como la Ingeniería del Software, la Visión Artificial, la Robótica o la Mecatrónica.

La experiencia del DSIE en estos campos es amplia y está avalada por la concesión de varios Proyectos nacionales y europeos, financiados con fondos públicos y en colaboración con varias empresas, así como por numerosas publicaciones en revistas y congresos de ámbito nacional e internacional. Cabe destacar la vocación del DSIE por la transferencia de tecnología al tejido industrial de su entorno, concretada en numerosos contratos firmados al amparo del artículo 11 de la Ley de Reforma Universitaria (LRU), y más recientemente del artículo 83 de la Ley Orgánica de Universidades (LOU).

1.2.1. Líneas de trabajo relacionadas con la Tesis

A continuación se resumen brevemente algunos de los trabajos realizados por el DSIE, tanto en lo relativo al desarrollo de VIPS, como dentro del área de la Ingeniería del Software, dada la relevancia que éstos han tenido para el desarrollo de esta Tesis Doctoral.

1.2.1.1. Desarrollo de Sistemas de Procesamiento de Información Visual

Tal y como se describirá en el capítulo siguiente, existe una amplísima variedad de sistemas agrupados bajo la denominación común de Sistemas de Procesamiento de Información Visual (VIPS). Entre ellos, el DSIE cuenta en la actualidad con varios prototipos totalmente funcionales que resultan de aplicación en sectores tan diversos como la inspección industrial, las telecomunicaciones, o la seguridad. El hecho de contar con todos estos bancos de pruebas proporciona un escenario inmejorable para la realización de este trabajo ya que, de este modo, se podrá demostrar la viabilidad y utilidad de los resultados obtenidos de manera palpable, utilizando entornos reales y no simulados.

- **Sistemas de Inspección Visual Automatizada (SIVA)**

El grupo DSIE ha desarrollado varios prototipos de SIVA para la detección de defectos en piezas mecánicas[2] [Fernández 99], la inspección de productos agroalimentarios[3] [Fernández 01], y la detección de zonas dañadas en cascos de buque para su posterior limpieza[4] [Fernández 05] (ver Figura 1).

- **Interfaces perceptuales basadas en biométricas faciales**

En el campo de los sistemas biométricos y en colaboración con miembros del grupo CoCi (Grupo de Computación Científica) de la Universidad de Murcia, también se han desarrollado dos prototipos de VIPS, uno para el reconocimiento de expresiones

[2] Proyecto SMARTMEC: On-line Quality Control, Production Process Assesment and Tracking System for Mechanical Parts (BRITE - 3743).

[3] Proyecto SIVAFRUT: Sistema de Inspección Visual Automatizada para el Control de Calidad de Gajos de Mandarina (CICYT - 1FD 97-1606-C02-01).

[4] Proyecto EFTCOR: Environmental Friendly and Cost-Effective Technology for Coating Removal. Proyecto V Programa Marco, Subprograma Growth (G3RD-CT-2002-00794). Proyecto complementario: CICYT DPI2002-11583-E.

faciales [García 00] [García 01a] [García 01b] y otro para la detección y seguimiento de caras en secuencias de vídeo [García 02] [Vicente 02], ambos destinados a la construcción de interfaces perceptuales[5]. Actualmente se trabaja en el uso de biométricas faciales para el desarrollo de un entorno virtual de videoconferencia en tiempo real (ver Figura 2).

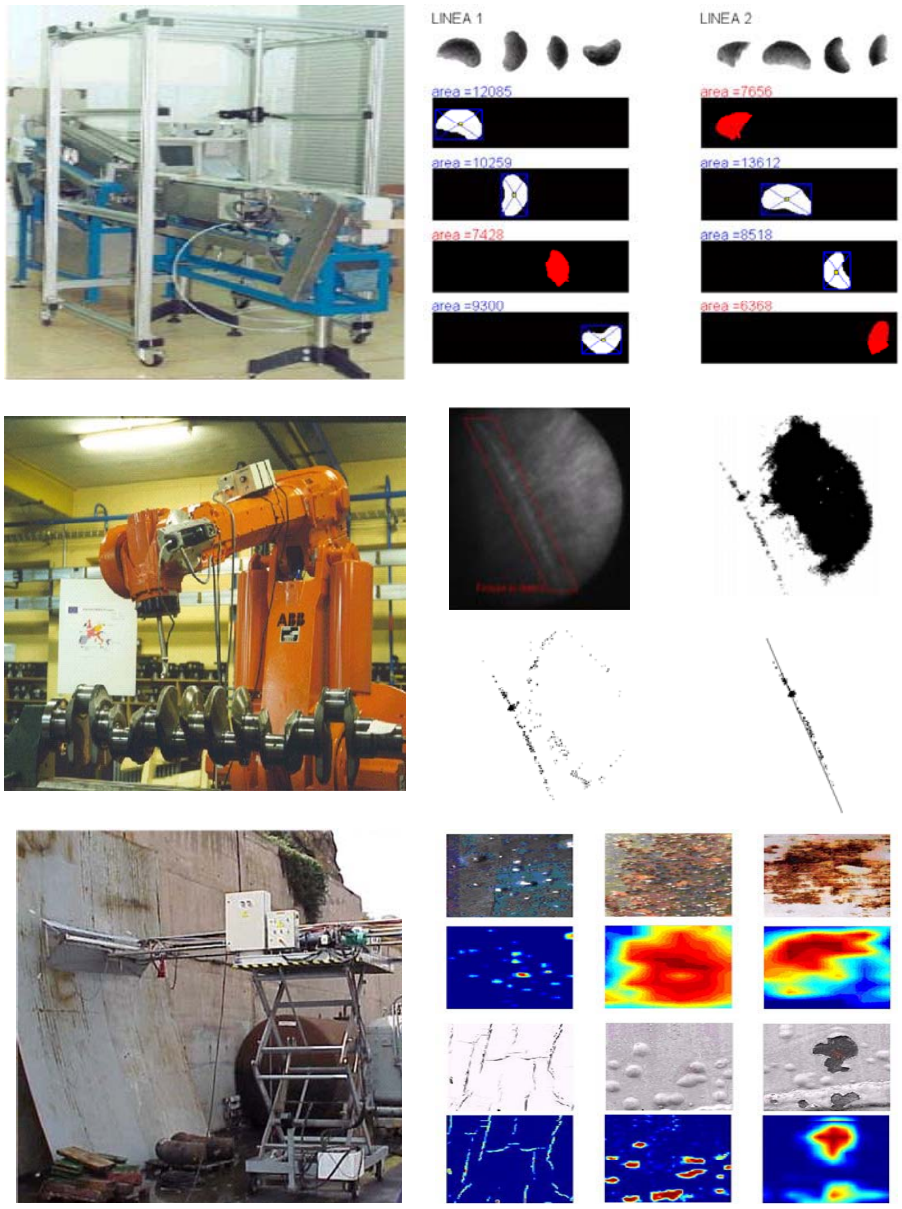


Figura 1. Algunos de los SIVA desarrollados por el DSIE.
A la izquierda prototipos industriales construidos, a la derecha aplicaciones desarrolladas para cada uno de ellos: clasificación de gajos de mandarina (arriba), inspección de cigüeñales (centro), localización de defectos en la superficie de cascos de buques (abajo).

[5] Una interfaz perceptual es aquella que interactúa con el usuario en función de lo que percibe de él (sus gestos, su voz, etc). Por ejemplo, el uso de interfaces perceptuales en aplicaciones de tele-enseñanza permitiría adaptar el contenido y el ritmo de las lecciones según el nivel de atención que se perciba en el alumno analizando, por ejemplo, su expresión facial.

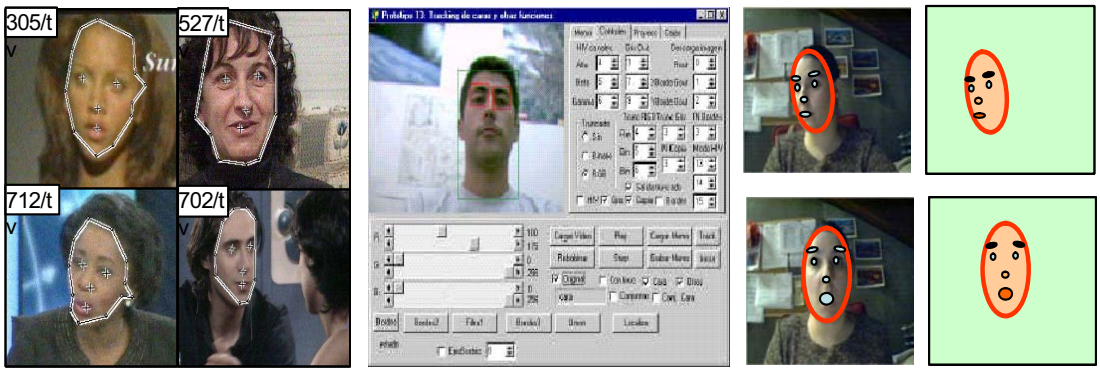


Figura 2. Sistemas basados en biométricas faciales.

Sistema de caracterización de expresiones faciales (izquierda), Sistema de seguimiento de caras en imágenes de vídeo (centro), y entorno virtual de video conferencia en tiempo real (derecha).

➢ Otros VIPS

Por último, en colaboración con la empresa británica Vision Base Ltd., se ha desarrollado un sistema de reconocimiento de fecha y hora en imágenes CCTV (Closed Cirtuit TeleVison), destinada a facilitar la localización de eventos críticos para la seguridad en imágenes de vídeo-vigilancia[6] [García 05a] (Ver Figura 3).

Figura 3. Sistema de detección de fecha y hora en imágenes CCTV.

1.2.1.2. Líneas de Productos y Arquitecturas Software

Como parte del Proyecto CoSIVA[7] se consiguió obtener una descripción de la arquitectura software genérica de los Sistemas de Inspección Visual Automatizada (SIVAs) [Vicente 04a] [Vicente 04b]. Estos resultados, incluidos como una parte de la presente Tesis Doctoral, son la base sobre la que posteriormente se ha descrito una línea de productos para el desarrollo no sólo de SIVAs sino, de manera más general, de

[6] Reconocimiento de Fecha y Hora en Imágenes CCTV (Contrato LOU Art. 83 Ref. 6298). Entidades participantes: grupo CoCI (Universidad de Murcia), grupo DSIE (Universidad Politécnica de Cartagena), Empresa Vision Base Ltd. (Reading, Reino Unido).

[7] Proyecto CoSIVA: Técnicas de Co-Diseño para Sistemas de Inspección Visual Automatizados (TIC 2000-1765-C03-02).

cualquier VIPS tal y como se detalla en capítulos siguientes de esta memoria.

Muy relacionado con este trabajo se ha descrito, como parte del Proyecto ANCLA[8], una arquitectura de referencia para robots de servicio tele-operados [Pastor 04]. En la actualidad, se trabaja en la especificación de una línea de productos software que facilite el desarrollo y mantenimiento de este tipo de sistemas [Álvarez 04] [Ortiz 03] [Ortiz 05] que, como los VIPS, también pueden considerarse mixtos.

A pesar del paralelismo existente entre ambos trabajos, cada uno de los proyectos ha seguido enfoques distintos, lo que en un futuro próximo permitirá, aunando esfuerzos, ofrecer una solución más completa y global al problema de cómo abordar el desarrollo de sistemas HW/SW, independientemente de cuál sea su dominio de aplicación. De este modo, por ejemplo, será posible integrar el enfoque orientado a aspectos incorporado en el Proyecto ANCLA, con el uso de técnicas de generación automática de software y de desarrollo basado en componentes utilizados en esta Tesis Doctoral.

1.3. Objetivos

Tal y como se ha comentado previamente durante la exposición de las motivaciones que han dado lugar al desarrollo de esta Tesis Doctoral (ver 1.1), el procedimiento actual de construcción de aplicaciones mixtas HW/SW y, en particular de los VIPS, presenta una serie de dificultades y limitaciones (aplicaciones eficientes para la resolución de problemas específicos implementadas como soluciones muy dependientes del HW y por lo general poco flexibles y reutilizables).

El objetivo principal que se plantea en el trabajo que aquí se presenta es definir una nueva metodología de desarrollo de VIPS que cubra todo el ciclo de vida de este tipo de productos, potenciando aspectos como la reutilización y la flexibilidad, sin olvidar otros como la eficiencia, el coste o el tiempo de desarrollo.

Para que esta nueva aproximación resulte verdaderamente de utilidad, otro de los objetivos que se plantea en este trabajo es construir una herramienta que soporte de manera integral la metodología propuesta, permitiendo la configuración y evaluación de los distintos productos de manera que se automatice, en la medida de lo posible, su evolución desde los requisitos iniciales hasta su implementación final.

La consecución de estos objetivos se ha abordado por etapas, siguiendo la planificación que se detalla a continuación:

- Realización de un estudio lo más amplio y representativo posible de la gran variedad de VIPS existentes en la actualidad, analizando detenidamente las siguientes cuestiones:
  - ¿Cuál es, de manera general, la estructura y el comportamiento de estos sistemas? ¿Qué tienen todos ellos en común y en qué se diferencian? ¿Es posible definir un patrón de diseño genérico y flexible a partir del que poder desarrollar nuevos VIPS sin tener que partir desde cero cada vez?

[8] Proyecto ANCLA: Arquitecturas diNámiCas para Sistemas de teLeoperAción (CICYT TIC 2003-07804-C05-02).

- ¿Qué proceso se sigue tradicionalmente para su construcción? ¿En qué sentido cabría mejorarlo?
- ¿Qué herramientas ofrece actualmente el mercado para la construcción de VIPS? ¿Qué fases de su ciclo de vida cubren? ¿Resultan éstas compatibles entre sí? ¿Es posible integrar varias de ellas bajo un marco de desarrollo común?

- Estudio de varios de los paradigmas de desarrollo de software actuales analizando su grado de adecuación y su posible contribución a la mejora del proceso de diseño e implementación de los VIPS.
- Definición de una nueva metodología para el desarrollo de VIPS basada en las conclusiones alcanzadas en los estudios previos. Esta metodología deberá:
  - Cubrir las distintas fases del ciclo de vida de estos productos para lo que, como se verá más adelante, será necesario incorporar varios paradigmas de desarrollo de software.
  - Permitir la construcción incremental de nuevos VIPS estableciendo, de manera precisa, cómo evolucionan los productos desde su concepción inicial hasta su implementación final.
  - Maximizar la reutilización de los artefactos software que sea necesario construir: desde los patrones genéricos de diseño, hasta las rutinas de bajo nivel desarrolladas para un hardware específico.
  - Retrasar, en la medida de lo posible, la elección de la plataforma sobre la que finalmente se implementará el sistema.
- Implementación de una herramienta que soporte de manera integral la nueva metodología propuesta evitando, en la medida de lo posible, re-implementar aquella funcionalidad que pueda incorporarse desde alguna otra herramienta ya existentes en el mercado.
- Realización de varios casos de estudio sencillos que permitan validar tanto la nueva metodología de desarrollo de VIPS como la herramienta construida para darle soporte.

1.4. Estructura de la Tesis Doctoral

Esta memoria se ha estructurado en seis capítulos, un apartado de bibliografía y dos apéndices. A continuación se expone de manera resumida el contenido de cada una de estas secciones.

- **Capítulo 1. Introducción.**

  En este capítulo se ha realizado una breve introducción en la que se han expuesto los problemas que se han abordado en el desarrollo de esta Tesis Doctoral. Además, se ha presentado el marco de trabajo en el que se encuadra esta investigación y se han establecido los objetivos de la misma.

- **Capítulo 2. Sistemas de Procesamiento de Información Visual (VIPS): definición, clasificación y herramientas empleadas en su construcción.**

  En este capítulo se define qué es un VIPS y se describen algunas de las disciplinas que suelen estar involucradas en su construcción. A continuación se realiza una clasificación de los VIPS agrupándolos en familias según su ámbito de aplicación. Por último se describen varias de las herramientas comerciales actualmente empleadas para su implementación.

- **Capítulo 3. Paradigmas de desarrollo de software y su aportación al diseño e implementación de VIPS.**

  En este capítulo se define el procedimiento que por lo general suele seguirse en la actualidad para construir VIPS, tratando de identificar los problemas y limitaciones que éste conlleva. A continuación se describen varios de los paradigmas actuales de desarrollo de software, analizando lo que cada uno de ellos puede aportar al proceso de diseño e implementación de los VIPS, así como sus limitaciones. Por último se propone una nueva metodología de tipo espiral, más centrada en el software y que gracias a la integración de varios de los paradigmas antes analizados permite resolver, en buena medida, los problemas y limitaciones del procedimiento actual.

- **Capítulo 4. Definición de la línea LPS-VIPS.**

  En este capítulo se define la línea de familias de productos LPS-VIPS, núcleo central de la nueva metodología para el desarrollo de VIPS propuesta en el capítulo anterior. Para realizar esta definición se han utilizado las metodologías PuLSE™ y KobrA de las que puede encontrarse una descripción detallada en el Apéndice I. El empleo de estas metodologías ha permitido definir el alcance de la línea LPS-VIPS, así como el modelo de análisis y la arquitectura software genérica de los VIPS, elementos esenciales del núcleo de recursos reutilizables de la línea LPS-VIPS.

- **Capítulo 5. IP-CoDER: herramienta para el desarrollo de prototipos ejecutables de VIPS a partir de componentes software.**

  En este capítulo se presenta la herramienta IP-CoDER implementada para dar soporte a la nueva metodología propuesta en el capítulo 3. El diseño de esta herramienta está basado en la definición de un nuevo modelo de componentes software que permite integrar y compatibilizar, de manera eficiente, la funcionalidad ofrecida por varias de librerías de procesamiento de imágenes existentes en el mercado. IP-CoDER implementa este modelo añadiendo a los componentes una dimensión gráfica que hace de ésta una herramienta de programación visual. Gracias a ella, el usuario puede diseñar nuevos VIPS simplemente seleccionando y conectando entre sí distintos componentes previamente creados y almacenados en una librería. A partir de estos diseños IP-CoDER es capaz de generar, de manera totalmente automática, los prototipos

ejecutables correspondientes de modo que el usuario puede validar su funcionamiento utilizando distintos datos de entrada.

- **Capítulo 6. Conclusiones y líneas de trabajo futuras.**

  En este último capítulo se exponen las aportaciones realizadas en esta Tesis Doctoral y los resultados concretos obtenidos. Además se proponen algunas líneas de investigación en las que resultaría interesante profundizar en el futuro.

- **Apéndice I. Descripción de las metodologías PuLSE™ y KobrA.**

  En este apéndice se realiza una introducción a las metodologías PuLSE™ y KobrA, ambas desarrolladas por el Fraunhofer Institute of Experimental Software Engineering (IESE) y que resultan de utilidad para la definición de líneas de productos software. Estas metodologías han sido las empleadas para definir la línea LPS-VIPS tal y como queda recogido en el capítulo 4.

- **Apéndice II. Definición de la Línea LPS-VIPS: resultados adicionales .**

  Este capítulo recoge parte de los resultados de la aplicación de las metodologías PuLSE™ y Kobra empleadas en el capítulo 4 para definir la línea LPS-VIPS. La presentación separada de estos resultados responde simplemente a un intento de hacer el Capítulo 4 más sencillo y ameno de leer.

- **Acrónimos.**

  Este apartado recoge los acrónimos utilizados a lo largo de la Tesis.

- **Bibliografía.**

  Este apartado recoge la bibliografía referenciada en la Tesis.

Capítulo 2

Sistemas de Procesamiento de Información Visual (VIPS)

Definición, clasificación y herramientas

En este capítulo se abordan varias cuestiones relacionadas con los Sistemas de Procesamiento de Información Visual (VIPS). En primer lugar y tras una breve introducción para situar estos sistemas en su contexto, se propone una clasificación en la que se agrupan los VIPS en familias según su ámbito de aplicación. A continuación, se realiza un estudio de varias de las herramientas disponibles en el mercado que resultan de utilidad para el desarrollo de los VIPS.

2.1. Introducción

Como se ha comentado en el capítulo de Introducción, este trabajo se centra en el desarrollo de Sistemas de Procesamiento de Información Visual (VIPS). Por ello parece conveniente, en primer lugar, definir este tipo de sistemas e identificar cuáles son sus características generales así como las disciplinas que suelen estar involucradas en su construcción.

Para comprender mejor qué y cómo son los VIPS puede resultar útil analizar previamente la definición de procesamiento humano de información visual que propone M. Scheiman (ver Definición 1). Esta definición introduce varios aspectos sobre los que merece la pena detenerse brevemente y que de manera resumida quedan recogidos en la Tabla 1.

"El procesamiento de información visual es la capacidad de reconocer, interpretar y recordar aquello que se ve, así como de integrarlo con otros tipos de información sensorial y con la experiencia previa, a fin de poder interactuar de manera adecuada y previsible con el entorno" [Scheiman 94].

Definición 1. Procesamiento de información visual humano.

En primer lugar, el proceso visual humano sólo es posible si se cumplen ciertos requisitos previos; el primero y más evidente es tener la capacidad de ver. Sin embargo, no basta con ver para saber lo que se está viendo; resulta igualmente necesario poder contrastar lo que se ve con ciertos referentes visuales aprendidos y de los que se conoce su significado preciso. Además, la visión no es la única fuente de información (aunque sí una de las más importantes) con las que cuenta el ser humano para poder desenvolverse en la compleja realidad que le rodea. Poder percibir el entorno utilizando varios de los sentidos de manera simultánea resulta tremendamente útil ya que cada uno de ellos suele proporcionar matices distintos y complementarios de una misma realidad. Sin embargo, resulta fácil intuir lo enormemente complejo que es el proceso humano asociado a la percepción ya que se deben filtrar y contrastar las múltiples fuentes de información disponibles, muchas veces útiles y coherentes, pero otras, redundantes, intrascendentes o incluso contradictorias.

Requisitos	Ser capaz de ver
	Contar con experiencia previa (referentes visuales)
	Disponer de otra información sensorial (auditiva, táctil, etc.)
Acciones involucradas en el proceso	Reconocer ➢ asociar lo que se ve con lo que se conoce
	Interpretar ➢ dar sentido a lo que se ve considerando el contexto y la experiencia previa
	Recordar ➢ registrar en la memoria lo que se ve
Finalidad	Interactuar con el entorno de manera adecuada y previsible

Tabla 1. Características del proceso visual humano.

Una vez cumplidos los requisitos anteriores se está en disposición de poder procesar la información visual que se percibe. Para ello, en primer lugar, se realiza un proceso de selección, identificación y reconocimiento de los objetos que aparecen en la escena. Reconocer un objeto implica asociarlo con algún referente visual previo y por lo tanto con el significado que a éste se le atribuye. Sin embargo, un mismo objeto puede tener distintos significados según el contexto en el que aparezca; seleccionar el significado correcto requiere un proceso de interpretación, en el que el contexto espacio-temporal desempeña un papel primordial.

Por último, una vez procesada e interpretada la información visual, ésta deberá guardarse en la memoria para servir como referente en el futuro. Obviamente, recordar absolutamente todo lo que se ve resultaría imposible. Por ello, existen mecanismos que seleccionan y almacenan esta información en distintos tipos de memoria [Ruiz 94]:

- Memoria a corto plazo (icónica): almacena lo que se acaba de ver durante un corto espacio de tiempo (uno a dos minutos). Asimilando el cerebro humano a un ordenador, ésta se correspondería con una memoria de tipo caché.
- Memoria a medio plazo (operativa): almacena los referentes visuales utilizados de manera cotidiana. Siguiendo con el símil anterior, ésta se correspondería con la memoria RAM (*Random Access Memory*) del ordenador.
- Memoria a largo plazo (consolidada): almacena de manera prolongada, incluso para toda la vida, los referentes visuales críticos, bien aprendidos y de los que depende nuestra supervivencia y bienestar. Por seguir con el símil anterior, ésta se correspondería con la memoria secundaria (por ejemplo, un disco duro).

En cuanto a su funcionalidad, el procesamiento de la información visual que percibe el ser humano le permite interactuar con su entorno de manera previsible (de acuerdo con unos referentes visuales sólidos aprendidos desde la experiencia) y adecuada (segura, coherente, gratificante, etc.). El hombre interactúa con el mundo que le rodea de formas muy diversas: comunicándose de forma oral y gestual, manipulando objetos, construyendo artefactos, etc. Para ello, cuenta con sistemas como el locomotor o el fonador, íntimamente ligados con sus sistemas de percepción.

2.1.1. ¿Qué es un VIPS?

Una vez establecido en qué consiste el proceso visual humano podría definirse lo que es un VIPS en los siguientes términos:

> *Un VIPS es un sistema informático capaz de procesar información visual expresada en formato digital y de reproducir, para problemas acotados y bajo ciertas restricciones, los resultados que consigue el sistema visual humano.*

Definición 2. Sistema de Procesamiento de Información Visual (VIPS).

Analizando detenidamente esta definición, inspirada en la anterior de Scheiman (ver Definición 1), es posible deducir algunas de las características y limitaciones principales de los VIPS respecto del sistema visual humano:

- Un VIPS es un sistema informático…

Todo sistema informático tiene un componente hardware (procesador/es, memorias, tarjetas de adquisición de datos, periféricos, etc.) y otro software (sistema operativo, controladores de dispositivo (drivers), aplicaciones de usuario, etc.). En el caso de los VIPS el software suele construirse de manera muy dependiente de la plataforma sobre la que posteriormente se instalará y ejecutará, por lo que estos sistemas suelen considerarse como centrados en el hardware (hardware-centric).

- … capaz de procesar información visual expresada en formato digital …

La información visual es por naturaleza analógica ya que las imágenes se forman por la reflexión de la luz al incidir sobre los objetos de una escena. Sin embargo, los ordenadores, tal y como los conocemos actualmente, sólo pueden procesar información en formato digital. Por ello, para que un VIPS pueda "ver" es necesario

dotarle de los dispositivos adecuados para capturar imágenes en formato digital, o bien para digitalizar los datos adquiridos en formato analógico. En cualquier caso, tanto si la digitalización la realiza el dispositivo de captura (cámara) como si la realiza el propio VIPS, este proceso (muestreo) conlleva una cierta pérdida de información.

En la actualidad existen dispositivos capaces de capturar/digitalizar imágenes de muy alta resolución, de modo que la pérdida de información resulta prácticamente despreciable. Sin embargo, para un tamaño fijo de imagen, cuanto mayor es la resolución de la captura más ocupa la imagen adquirida en la memoria (ver Figura 4). Dado que éste es un recurso limitado y costoso en todos los sistemas informáticos, por lo general resulta necesario alcanzar un compromiso entre la resolución mínima necesaria para ver las imágenes con nitidez y el tamaño máximo de memoria que éstas pueden ocupar en función de las limitaciones del sistema.

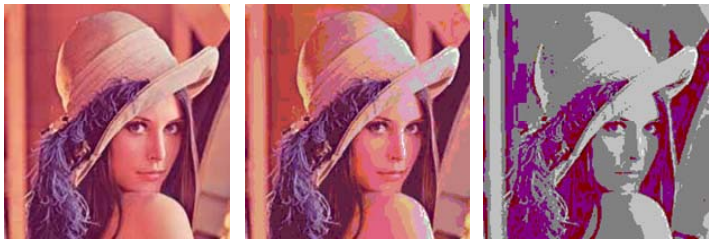


Figura 4. Resolución (bits por píxel) vs. tamaño en memoria de las imágenes (KB).
(a) 24 bits/135 KB, (b) 8 bits/46,1KB, y (c) 4 bits/22,6 KB

El sistema visual humano es enormemente complejo, tanto por su estructura como por las tareas que lleva a cabo y, en consecuencia, resulta muy difícil de imitar. Tareas tan sencillas para cualquier persona como identificar a alguien conocido, seguir un objeto móvil con la mirada, o reconocer un objeto incompleto o parcialmente ocluido (ver Figura 5) pueden resultar tremendamente difíciles de programar en un VIPS, dada la complejidad de los procesos involucrados en la visión humana (algunos inconscientes, otros incluso desconocidos [Vergés 02]). Sólo algunos modelos computacionales de inspiración biológica, como los basados en redes neuronales artificiales [Rosenblatt 62], tratan de reproducir más o menos fielmente estos procesos. Otros, sin embargo, prefieren considerar las imágenes como señales bidimensionales que procesar aplicando distintos tipos de filtros, o como matrices de datos sobre las que realizar ciertas operaciones lógicas, aritméticas, estadísticas, etc.

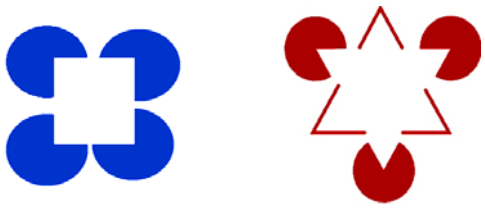


Figura 5. Imágenes que ilustran la complejidad del proceso visual humano.
En ambos dibujos aparecen figuras implícitas que el ser humano es capaz de reconocer utilizando mecanismos inconscientes para unir líneas y completar contornos.

- … reproduciendo los resultados del sistema visual humano …

Un VIPS puede reproducir, con un cierto grado de precisión, los resultados del sistema visual humano, siempre que cuente con información lo suficientemente precisa y completa acerca del problema que trata de resolver y que se le haya programado con la lógica adecuada para extraer, procesar e interpretar la información que se le suministra.

Si bien es cierto que los VIPS adolecen de muchas limitaciones respecto del sistema visual humano, también lo es que tienen ciertas ventajas respecto de éste que le permiten, para problemas concretos, obtener mejores resultados. Por ejemplo, en la visión humana desempeñan un papel determinante factores como la atención, el cansancio, la subjetividad, o los defectos visuales congénitos o adquiridos. Sin embargo los VIPS son capaces de repetir las tareas para las que han sido programados de manera objetiva, continua y sin cansarse o descentrarse durante largos períodos de tiempo. Así, por ejemplo, en ciertas tareas repetitivas que requieren un elevado grado de atención como son la catalogación de muestras, el conteo de objetos o la detección de defectos en ciertos productos, los VIPS suelen exhibir un rendimiento superior al de los operadores humanos. Sin embargo, en el caso de tareas en las que se requiere un cierto grado de subjetividad como, por ejemplo, detectar personas que se comportan de manera sospechosa, queda lejos aún el momento en el que los VIPS puedan conseguir resultados equiparables a los obtenidos por las personas.

- … para problemas bien acotados y bajo ciertas restricciones...

Como ya se ha comentado con anterioridad, los VIPS tienen una capacidad de memoria y de procesamiento limitadas. Si bien es cierto que la tecnología avanza a pasos de gigante ofreciendo día a día sistemas cada vez más potentes y asequibles, la ingente cantidad de datos contenidos en cada imagen y la complejidad de los algoritmos necesarios para procesarlos conduce inexorablemente a la necesidad de desarrollar VIPS altamente especializados. Así, a diferencia del sistema visual humano que es capaz de desenvolverse en un entorno altamente cambiante y bajo circunstancias muy diversas, los VIPS sólo son capaces de resolver problemas muy concretos en entornos relativamente simplificados.

La Tabla 2 recoge, de manera resumida, las características principales de los VIPS. En la Figura 6 se muestra un ejemplo de VIPS en el que se ponen de manifiesto algunas de estas características.

Requisitos	Contar con los dispositivos adecuados para obtener imágenes del mundo real en formato digital, o bien tener acceso a ellas desde un fichero, ya sea local o remoto.
	Conocer qué características o patrones visuales resultan de interés para cada aplicación concreta y cuáles son sus valores de referencia (mínimo, máximo, media, etc.).
	Contar con los dispositivos adecuados para obtener otros tipos de información sensorial en formato digital, así como con los mecanismos que permitan sincronizar los datos procedentes de las diversas fuentes[9].
Acciones involucradas en el proceso	Preprocesamiento ➢ Adecuar en la medida de lo posible la imagen de entrada (eliminar ruido, mejorar el brillo, el contraste, el color, etc.).
	Procesamiento ➢ Detectar, segmentar y caracterizar las zonas de interés en la imagen.
	Clasificación ➢ Determinar en qué medida se ajustan los resultados obtenidos para cada característica a los valores establecidos como referencia (reconocimiento de patrones).
	Interpretación ➢ Asociar los resultados de la clasificación con un significado válido en el contexto de la aplicación.
	Almacenar/Mostrar/Transferir los resultados.
Finalidad	Interactuar con un entorno acotado y que por lo general representa una simplificación de la realidad de manera previsible y adecuada (precisa, eficiente, robusta, etc.).

Tabla 2. Características de los VIP.

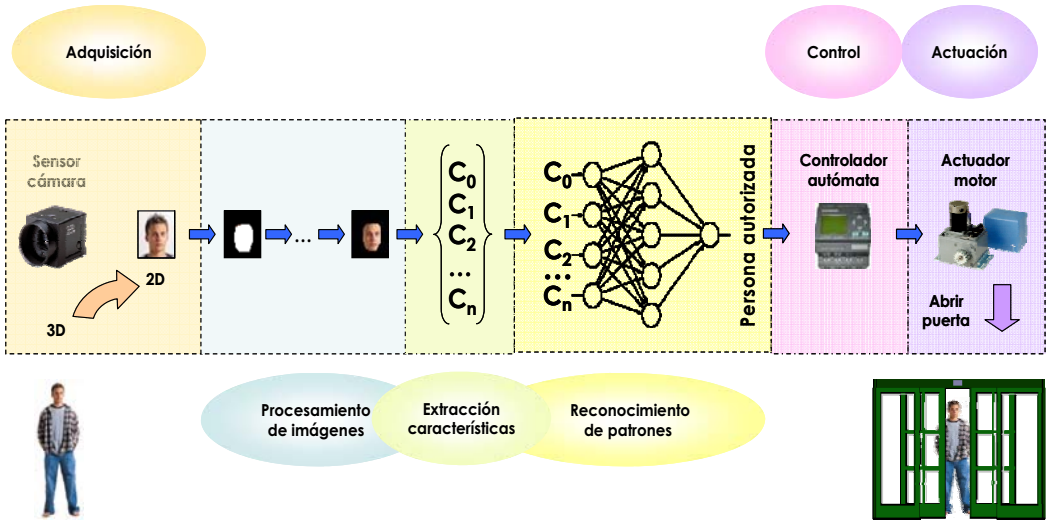


Figura 6. Ejemplo de VIPS basado en biométricas faciales.
Se muestran los procesos que típicamente lleva a cabo un VIPS adquisición y procesamiento de imágenes, extracción de características, reconocimiento de patrones, control, y actuación.

[9] El procesamiento de imágenes es de por sí una tarea computacionalmente muy costosa. Por ello, el procesamiento simultáneo de otros tipos de información sensorial no suele ser viable, salvo que se trate de datos muy simples y poco costosos de procesar (p. e. una señal digital de presencia/ausencia), o que sea estrictamente necesario para una aplicación concreta.

2.1.2. Disciplinas relacionadas

A continuación se realiza un breve repaso de las disciplinas que tradicionalmente están involucradas en el desarrollo de los VIPS (ver Figura 7), haciendo especial hincapié en campos como la Visión Artificial y otros afines como el Procesamiento de Imágenes o el Reconocimiento de Patrones, dada su especial relevancia a la hora de construir estos sistemas.

Figura 7. Disciplinas relacionadas con el desarrollo de VIPS.

Cada una de las disciplinas que se analizan a continuación está directamente relacionada con alguna de las características de los VIPS que se acaban de describir.

- Percepción del entorno ➢ Sensorización (p. e. adquisición de imágenes).
- Procesamiento de información visual ➢ Visión artificial / Procesamiento de imágenes / Reconocimiento de patrones/ Tratamiento digital de la señal.
- Interacción con el entorno ➢ Control / Actuación / Comunicación.

La Ingeniería del Software se ha incluido en esta lista ya que, como se demostrará a lo largo de este trabajo, su aportación al diseño e implementación de los VIPS resulta esencial a la hora de aumentar el grado de flexibilidad, modularidad y reutilización de estos sistemas, cualidades de las que actualmente carecen. Dado que tradicionalmente la Ingeniería del Software ha sido ajena al desarrollo de este tipo de productos, la incorporación de esta disciplina como eje central de una nueva metodología de desarrollo de VIPS que se presentará en el capítulo siguiente, supone una de las principales aportaciones de esta Tesis.

2.1.2.1. Visión Artificial

La Visión Artificial, también conocida como Visión por Computador (del inglés *Computer Vision*), tiene sus orígenes en la Inteligencia Artificial, aunque en la actualidad se la considera una disciplina con entidad propia, dado el enorme progreso que ha experimentado en las últimas décadas.

En la literatura es posible encontrar numerosas definiciones del término *Visión Artificial* [Nalwa 93] [Trucco 98] [Shapiro 01] [Forsyth 02]. Salvo por pequeños matices, todas ellas coinciden esencialmente en lo siguiente:

> *La Visión Artificial comprende un conjunto de técnicas computacionales destinadas a descubrir la estructura y las propiedades del mundo tridimensional y dinámico que nos rodea, a partir de una o varias imágenes bidimensionales obtenidas de él.*

Definición 3. Visión Artificial.

Entre las propiedades que la Visión Artificial trata de descubrir acerca del mundo para reconocerlo, interpretarlo o reconstruirlo, se pueden incluir las siguientes:

- Propiedades de los objetos que aparecen en la imagen, relacionadas con:
  - su geometría: forma, tamaño, posición, etc.
  - su movimiento: dirección, velocidad, aceleración, etc.
  - el material del que están hechos: color, textura, brillo, etc.
- Propiedades globales de la escena recogida en una imagen:
  - Iluminación: color, intensidad, dirección, etc. de las luces.
  - Movimiento: cambio de posición del observador (cámara que captura las imágenes).

El problema que aborda la Visión Artificial es de una complejidad comparable a la de resolver un sistema indeterminado (con más incógnitas que ecuaciones). Existen diversos motivos que pueden dar lugar a situaciones de indeterminación o ambigüedad visual: por ejemplo, distintas escenas reales en las que aparecen objetos diferentes, pueden producir imágenes idénticas. A esto hay que añadirle la ambigüedad que introduce el hecho de tratar de analizar un mundo tridimensional a través de imágenes bidimensionales (reducción de la información). Para hacer frente a estas indeterminaciones la visión artificial recurre a dos estrategias: (1) obtener más información, esto es, más imágenes de la escena que se está analizando y (2) asumir ciertas simplificaciones (procurando que sean lo más realistas posible) a fin de reducir la complejidad del problema.

Sin embargo, aún cuando es posible aplicar alguna de estas estrategias (no siempre resulta viable), solventar los problemas que se plantea la Visión Artificial de manera adecuada, esto es, en un tiempo razonable y con la suficiente precisión y robustez, no resulta en absoluto sencillo. De hecho, a pesar de los grandes avances realizados en este campo en las últimas décadas, aún son muchos los problemas sin resolver o para los que no se ha desarrollado una solución totalmente satisfactoria.

El enorme interés que despierta esta disciplina, tanto en la comunidad científica como en el ámbito empresarial, resulta evidente a la vista del ingente volumen de publicaciones y conferencias especializadas en esta materia, así como por la aparición en el mercado de nuevos productos que incorporan los últimos avances en este campo (auto-calibración de cámaras digitales, guiado de robots autónomos, construcción de entornos virtuales, etc.).

➢ Disciplinas Relacionadas con la Visión Artificial

Tradicionalmente, la Visión Artificial ha estado muy ligada a otras disciplinas como la Psicología, la Neurociencia, la Robótica, el Procesamiento de Imágenes, o el Reconocimiento de Patrones [Trucco 98]. De hecho, las fronteras que delimitan estas áreas de conocimiento son todavía hoy relativamente difusas, existiendo un cierto solapamiento entre todas ellas. A continuación se introducen brevemente dos de estas disciplinas: el Procesamiento de Imágenes y el Reconocimiento de Patrones, dada su estrecha vinculación con este trabajo y su importante contribución al campo de la Visión Artificial.

Procesamiento de imágenes

El procesamiento de imágenes, como la visión artificial, es un área muy extensa y enormemente activa en la actualidad. Procesar una imagen implica extraer información de ella, o transformarla en otra "mejor" en algún sentido (por ejemplo, más nítida, más fácil de caracterizar, o que ocupa menos en memoria).

Como se acaba de ver la visión artificial trata de modelar el mundo tridimensional que nos rodea a partir de imágenes que, en su inmensa mayoría, necesitan un cierto procesamiento previo para resultar de utilidad [Trucco 98]. Esto explica el considerable solapamiento existente entre ambas disciplinas y el hecho de que algunos autores incluyan el procesamiento de imágenes como parte de la visión artificial.

Entre los problemas que típicamente aborda el procesamiento de imágenes, cabe destacar los siguientes:

- Los relativos a la restauración y el realzado de imágenes para mejorar su calidad [González 02]: eliminación de ruido, mejora del contraste, el brillo, el enfoque, etc.
- Los relacionados con la extracción de características, bien de la imagen en su conjunto o de los objetos contenidos en ella [González 02]: color, forma, tamaño, textura, etc.
- Los relativos a la compresión de imágenes: reducción del tamaño que ocupan las imágenes en memoria: estándares JPEG [JPEG] y MPEG [MPEG].

Para abordar estos problemas existe una gran variedad de algoritmos de procesamiento de imágenes de los que a continuación se enumeran sólo algunos:

- Composición de imágenes aplicando operadores lógicos (and, or, xor, etc.) o aritméticos (suma, resta, etc.).
- Transformaciones de distintos tipos: geométricas (reducción/aumento de tamaño, rotación), morfológicas (apertura/cierre, erosión/dilatación), de un espacio de color en otro (RGB, HSI, YCrCb), etc.

- Filtrados de distintos tipos: espacial (por ejemplo, mediante la convolución de máscaras), temporal (por ejemplo, utilizando filtros de Kalman), en frecuencia (por ejemplo, mediante una transformada de Fourier).
- Extracción y caracterización de contornos y regiones, etc.

Reconocimiento de Patrones

El Reconocimiento de Patrones consiste en extraer la información relevante contenida en los datos de entrada (imágenes en este caso) para, comparando esta información con unos patrones preestablecidos, poder tomar una decisión o realizar una clasificación de los datos originales [Duda 00].

Nótese que en esta definición se incluye como primera etapa del reconocimiento de patrones la extracción de características, también incluida como parte del procesamiento de imágenes. Esto demuestra lo difícil que resulta a veces discernir dónde están los límites entre estas disciplinas (ver Figura 6). Algunos autores prefieren considerar el procesamiento de imágenes como un conjunto de transformaciones imagen-imagen de modo que la extracción de características queda claramente fuera de esta disciplina. Otros autores, sin embargo, consideran que la extracción de características y el procesamiento de imágenes están tan íntimamente ligados que resultan inseparables.

En cualquier caso, la selección del conjunto de características que se deben extraer de una imagen, o más concretamente de los objetos que aparecen en ella, resulta clave para lograr clasificarlos correctamente. Así, debe escogerse un conjunto suficiente de características (ni tantas que resulte excesivamente costoso calcularlas, ni tan pocas que no aporten la suficiente información para distinguir a unos de otros), tratando de que éstas sean lo más discriminantes que sea posible[10].

Existen distintas aproximaciones al problema de cómo clasificar o reconocer un objeto (expresado como un conjunto o vector de características) a partir de una serie de patrones preestablecidos (vectores que recogen los valores típicos de las características para cada clase de objetos). Entre ellos cabe destacar el enfoque estadístico, el sintáctico, el lógico-combinatorio, o el basado en redes neuronales (RRNN) [Duda 00].

2.1.2.2. Ingeniería de Sistemas y Automática

Según la definición del ISA (Instrument Society of America), "*la automatización comprende el diseño, desarrollo y aplicación de dispositivos y sistemas capaces de percibir el entorno y controlar los procesos y las operaciones que se llevan a cabo en él*". De esta definición se desprende que, de manera general, los sistemas automáticos deben contar al menos con los siguientes elementos:

- Sensores: son los encargados de proporcionar al sistema información acerca del entorno. Existen infinidad de tipos: cámaras, sensores de paso, de temperatura, de presión, etc.

[10] Una característica es tanto más discriminante cuanto más difieren los valores que ésta toma para las distintas clases de objetos. Por ejemplo, para separar coches de camiones, el tamaño resulta mucho más discriminante que el color.

- Sistema de control: es el responsable de (1) obtener y procesar los datos de los sensores; (2) en función de estos datos, decidir qué acción debe realizarse a continuación; (3) enviar las órdenes correspondientes a los actuadores. Para poder llevar a cabo todas estas tareas, el sistema de control debe tener una cierta capacidad de memoria, de procesamiento y de comunicación con el entorno. Así, esta labor puede llevarla a cabo un autómata, un PC industrial, o incluso un ordenador convencional cuando las condiciones del entorno y los requisitos del sistema así lo permitan.
- Actuadores: son los encargados de interactuar con el entorno siguiendo las órdenes que reciben del sistema de control. Existen distintos tipos de actuadores: brazos robóticas, motores (por ejemplo, para abrir una puerta), sopladores neumáticos, elevadores hidráulicos, etc.
- Interfaz hombre-máquina: permite a los usuarios interactuar con el sistema de manera activa (p. e. modificando alguno de los parámetros del sistema de control a través de un teclado), o pasiva (p. e. observando los datos del proceso en una pantalla).

El término automatización, por lo general referido a procesos industriales (automatización industrial), persigue reducir costes de mano de obra, mejorando a la vez aspectos como la productividad o la calidad final de los productos. Aunque las ventajas de la automatización resultan evidentes, también existen algunos inconvenientes asociados a ella: elevado coste de puesta en marcha, costes derivados del mantenimiento, poca flexibilidad para adecuarse a los cambios del mercado, etc.

Aunque algunos procesos industriales ya se habían automatizado a pequeña escala con anterioridad, no es hasta mediados del siglo XX, con la llegada al mercado de los primeros ordenadores lo suficientemente potentes y asequibles, cuando se puede hablar de líneas de producción masivamente automatizadas. Sin embargo, una de las pocas tareas que más se resiste a la automatización es precisamente la inspección visual de la producción, como se comentará más adelante al hablar de los Sistemas de Inspección Visual Automatizados (SIVA).

2.1.2.3. Ingeniería de Telecomunicación

La Ingeniería de Telecomunicación se encarga de los procesos involucrados en la transmisión de información a distancia, entre los que pueden incluirse la conversión y/o codificación de datos en el momento de la transmisión y/o de la recepción, la recuperación de las posibles pérdidas de información ocasionadas por la existencia de imperfecciones o de ruido en el medio, etc.

En sus comienzos las telecomunicaciones eran predominantemente analógicas pero la rápida proliferación de los dispositivos digitales y ópticos ha dado lugar a lo que se ha dado en llamar la nueva era de las Tecnologías de la Información y las Comunicaciones (TIC). En este contexto surgen dos disciplinas muy relacionadas con el desarrollo de los VIPS:

- En el área de las telecomunicaciones aplicadas a la informática surge la Telemática como disciplina responsable del desarrollo de nuevos medios de transmisión como la fibra óptica o el más reciente PLC (*Power Line Communication*), nuevos protocolos de comunicaciones, nuevos diseños de redes intranet e internet, nuevos servicios como los de control remoto de dispositivos (satélites, robots industriales, etc.) o los de telefonía IP (VoIP, *Voice over IP*), etc.
- Dentro del área del diseño de circuitos integrados con aplicación a las telecomunicaciones surge como una nueva disciplina el Procesamiento Digital de las Señales y dentro de ella, como una especialidad, el Procesamiento de Imágenes Digitales (imágenes vistas como señales bidimensionales), íntimamente ligado con el campo del procesamiento de imágenes tradicional (ver 2.1.2.1) y por lo tanto de aplicación directa a los VIPS.

De un modo u otro todos los VIPS hacen uso de alguna de estas disciplinas, bien durante el proceso de tratamiento de la información visual, bien cuando necesitan comunicarse con sistemas externos de los que reciben o a los que envían datos.

2.1.2.4. Ingeniería del Software

El término "Ingeniería del Software" fue utilizado por primera vez en 1968 durante la celebración de la conferencia de la OTAN en la ciudad alemana de Garmisch. Desde entonces con él se designa al conjunto de métodos, técnicas y herramientas que gestionan el proceso integral de desarrollo del software y que suministran las bases para construir programas eficientes y de calidad, siempre dentro de unos plazos y con unos costes acotados.

La Ingeniería del Software puede considerarse una materia multidisciplinar en la que convergen otras como las matemáticas, la lingüística, la arquitectura, otras ingenierías como la electrónica o la industrial, etc. Es precisamente este carácter multidisciplinar el que permite aplicar las técnicas y metodologías propias de la Ingeniería del Software para diseñar e implementar una amplia variedad de productos que van desde aplicaciones de gestión de bases de datos a sistemas operativos, aplicaciones de diseño asistido por ordenador, sistemas de videoconferencia, etc.

Sin embargo, como ya se comentó en el capítulo de Introducción, no es común emplear un enfoque de Ingeniería del Software cuando se trata de implementar sistemas en los que, como ocurre en el caso de los VIPS, el componente hardware desempeña un papel central mientras que el software, aunque no menos importante, debe subordinarse al primero para tratar de aprovechar al máximo su rendimiento. Para ello, el componente software de estos sistemas suele programarse empleando lenguajes de muy bajo nivel, a veces específicos para una determinada plataforma hardware, lo que conduce a desarrollos eficientes pero poco flexibles y prácticamente nunca reutilizables.

En este contexto, el trabajo que aquí se presenta pretende demostrar cómo el empleo de un enfoque de Ingeniería del Software permite eludir éstas y otras de las muchas limitaciones derivadas del procedimiento actual de construcción de los VIPS (muy centrado en el hardware). De hecho, una de las principales aportaciones de esta Tesis consiste en la definición de una nueva metodología de desarrollo de VIPS

centrada en el software y que se basa la integración de varios de los paradigmas de desarrollo de software más ampliamente aceptados y difundidos en la actualidad. Como se verá más detalladamente en el capítulo siguiente, esta metodología permitirá extraer y explotar los muchos puntos que tienen en común todos estos sistemas al pertenecer a un mismo dominio de aplicación, a la vez que permitirá recoger y modelar de manera sistemática sus diferencias. De este modo, a partir de los rasgos identificados como comunes a todos los VIPS podrá desarrollarse un núcleo de recursos reutilizables que serán válidos, de manera general y salvo adaptaciones puntuales, a la hora de construir cualquier nuevo sistema de este tipo. Como resulta evidente, esto redundará en una reducción del tiempo de salida al mercado de estos productos a la vez que aumentará su flexibilidad y grado de reutilización.

2.2. Clasificación de los VIPS

Los VIPS pueden clasificarse atendiendo a distintos criterios como por ejemplo, la complejidad computacional de la tarea visual que resuelven, el nivel de eficiencia, precisión, o seguridad que requieren, su tamaño, coste, etc. Sin embargo, lo más común es clasificar los VIPS atendiendo a su funcionalidad, esto es, al tipo de aplicación al que se destinan. Según este criterio es posible identificar varias categorías o familias de VIPS, entre las que a continuación destacaremos las siguientes: sistemas de inspección visual automática, sistemas de seguridad basados en biométricas, aplicaciones de imagen médica, sistemas de ayuda a la navegación, sistemas de teledetección e interfaces perceptuales.

2.2.1. Inspección visual automática

La automatización de muchos de los procesos de fabricación, como el ensamblaje mecánico de piezas, el transporte de mercancías, o la manipulación de productos peligrosos, entre otros, ha constituido una verdadera revolución industrial. Sin embargo, en la mayoría de lo casos, el proceso de inspección visual de la producción sigue siendo realizado hoy en día por numeroso personal especializado. Obviamente, mantener a todo este personal incrementa los costes de producción hasta el punto de que muchas industrias, consideradas tecnológicamente muy avanzadas, han tenido que establecer sus factorías en países en los que el menor coste de la mano de obra les permitiera seguir siendo competitivas.

Para cualquier inspector cualificado, la detección de taras o la catalogación de los productos en función de su forma, tamaño, color o textura, resultan labores sencillas y rutinarias. Por lo general, la complejidad de estas tareas reside en conseguir mantener un alto nivel de atención visual de manera prolongada y proceder de manera objetiva a la hora de decidir sobre la calidad de los productos. Por el contrario, los Sistemas de Inspección Visual Automáticos o Automatizados, comúnmente denominados SIVA, pueden mantener de manera incansable toda su atención sobre la tarea que deben desarrollar y reproducir siempre, de manera objetiva y constante, los criterios de selección o catalogación para los que han sido entrenados [Newman 95].

La inspección visual automática se define como un proceso automatizado de control de calidad que permite determinar, gracias al empleo de técnicas de procesamiento de imágenes y reconocimiento de patrones, en qué medida se ajusta un determinado producto a las especificaciones dadas para su fabricación [Chin 82].

Definición 4. Inspección Visual Automática.

La inspección visual automática, que se acaba de definir, trata de maximizar simultáneamente tanto la eficacia como la eficiencia del proceso de inspección, entendiendo ambas cualidades como se indica a continuación:

- Eficacia: De acuerdo con las especificaciones de la producción se debe detectar, en la medida de lo posible, el total de productos defectuosos, rechazando simultáneamente la menor cantidad de productos en buen estado. Para ello resulta crucial:
  - Identificar qué características visuales del producto resultan más discriminantes, esto es, qué distingue a los productos en buen estado de los que no lo están (discriminación positiva), o viceversa, qué características determinan la existencia de un defecto (discriminación negativa).
  - Seleccionar los algoritmos de procesamiento de imágenes y/o de reconocimiento de patrones que permiten extraer estas características discriminantes, así como los algoritmos de clasificación que permiten catalogar cada producto como bueno o malo en función del valor que tomen estas características.
- Eficiencia: La velocidad de inspección debe ser acorde a los ritmos de producción, de forma que esta tarea no suponga un cuello de botella para el proceso productivo global. Para ello resulta necesario seleccionar una configuración software/hardware cuyo coste sea asumible, y que permita ejecutar los algoritmos anteriores de la forma más rápida y robusta posible.

Además de estas dos cualidades, los SIVA presentan las siguientes ventajas frente a la inspección visual llevada a cabo por expertos:

- Permiten la inspección exhaustiva de la producción. En muchos casos, la cantidad, proximidad y velocidad con la que llegan los productos, impide que una persona pueda inspeccionarlos en su totalidad.
- Son no invasivos, esto es, no alteran los productos durante el proceso de inspección. Algunas características de ciertos productos sólo son visibles para una persona si ésta puede manipularlos (abriéndolos, cogiéndolos para poder verlos en su totalidad, o incluso rompiéndolos). En el caso de los SIVA esto se puede evitar utilizando modelos de iluminación adecuados (p. e. contraluz), o técnicas de adquisición de imágenes que permitan observar características internas de los objetos (p. e. rayos X [Mery 03] o ultrasonidos [Winter 05]).
- Permiten la observación remota cuando se trata de productos peligrosos o de difícil acceso para las personas.
- Sus resultados son objetivos, predecibles y reproducibles.

- Tienen un cierto grado de flexibilidad ya que es posible ajustar algunos de sus parámetros de modo que la inspección sea más o menos estricta. Obviamente, los inspectores humanos son mucho más flexibles que cualquier SIVA, si bien suelen aplicar los cambios de criterio en la inspección de forma menos precisa y a veces algo arbitraria.

Sin duda, estas cualidades despiertan un gran interés en muchos sectores de la industria que ven en la incorporación de los SIVA a sus procesos productivos la oportunidad para mejorar la calidad de sus productos, reduciendo simultáneamente y de forma drástica los costes de producción. Entre los sectores que con más fuerza han apostado por el desarrollo de SIVA para automatizar la inspección de sus productos cabe destacar los siguientes: textil [Ibarra 95] [Brzakovic 97] [Kwak 00] [Mamic 00] [Schael 01], agroalimentario [Recce 96] [Noordam 00] [Ribeiro 00] [Fernández 01], metalúrgico [Fernández 97] [Sankaran 98] [Kim 99][Mery 03], de la madera y sus derivados [González 96] [Bhandarkar 99] [Vitrià 05], de la microelectrónica [Ko 00] [Chen 01], de la automoción [Viebig 04] [Schumacher 04] [Rivas 05], etc. En la Figura 8 se muestran algunos ejemplos de estos SIVA.

A pesar del enorme interés que estos sistemas suscitan en la industria, queda lejos el momento en el que su implantación pueda considerarse generalizada ya que son muchos los inconvenientes que quedan aún por resolver (elevado coste, poca flexibilidad, etc.). Es por ello que la inspección visual automática centra en la actualidad muchos de los esfuerzos que llevan a cabo tanto los investigadores como la industria.

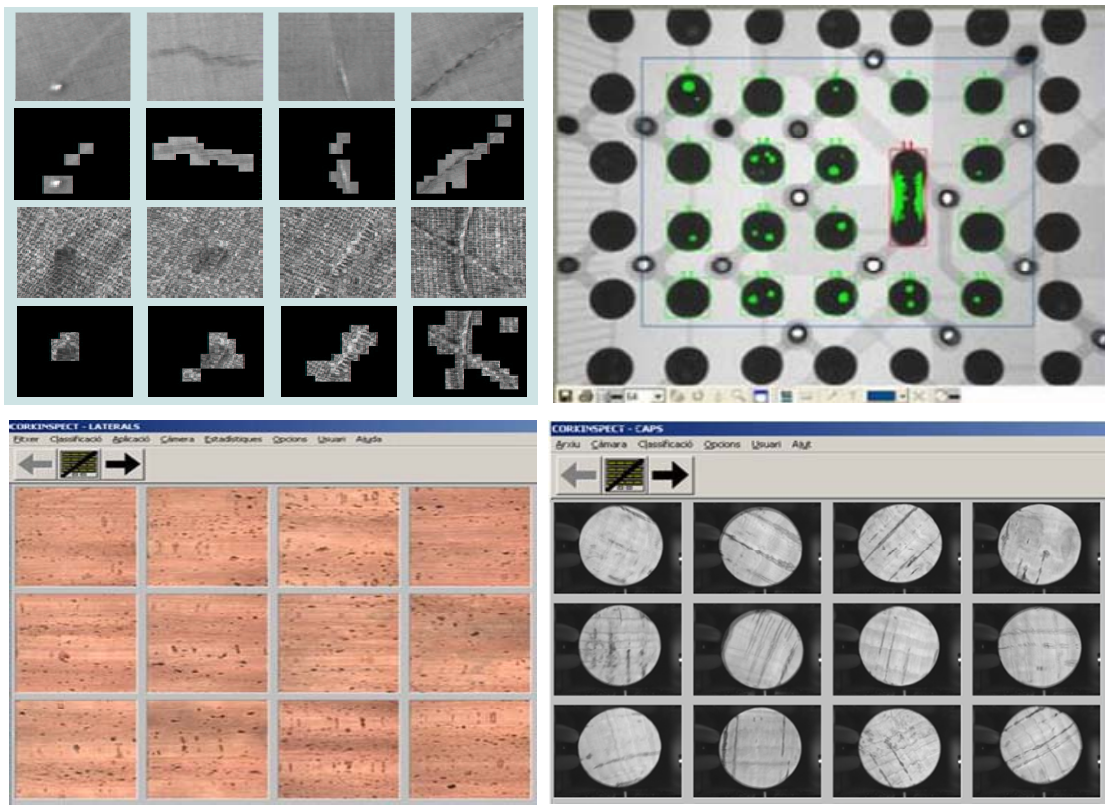


Figura 8. Sistemas de Inspección Visual Automatizados (SIVA)
Arriba izquierda: Detección de defectos en textiles [Shael 01], derecha: Detección de defectos en PCBs. Abajo: Aplicación CorkInspect para inspección de corcho [Vitrià 05]

2.2.2. Seguridad basada en biométricas

El término de origen griego "Biométrica" (medida de datos biológicos) se emplea en la actualidad para referirse al proceso de identificación automática de personas a través del análisis de su fisonomía y comportamiento[11] [Jain 03]. Las biométricas permiten identificar a las personas atendiendo a cómo son o a cómo se comportan y, siendo estas características personales e intransferibles, permiten realizar una identificación más fiable y precisa que aquellas que se basan en lo que las personas tienen (p. e. llaves o tarjetas de identificación) o en los datos que conocen (p. e. claves de acceso).

Cualquier característica humana puede ser considerada biométrica siempre que sea universal (todos los individuos la posean), distintiva (permita diferenciar a un individuo de otro), permanente (invariable en el tiempo al menos en lo sustancial), y por supuesto, cuantificable (medible de manera objetiva). Sin embargo, en la práctica no todas las biométricas son válidas o útiles ya que a la hora de construir una aplicación de este tipo deben considerarse factores tales como su rendimiento, su nivel de vulnerabilidad o la aceptación que éstas tienen entre los usuarios [Jain 04]. Por ejemplo, las huellas digitales [Maltoni 03] o el iris [Daugman 93], son dos de las biométricas más seguras, sin embargo, muchas personas son reticentes a permitir que se obtenga esta información de ellas. Por el contrario, la voz [Chaudhari 03] o el rostro [Li 05] de una persona son características menos distintivas ya que dos personas pueden tener timbres de voz o rasgos faciales muy similares. Sin embargo, esta información es sencilla de obtener ya que no requiere una donación activa por parte del individuo, que de hecho puede ni siquiera percatarse de que hay un micrófono o una cámara obteniendo esta información. En la Figura 9 se recogen éstas y otras de las medidas biométricas más utilizadas en la actualidad.

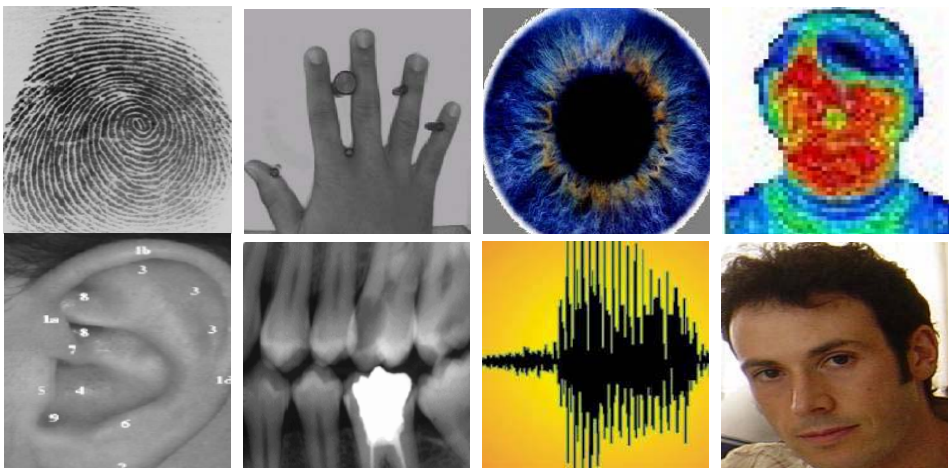


Figura 9. Algunos ejemplos de medidas biométricas.
De izquierda a derecha y de arriba abajo: huella digital, geometría de la mano, iris, termografía facial, forma de la oreja, radiografía bucal, señal de voz, rostro.

[11] Esta definición propuesta por A. K. Jain para el término *biométrica* es una de las más ampliamente aceptadas, aunque quizá sería más exacto utilizar el término *antropométrica*.

En la actual era de la información en la que el volumen de las transacciones económicas a través de la red (banca on-line, comercio electrónico, etc.) crece día a día a pasos agigantados, la seguridad y la confidencialidad se han convertido en una de las prioridades para un sinfín de empresas. El control del acceso no sólo a datos confidenciales sino también a lugares restringidos, requiere de un proceso altamente fiable que permita identificar a las personas autorizadas, impidiendo el acceso a aquéllas que no lo están. En este terreno, la aplicación de técnicas basadas en biométricas se perfila, sin duda, como una de las soluciones más plausibles. En la Figura 10 se muestran algunos de los dispositivos biométricos de control de acceso e identificación personal disponibles hoy en día en el mercado.

(a) (b) (c)

Figura 10. Dispositivos de seguridad biométricos.
Siemens Biometric Matching on Card Version 4.01, (b) Veridicom 5th Sense, (c) Chip de identificación implantable bajo la piel, de la empresa VeriChip.

Aunque la utilidad de emplear biométricas para la construcción de sistemas de seguridad es evidente, no es ésta su única aplicación. El uso de biométricas se abre paso en sectores tan diversos como el de la compresión de imágenes de vídeo (codificación de caras en imágenes de video-conferencia), la creación de aplicaciones usuario-reactivas (reaccionan al estado de ánimo del usuario), o la tele-medicina (monitorización a distancia de pacientes mediante el análisis de ciertos parámetros biométricos).

2.2.3. Aplicaciones de imagen médica

En medicina se utilizan habitualmente distintos tipos de imágenes para el diagnóstico de los pacientes: tomografías computerizadas (CT), imágenes de resonancia magnética (RMI), imágenes de ultrasonido (USI), imágenes de microscopía (óptica y electrónica), imágenes Doppler, etc.

La aplicación de técnicas de procesamiento de imágenes y de reconocimiento de patrones a este tipo de imágenes permite obtener resultados de un enorme interés para la práctica médica. Por ejemplo, en la actualidad es posible analizar muestras de sangre o de tejidos utilizando algoritmos morfológicos que permiten segmentar y caracterizar ciertos tipos de células (ver Figura 11) en busca de anomalías o simplemente para su recuento [Angulo 03] [Wählby 03].

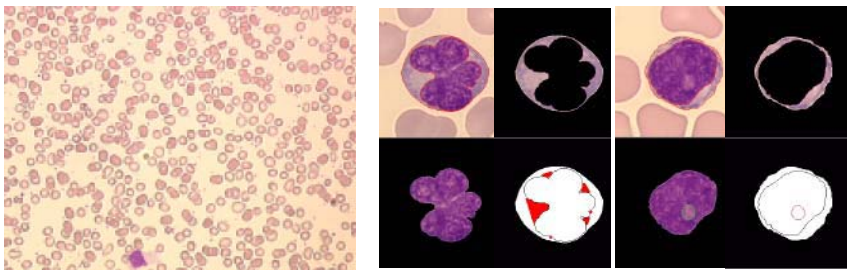


Figura 11. Segmentación y caracterización de leucocitos en sangre [Angulo 03].

Otra de las áreas en las que el procesamiento de imágenes médicas resulta de utilidad es en la elaboración de modelos tridimensionales del cuerpo humano o de partes de él. La elaboración de estos modelos requiere contar con una secuencia de imágenes adquiridas de forma consecutiva así como con los algoritmos adecuados para calcular las correspondencias entre los puntos de una imagen y los de su siguiente en la secuencia. Serán los resultados de estas correspondencias (*matching*) los que permitan deducir la tercera dimensión de coordenada en la imagen final.

Sin duda, estos modelos constituyen verdaderas enciclopedias visuales que resultan de gran ayuda tanto para el estudio como para la práctica de la medicina. Uno de los proyectos más ambiciosos en esta línea es el denominado *The Visible Human Project* [VHProject] que cuenta con una extensa base de datos de imágenes CT y RMI, así como con fotografías de criosecciones a color del cuerpo humano. Además, como parte de dicho proyecto se han desarrollado varias herramientas de consulta y visualización de estas imágenes, así como un entorno virtual tridimensional denominado AnatQuest [ANATQUEST] con el que es posible visualizar los distintos órganos y sistemas del cuerpo humano de manera independiente y diferenciada (ver Figura 12).

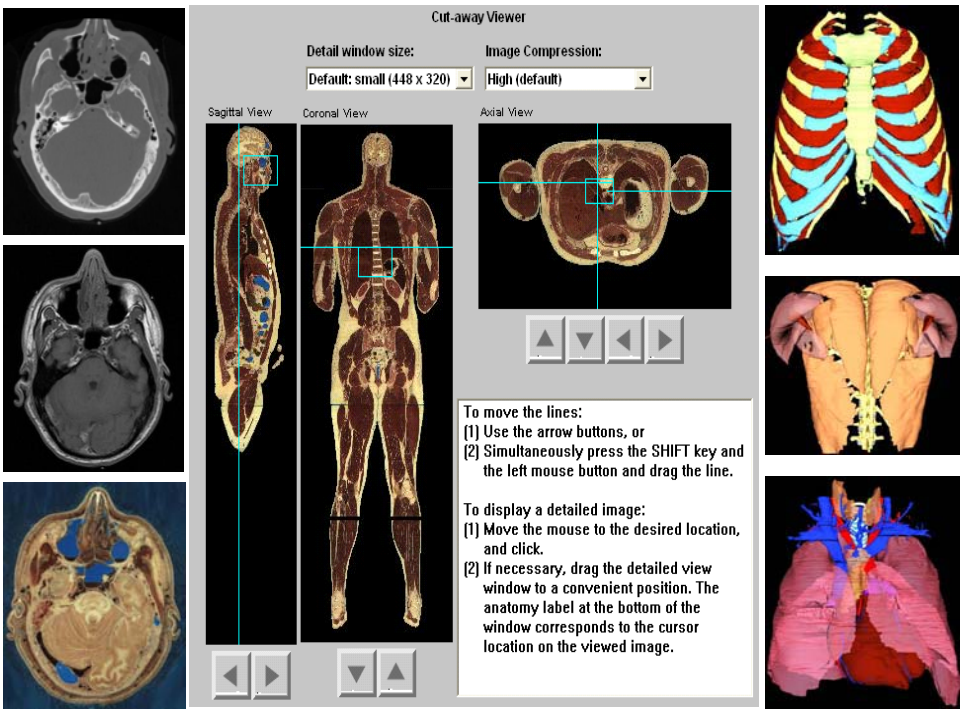


Figura 12. Reconstrucción 3D a partir de imágenes médicas [ANATQUEST].

Por último, y aunque existen otras muchas aplicaciones de tipo VIPS basadas en imágenes médicas, sólo mencionaremos un ejemplo más en relación con la cirugía asistida por ordenador [Hazan 03]. En este caso, el hecho de poder contar con un modelo tridimensional preciso de la zona que se va a intervenir en cada paciente permite a los cirujanos planificar al detalle y con anterioridad sus intervenciones. Además, cuando se trata de operaciones de riesgo que requieren un elevado grado de precisión, el hecho de poder contar con imágenes tridimensionales ampliadas de la zona que se está interviniendo, supone una ventaja indudable (ver Figura 13).

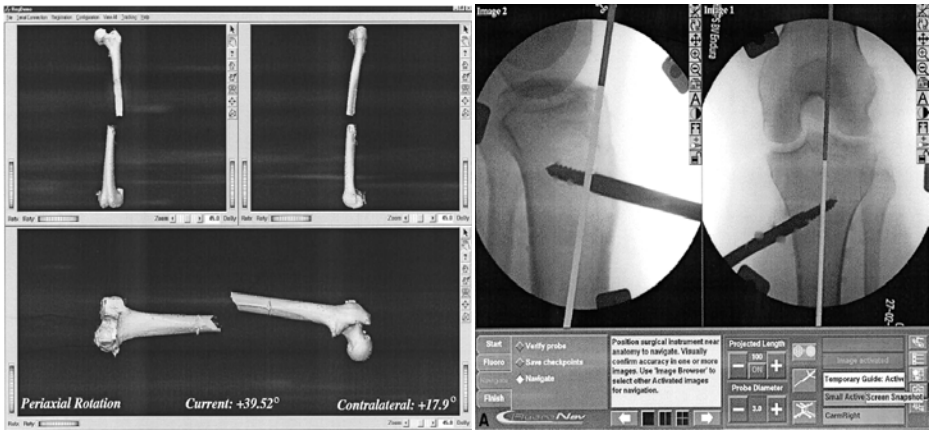


Figura 13. Sistema de cirugía guiado por computador.
Izquierda: Planificación de la intervención utilizando una reconstrucción 3D del hueso a reparar a partir de imágenes CT. Derecha: Seguimiento en tiempo real de la intervención.

2.2.4. Interfaces preceptuales de usuario

La disciplina denominada HCI (HCI, *Human-Computer Interaction*), de reciente desarrollo, estudia los distintos modos en los que el hombre puede interactuar con las máquinas y, en particular, con los ordenadores, a través de distintos tipos de interfaces. De entre ellas, pueden considerarse preceptuales aquellas que son interactivas, multimodales y que permiten que la comunicación hombre-máquina sea rica, natural y eficiente, lo que no se consigue con las actuales interfaces gráficas de usuario (GUI, *Graphical User Interface*), ni utilizando los dispositivos de entrada/salida estándar como el teclado, el ratón o la pantalla.

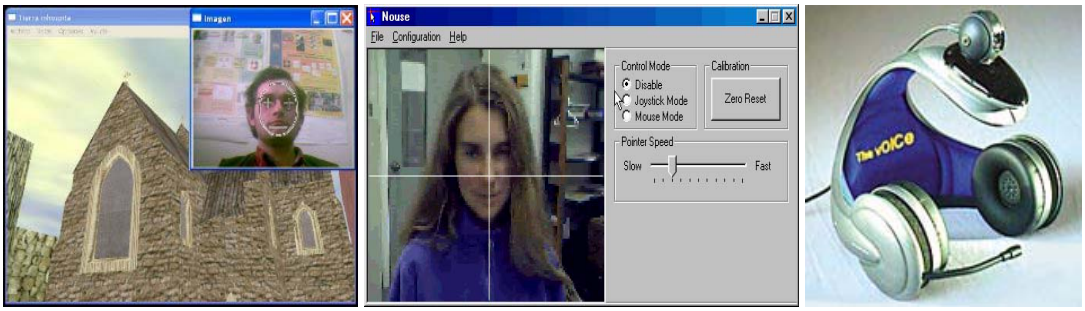
Las Interfaces Preceptuales de Usuario (PUI, *Perceptual User Interfaces*) se caracterizan por combinar (1) las capacidades naturales del hombre (comunicación, movimiento, conocimiento, etc.), (2) los dispositivos de entrada y salida del ordenador y (3) la aplicación de algoritmos de percepción y razonamiento.

El diseño de estas interfaces se inspira en la forma en que las personas suelen interactuar entre sí y con el entorno (ya sea de manera verbal o no verbal) con objeto de que resulten más naturales y sencillas de utilizar. Para ello, los dispositivos de entrada y salida deben ser transparentes para el usuario y deben permitirle comunicarse con el ordenador (y a éste con el usuario) de manera similar a como lo haría con otra persona, por ejemplo, mediante un diálogo hablado. Esto requiere hacer uso de distintas técnicas y tecnologías como, por ejemplo, el reconocimiento y generación de lenguaje hablado, la visón por computador, la visualización y animación de gráficos, el uso sensores hápticos, etc.

Los ordenadores han evolucionado de manera espectacular. Mientras que hasta épocas muy recientes se utilizaban casi exclusivamente como procesadores de texto o para la gestión de bases de datos, en la actualidad se utilizan, además, con propósitos tan diversos como la video-conferencia o la tele-enseñanza, por poner sólo algunos ejemplos. De hecho, no sólo ha cambiado su uso sino también su forma y diseño, de modo que ahora, además de ordenadores de sobremesa y portátiles, es posible encontrar pequeños ordenadores en todos los ámbitos de la vida cotidiana: desde en una PDA (*Personal Digital Assistant*) o en un teléfono móvil, hasta en una nevera o en un automóvil. En la mayoría de los casos, estos sistemas van dirigidos al gran público y por lo tanto deben poder utilizarse sin necesidad de ser un experto en el manejo de ordenadores. Es precisamente en este campo en el que el desarrollo de interfaces perceptuales cobra su mayor importancia.

En la actualidad se están desarrollando numerosos trabajos en este campo, de entre los que se han elegido, a modo de ejemplo, los siguientes.

En el área de las aplicaciones destinadas al entretenimiento cabe mencionar algunos intentos de desarrollar aplicaciones que, en lugar de utilizar como dispositivos de entrada los tradicionales (ratón y teclado), emplean cámaras web de bajo coste que permiten al ordenador "ver" al usuario y reaccionar, por ejemplo, a cambios en su posición corporal. En esta línea se han desarrollado aplicaciones como las descritas en [García 05b] (navegación en un entorno virtual tridimensional utilizando, a modo de ratón 3D, los movimientos de la cara del usuario, ver Figura 14a) o en [Gorodnichy 04] (desplazamiento del cursor de la aplicación siguiendo el movimiento de la nariz del usuario, ver Figura 14b). También existen aplicaciones en las que lo que se trata de hacer mas amigable es la forma en que el ordenador presenta los datos al usuario utilizando, por ejemplo, unas gafas de visión tridimensional en lugar de la pantalla lo que permite al usuario visualizar, por ejemplo, el escenario de un juego de manera mucho más realista [I-GLASSES].



(a) **(b)** **(c)**

Figura 14. Ejemplos de Interfaces preceptúales.
(a) Sistema de navegación en un mundo virtual 3D utilizando la cara como ratón. (b) Sistema de control del cursor utilizando la nariz (c) Sistema The vOICe.

Otras aplicaciones tratan de facilitar la interacción entre el usuario y el ordenador en los dos sentidos modificando no sólo los dispositivos de entrada sino también los de salida. Este es el caso del sistema "The vOICe" [Amedi 05] que permite a las personas ciegas "oír" la imagen que tienen ante sí y no pueden ver.

2.2.5. Sistemas de ayuda a la navegación

En las últimas décadas, los trabajos relacionados con el desarrollo de sistemas de ayuda a la navegación para todo tipo de dispositivos móviles, desde robots autónomos [López 03] hasta automóviles [Bertozzi 02] [Grasby 04], han experimentado un avance espectacular, tanto en su número como en el nivel de los resultados alcanzados [DeSouza 02]. Muchos de estos sistemas utilizan para modelar el entorno en el que deben desenvolverse varias fuentes de información de manera simultánea (cámaras, sensores de ultrasonidos, sistemas GPS, etc.), cuyos datos deben combinarse y analizarse convenientemente a fin de extraer toda aquella información que pueda resultar de utilidad para guiar, de manera segura y eficiente, su desplazamiento por el medio (detección de obstáculos, indicadores de dirección y/o sentido, marcadores de peligro, etc.).

Comúnmente los sistemas de navegación se clasifican en función de ciertos parámetros relativos al tipo de entorno en el que éstos se desenvuelven, por ejemplo, si éste es interior o exterior, estructurado (por ejemplo, un almacén con pasillos distribuidos de manera regular) o no estructurado (por ejemplo, una carretera en la que pueden aparecer un sinfín de objetos de manera inesperada), parcialmente conocido a priori (por ejemplo, mediante un mapa) o no, etc. Otro modo de clasificar estos sistemas es atendiendo al tipo de estrategias que emplean para interactuar con su entorno (estrategias activas, reactivas, inspiradas en el comportamiento de ciertas especies animales [LOCUST], etc.). En la siguiente figura se muestran algunos ejemplos de sistemas de navegación.

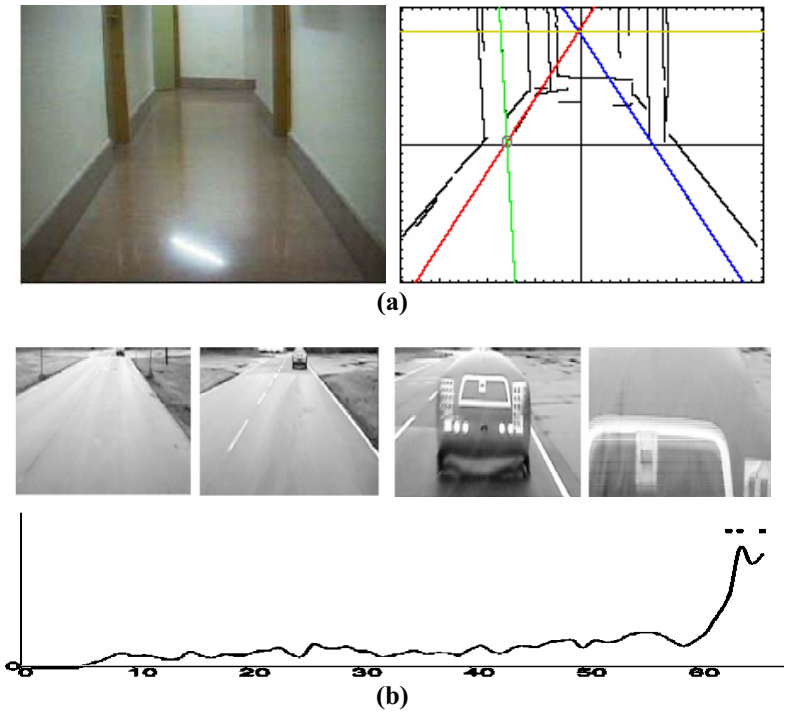


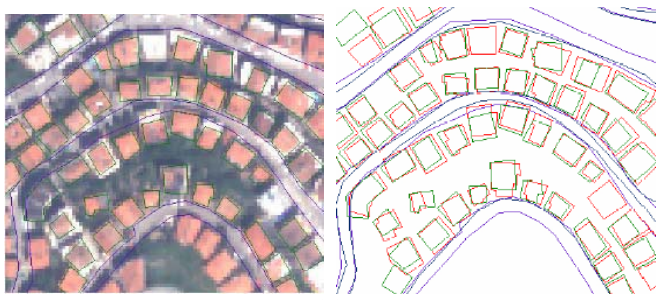
Figura 15. Ejemplos de sistema de navegación basados en visión *artificial*.
(a) Sistema de navegación para robots en entornos estructurados [López 03] (b) LOCUST: Sistema de detección de colisiones para vehículos inspirado en el sistema visual de las langostas [LOCUST].

2.2.6. Sistemas de información geográfica

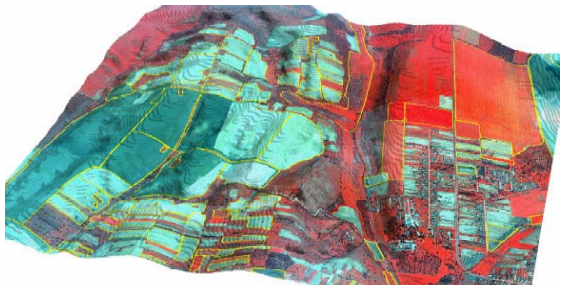
En las últimas décadas los sistemas de información geográfica (GIS, *Geographical Información Systems*) en los que se aplican técnicas de visión artificial, procesamiento de imágenes y reconocimiento de patrones a imágenes obtenidas vía satélite, se han utilizado únicamente desde un enfoque experimental. Las agencias espaciales de algunos países han tratado de demostrar su utilidad en el campo de la cartografía, la búsqueda de recursos minerales, la reconstrucción de mapas tridimensionales, la monitorización del entorno (detección de incendios, seguimiento fenómenos meteorológicos, etc.) [Sirta 97]. Es de esperar que en el próximo futuro, el aumento en el número de satélites de observación que proporcionan imágenes de alta resolución a velocidades cada vez mayores (casi en tiempo real), así como los avances que se han conseguido tanto en el campo de la visión artificial (y disciplinas relacionadas) como en el de la electrónica o la informática, permitan que la teledetección pase de la fase experimental a una más operativa.

En los sistemas de información geográfica, el procesamiento y análisis de imágenes desempeña un papel fundamental para garantizar la calidad de las imágenes que se reciben, establecer y explotar la complementariedad de las distintas fuentes (satélites) y suministrar herramientas semi-automáticas para procesar y diseminar, en tiempo real, la información que se extrae de las imágenes.

Existen varias empresas que suministran aplicaciones para el procesamiento de imágenes obtenidas vía satélite, como por ejemplo MIR Télédétection[12], aunque también existen aplicaciones abiertas y de libre distribución como GRASS [GRASS].



(a)



(b)

Figura 16. Ejemplos de GIS aplicados a cartografía (a) urbana (b) de cultivos.

[12] Información de la empresa MIR Télédétection disponible en:
http://www.mirteledetection.com/services_en.html

2.3. Herramientas para procesamiento de imágenes

Como se comentó en el capítulo de Introducción, en la actualidad los VIPS suelen construirse como sistemas mixtos hardware/software (HW/SW). El peso de cada uno de estos componentes variará en función de:

- los requisitos que se establezcan para cada VIPS en particular (velocidad de respuesta, precisión, flexibilidad, robustez, etc.),
- los recursos con los que se cuenta (nivel de conocimientos acerca de las técnicas de procesamiento de imágenes y del lenguaje de programación o herramienta que se vaya a emplear, etc.) y,
- las exigencias impuestas por el mercado (productos similares desarrollados por la competencia, tiempo de salida al mercado, número de sistemas del mismo tipo que van a construir, precio, etc.).

De este modo, potenciar en un VIPS cualidades como su velocidad de respuesta requerirá dar un mayor peso al componente HW (por lo general más eficiente aunque también más caro y costoso de desarrollar), mientras que si lo que se desea es potenciar otras como la flexibilidad o la rapidez de desarrollo será el componente SW el que cobre mayor importancia.

Las herramientas que se comentan a continuación se han agrupado atendiendo a si se utilizan para desarrollar el componente SW o el HW de los VIPS. El primero de ellos estará formado por un conjunto de aplicaciones que se ejecutarán, de manera secuencial, sobre un procesador convencional (microprocesador). Por otra parte, el componente HW estará formado por una serie de dispositivos HW de propósito específico, por ejemplo ASICs (*Application-Specific Integrated Circuit*) o DSPs (*Digital Signal Processor*) o más general, por ejemplo FPGAs (*Field Programmable Gate Array*), con capacidad para procesar datos en paralelo.

El desarrollo del componente SW suele ser mucho menos costoso en tiempo que el del correspondiente HW dado que, como se verá a continuación, las herramientas empleadas en el primer caso suelen ser de más alto nivel y por lo tanto más sencillas e intuitivas de usar que las empleadas en la descripción y/o programación del HW.

2.3.1. Herramientas de programación SW

En la actualidad es posible encontrar distintos tipos de herramientas para procesamiento de imágenes:

- Entornos de edición de imágenes. Estas herramientas, que por lo general proporcionan una interfaz gráfica de usuario, no permiten ningún tipo de programación, sino sólo leer y guardar imágenes en distintos formatos (BMP, JPG, GIF, etc.) y procesarlas utilizando ciertas opciones predefinidas en la aplicación (ajuste de contraste, brillo, color, etc.). Algunos ejemplos de herramientas de este tipo son Microsoft® Office Picture Manager o Corel® Paint Shop™ Pro®.

- Entornos de programación de aplicaciones de procesamiento de imágenes. Estas herramientas, a diferencia de las anteriores, permiten desarrollar, compilar y ejecutar código escrito en un lenguaje de programación que por lo general proporciona la propia herramienta y que puede ser de tipo textual o visual. A este grupo pertenecen herramientas como Matlab [MATLAB] y VisiQuest [VISIQUEST].
- Librerías de funciones de procesamiento de imágenes. Estas herramientas ofrecen un conjunto de funciones que pueden invocarse desde programas escritos utilizando un lenguaje de programación compatible con ellas. La mayoría de estas librerías están optimizadas para alguna plataforma hardware, como es el caso de las librerías MIL (*Matrox Imaging Library*) [MIL] optimizadas para el hardware de Matrox®, o de las IPP (*Integrated Performance Primitives*) [IPP] o las OpenCV (*Open Computer Vision*) [OPENCV], ambas optimizadas para los procesadores de Intel®.

2.3.1.1. Entornos de programación para el desarrollo de VIPS

A continuación se describen algunos ejemplos de los dos últimos tipos de herramientas ya que éstas permiten programar aplicaciones y por lo tanto son de utilidad para el desarrollo de VIPS.

- **Herramienta VisiQuest® de AccuSoft Corporation™ [VISIQUEST]**

VisiQuest® es un potente entorno de programación visual para procesamiento y visualización de imágenes. La apariencia de un programa desarrollado utilizando VisiQuest® es muy similar a un diagrama de bloques en el que cada módulo (glyph) representa una función, rutina u operador. Estos módulos están interconectados entre sí mediante líneas que representan el flujo de información entre las distintas funciones.

VisiQuest® proporciona mecanismos de abstracción y control típicamente no incluidos en otros lenguajes de programación visual. Por ejemplo, es posible definir jerarquías de bloques (anidamiento), realizar iteraciones, parametrizar los bloques, etc. lo que, unido al lo intuitivo de su manejo, hace de VisiQuest® un entorno de programación muy sencillo de manejar, incluso para usuarios no expertos en programación.

La creación de un programa requiere que el usuario seleccione, en primer lugar, los bloques que desea incorporar al mismo y que los distribuya sobre la superficie gráfica de diseño. Posteriormente, deberá conectarlos indicando el orden y el sentido en el que éstos deben intercambiar la información entre sí. El flujo de control de estos programas puede alterarse añadiendo estructuras de selección y control para bifurcar, combinar o repetir la funcionalidad asociada a uno o varios de los bloques. Los datos de entrada de los *glyphs* (parámetros de las funciones), puede fijarlos el usuario de manera interactiva, o bien dejar que sea el propio programa el que los calcule durante su ejecución.

VisiQuest® proporciona la posibilidad de crear nuevos glyphs (además de los incluidos en la herramienta) a partir de código C/C++ y más recientemente también a partir de código Matlab. Todos los bloques VisiQuest® pueden incluir señales de control tanto en sus entradas como en sus salidas lo que permite, entre otras cosas, sincronizar

su ejecución indicando, por ejemplo, que un bloque debe esperar a que acabe otro antes de comenzar a ejecutarse. Los parámetros de cada bloque pueden ser de cualquiera de los tipos primitivos proporcionados típicamente por otros lenguajes de programación (entero, real, booleano, etc.).

VisiQuest® permite definir jerarquías de glyphs (bloques complejos que contienen en su interior otros más simples), lo que facilita la legibilidad y modularidad de los programas y hace de ésta una potente herramienta de prototipado rápido. Además, VisiQuest® puede instalarse sobre entornos Windows®, Mac OSX, Linux, etc.

- **Herramienta Simulink® de The Mathworks™ [SIMULINK]**

Como en caso anterior, Simulink® es también un entorno de programación visual que permite el desarrollo de una amplia variedad de sistemas dinámicos (lineales, no lineales, discretos, de tiempo continuo, etc.). Esta herramienta permite realizar diagramas de bloques arrastrando y pegando componentes (*drag-and-drop*) de alguna de las librerías disponibles, así como modificar los parámetros del modelo y visualizar los resultados a medida que estos se obtienen durante la ejecución de una simulación del sistema que se está diseñando. También, como en el caso anterior, Simulink® permite al usuario crear nuevos componentes, adaptar los existentes, o anidarlos para crear bloques complejos.

La conexión directa existente entre Simulink® y el entorno de programación textual MATLAB®, ampliamente utilizado en ingeniería, permite incorporar a las ya de por sí extensas librerías (toolboxes) de Simulink® las disponibles para MATLAB®, y en particular las dedicadas a la adquisición y procesamiento de imágenes (*Image Acquisition Toolbox* [MATLAB IAT] y *Image Processing Toolbox* [MATLAB IPT], respectivamente). Además, MATLAB permite incorporar la funcionalidad ofrecida por varias de las librerías de procesamiento de imágenes que se discuten en el apartado siguiente, lo que a su vez permite que estas puedan ser utilizadas desde Simulink®. Gracias a estas incorporaciones Simulink®, inicialmente orientado al diseño de sistemas de control, permite en la actualidad la construcción de aplicaciones de tipo VIPS.

Simulink® puede ejecutarse sobre plataformas Windows®, Mac OSX, UNIX y OpenVMS, siendo posible transferirse los modelos diseñados para una plataforma a otra conservando sus características y funcionalidad.

- **Otras herramientas**

Además de estos entornos de programación existen otros como WIT [WIT] o Sherlock [SHERLOCK] de la casa DALSA Coreco® que, como los anteriores, también utilizan un lenguaje de programación visual basado en diagramas de bloques para el desarrollo de aplicaciones de procesamiento de imágenes optimizadas para MMX® y compatibles con el hardware de Coreco®.

Del mismo modo, la herramienta LabView® [LABVIEW] de Nacional Instruments™ proporciona también un entorno de programación visual basado en bloques y permite, con ciertas limitaciones, el desarrollo de aplicaciones de procesamiento de imágenes ya que está principalmente concebida para la construcción de sistemas relacionadas con la instrumentación electrónica y el control de señales.

2.3.1.2. Librerías de procesamiento de imágenes

A continuación se describen algunas de las librerías de procesamiento de imágenes actualmente disponibles en el mercado. Se hará especial hincapié en las ofrecidas por las casas Intel© y Matrox© ya que éstas son las empleadas por la herramienta de prototipado de VIPS denominada IP-CoDER que ha sido desarrollado como parte de este trabajo y cuyo diseño, implementación y funcionamiento se expondrá detalladamente en el Capítulo 5.

- **Librerías de Intel®**

La empresa Intel® ofrece varias librerías para procesamiento de imágenes que están optimizadas para sus procesadores MMX®, aunque son bastante portables ya que pueden instalarse sobre cualquier sistema operativo y plataforma hardware. Entre ellas cabe destacar las IPP (*Integrated Performance Primitives*) [IPP] y las OpenCV (*Open Computer Vision*) [OPENCV].

Estas últimas son librerías escritas en ANSI C/C++ cuyo código es abierto y de libre distribución. Las OpenCV están basadas en una versión anterior de las IPP denominada IPL (Image Processing Library) que ofrecían un conjunto de funciones de procesamiento de imágenes de bajo nivel, que OpenCV ha extendido hasta completar un juego de más de 350 funciones. Entre ellas cabe destacar algunas como las de adquisición de imágenes (desde una amplia variedad de dispositivos, entre ellos los de la casa Matrox®), las de calibración de cámaras, las de reconocimiento de patrones, las de seguimiento y análisis de objetos en movimiento, o las de reconstrucción 3D. Cabe destacar el hecho de que las OpenCV son compatibles con un entorno de prototipado y desarrollo tan ampliamente utilizado en Ingeniería como es Matlab®. Esto es posible gracias a un toolbox que proporciona las interfaces entre ambas herramientas.

- **Librerías de Matrox®**

La casa Matrox® ofrece, entre sus productos, las librerías MIL (Matrox® Imaging Library) [MIL] que están optimizadas para sus tarjetas de adquisición y digitalizadoras. Estas librerías pueden instalarse sobre sistemas operativos Linux y Windows® (2000 y XP). Sin embargo, en los entornos Windows® sólo es posible incorporar estas librerías a programas escritos utilizando entornos de desarrollo Microsoft® (Visual C++®, Visual Basic®, .NET®), mientras que en entornos Linux éstas son compatibles con GCC (*GNU Compiler Collection*).

Como en el caso anterior, las MIL proporcionan un amplio conjunto de funciones entre las que se incluyen algunas para procesamiento de imágenes a bajo nivel, análisis de blobs, detección de bordes, reconocimiento de patrones, calibración, OCR (*Optical Character Recognition*), etc.

Junto con las MIL, Matrox® suministra el producto ActiveMIL consistente en un conjunto de controladores OCX que facilitan el desarrollo de las aplicaciones de procesamiento de imágenes que utilizan las MIL, permitiendo al programador incluir estos controladores en la interfaz gráfica del programa, de manera que pueda configurarlos utilizando una plantilla en lugar de programar el código correspondiente.

- **Otras librerías**

Además de las antes mencionadas cabe destacar otras librerías de procesamiento de imágenes como Java Advanced Imaging (JAI)[JAI] de Sun Microsystems® (extensión de la API básica de Java que proporciona un conjunto de clases específicas para el de desarrollo de aplicaciones de procesamiento de imágenes), o las librerías (*toolboxes*) de The Mathworks® para adquisición (*Image Acquisition Toolbox*) [MATLAB IAT] y procesamiento de imágenes (*Image Processing Toolbox*) [MATLAB IPT], ambas escritas utilizando el lenguaje de programación propio de Matlab®.

2.3.2. Herramientas para la especificación/programación del HW

Los dispositivos que pueden incluirse como parte del componente HW de un VIPS y, de manera más general de cualquier sistema mixto HW/SW, pueden clasificarse del siguiente modo según su menor o mayor grado de flexibilidad:

- **Dispositivos no programables ni reconfigurables**. Este es el caso de los ASIC (A*pplication-Specific Integrated Circuit*), dispositivos de propósito muy específico que suelen producirse a gran escala y bajo coste. La arquitectura y la funcionalidad de estos dispositivos suelen especificarse utilizando un HDL[13] (*Hardware Description Language*), a partir del cual es posible realizar una simulación software para validar el correcto funcionamiento del dispositivo antes de proceder a su implementación final de manera masiva. Una vez implementado el ASIC, no es posible modificar su arquitectura ni funcionalidad.
- **Dispositivos programables** (PLD, *Programmable Logic Device*). Este es el caso de los DSP que si bien, como en el caso de los ASIC, tienen una arquitectura no modificable que se especifica utilizando un HDL ofrecen, como los procesadores convencionales (microprocesadores), un conjunto de instrucciones que permiten programar su lógica. Dado que los DSP son procesadores de propósito específico para el tratamiento de señales digitales, tanto su arquitectura como su juego de instrucciones están optimizados para este tipo de tareas.
- **Dispositivos reconfigurables**. Este es el caso de las FPGA que, además de ser programables como los DSP, tienen una arquitectura reconfigurable (celdas que pueden conectarse de distintos modos dando lugar a diferentes configuraciones hardware). Tanto la descripción de la arquitectura como la programación de la lógica de las FPGA se realiza utilizando los denominados lenguajes SLD (*System-Level Design Language*).

Entre los lenguajes de especificación de hardware (HDL) antes mencionados cabe desatacar, por su elevado grado de implantación, VHDL (VHSIC HDL, *Very-High-Speed Integrated Circuit HDL*) [VHDL] desarrollado por el Ministerio de Defensa norteamericano y Verilog® HDL [VERILOG] de Cadence® Design Systems Inc.

[13] Al contrario de lo que ocurre con los lenguajes de programación clásicos como C/C++ (secuenciales), los HDL suelen incluir mecanismos específicos para expresar, de manera explícita, la concurrencia inherente al procesamiento hardware.

Del mismo modo, entre los lenguajes SLD para programación de dispositivos reconfigurables, cabe destacar HandelC [HANDELC] de Celoxica® o SystemC™ [SYSTEMC] de OSCI (*Open SystemC™ Initiative*).

Aunque no se trate de un HDL ni de un SDL conviene destacar, por su relación directa con este trabajo, la herramienta System Generador® [SYSTEM GENERATOR] de Xilinx®. Ésta proporciona un conjunto de bloques Simulink con los que es posible simular, de manera precisa, el comportamiento de las FPGA de Xilinx. Estos bloques pueden integrarse con otros de los proporcionados por Simulink de modo que es posible construir modelos mixtos que contengan bloques System Generator (simulación SW del componente HW) y otros bloques Simulink (componente SW).

Capítulo 3

Paradigmas de desarrollo de software

Aportación al diseño e implementación de VIPS

Este capítulo describe cómo la incorporación de algunos de los paradigmas actuales de desarrollo de software al proceso de diseño e implementación de los VIPS potencia la reutilización y aumenta el grado de flexibilidad de estos productos, a la vez que reduce ostensiblemente el coste y el tiempo que es necesario invertir en su desarrollo.

Para ello, en primer lugar, se describirá el procedimiento que se sigue en la actualidad para construir los distintos tipos de VIPS, reseñando cuáles son los principales inconvenientes y limitaciones que éste presenta. A continuación se analizará cómo solventar estas deficiencias incorporando al proceso algunas de las tendencias más actuales dentro del campo de la Ingeniería del Software. Un breve repaso a estos nuevos paradigmas de desarrollo de software, observando la repercusión que han tenido en la construcción de otros tipos de sistemas, permitirá establecer el alcance y grado de aportación de cada uno de ellos al diseño e implementación de los VIPS. Por último, y como más destacado, se propone una nueva metodología de desarrollo de VIPS que cubre todas las fases del ciclo de vida de estos productos y que se basa en la integración de varios de los paradigmas previamente descritos, constituyendo una de las principales aportaciones de esta Tesis.

3.1. Introducción

Como queda recogido en el capítulo anterior, existen diversas disciplinas involucradas en la construcción de los VIPS cuyo grado de aportación al proceso de desarrollo de estos sistemas depende en gran medida del tipo de aplicación al que se destinen. Así, por ejemplo, en los sistemas de teledetección una buena gestión de las comunicaciones resulta esencial, mientras que en el caso de los Sistemas de Inspección Visual Automatizados (SIVA) es el componente de control el que suele tener mayor importancia.

En cualquier caso y con independencia del tipo de VIPS del que se trate, las tareas relacionadas con el procesamiento, la caracterización y la clasificación de la información visual aparecen siempre y son parte fundamental de estos sistemas; de hecho, pueden considerarse el núcleo lógico de los VIPS y el denominador común compartido por todos ellos.

Del mismo modo, los requisitos que guían el diseño y la implementación de los distintos tipos de VIPS varían también en función de sus distintas aplicaciones. Así, es razonable pensar que a la hora de construir un sistema de seguridad biométrico tendrán mayor peso los requisitos relacionados con la fiabilidad o la robustez del sistema, frente a otros como su velocidad de respuesta. Por el contrario, en el caso de los SIVA, el proceso de inspección deberá ser ante todo veloz para no retrasar el ritmo global de la producción, incluso si ello conlleva incurrir en un cierto número de errores. En cualquier caso, el alto nivel de rendimiento que suele exigirse a estos sistemas hace que resulte necesario, en muchas ocasiones, utilizar dispositivos hardware especializados que permitan alcanzar los requisitos impuestos a estos sistemas (cámaras de alta resolución, autómatas programables, buses de comunicaciones robustos para entornos industriales, procesadores hardware de propósito específico, etc.).

A continuación se analizan algunos de los requisitos que típicamente gobiernan el proceso de construcción de los VIPS. Este análisis ayudará a comprender mejor el procedimiento que se sigue en la actualidad para implementar estos sistemas, el cual se describirá más adelante en el presente capítulo.

3.1.1. Requisitos típicamente impuestos a los VIPS

Los procesos de análisis, definición y gestión de los requisitos resultan vitales a la hora de abordar, con una cierta garantía de éxito, el diseño y la implementación de cualquier producto software, más aún hoy en día cuando la complejidad de las aplicaciones es tan elevada. En los últimos años se han sucedido grandes cambios en el campo de la Ingeniería de Requisitos (ver Definición 5), siendo actualmente una de las disciplinas más activas dentro del campo de la Ingeniería del Software.

> *La ingeniería de requisitos estudia los servicios que el cliente demanda de un sistema y las restricciones bajo las que éste debe desarrollarse y operar [Sommerville 04].*
>
> *La ingeniería de requisitos comprende el conjunto de actividades que se llevan a cabo a lo largo de todo el ciclo de vida del software y que buscan identificar las necesidades del cliente, documentarlas, verificar que son consistentes entre sí y analizarlas tratando de encontrar nuevos requisitos, así como el conjunto de procesos que dan soporte a estas actividades y los mecanismos de trazabilidad que permiten realizar su seguimiento [DoD 91].*

Definición 5. Ingeniería de Requisitos.

Existen distintas formas de catalogar los requisitos asociados al software, aunque quizá una de las más intuitivas y ampliamente aceptadas sea la de Ian Sommerville quien propone la siguiente clasificación [Sommerville 04]:

- Requisitos funcionales: especifican los servicios que debe proporcionar el sistema, cuál debe ser su reacción ante las distintas entradas y cómo debe comportarse frente a las distintas situaciones que se puedan plantear.
- Requisitos no funcionales: establecen las restricciones bajo las que debe desarrollarse (legibilidad, documentación, etc.) y operar el sistema (tiempo de respuesta, recursos consumidos, obligatoriedad del uso de un determinado lenguaje de programación / sistema operativo / herramienta de desarrollo por motivos de compatibilidad, portabilidad o simplemente para ajustarse a la política de la empresa, etc.).
- Requisitos propios del dominio: Características del sistema específicas del dominio de aplicación al que éste pertenece. Pueden ser nuevos requisitos funcionales, restricciones sobre requisitos previamente definidos, o especificaciones de cómo llevar a cabo ciertas operaciones.

El cumplimiento de los requisitos funcionales de un determinado producto software está directamente relacionado con su eficacia. De manera general, y en particular en el caso de los VIPS, un sistema eficaz es aquel que proporciona la funcionalidad adecuada para resolver las tareas que tiene encomendadas.

Un sistema es eficaz o no – cumple o no con lo que se espera de él – y por lo tanto no cabe decir que un sistema es más eficaz que otro. La eficacia es un requisito ineludible y universal ya que todo sistema debe al menos cumplir con la finalidad para la que ha sido concebido. Ahora bien, a la hora de construir un VIPS puede suceder que la concurrencia de ciertos requisitos adicionales (coste, velocidad, precisión, consumo, peso, etc.) haga que resulte imposible implementar toda la funcionalidad inicialmente prevista. En este caso será necesario ajustar los requisitos funcionales atendiendo a criterios más realistas.

La funcionalidad de un VIPS está directamente relacionada con la selección de los algoritmos que éste debe implementar, el orden en el que han de ejecutarse y los mecanismos de comunicación y sincronización necesarios para su correcta interacción. Por lo general, existen varias combinaciones de algoritmos que resuelven de forma igualmente eficaz un mismo problema. La elección de una u otra solución se hace entonces tratando de maximizar (de manera priorizada) el resto de los requisitos impuestos al sistema, y en particular, su eficiencia. Un VIPS es tanto más eficiente cuanto mayores son sus prestaciones (velocidad, precisión, robustez, etc.) y menor la cantidad de recursos que consume (capacidad de memoria y/o de procesamiento, energía, etc.).

A la hora de optimizar el rendimiento de cualquier sistema debe prestarse especial atención a aquellas tareas que tienden a consumir más recursos, ya que éstas pueden convertirse en cuellos de botella que frenen el rendimiento global del sistema. En el caso de lo VIPS, las tareas más costosas suelen ser las relacionadas con el tratamiento de la información visual, dado el ingente volumen de información contenido en cada imagen y la complejidad de algunos de los algoritmos necesarios para su procesamiento. Por ello, gran parte de los esfuerzos que se invierten a la hora de construir un VIPS suelen dedicarse a tratar de implementar estas tareas de la forma más eficiente posible.

Para tratar de mejorar la eficiencia de un VIPS, y en particular la de las tareas relacionadas con el procesamiento de la información visual, existen dos posibles enfoques no excluyentes: mejorar la eficiencia de los algoritmos empleados (componente software) y/o ejecutarlos sobre una plataforma más potente (componente hardware).

En cuanto a las mejoras en la eficiencia del componente software de los VIPS, y en particular de los algoritmos asociados a las tareas de procesamiento de imágenes y reconocimiento de patrones, se han logrado algunos avances en los últimos años que han permitido mejorar el rendimiento de algunos de los algoritmos clásicos, así como desarrollar nuevos procedimientos para abordar ciertos problemas de forma más eficiente.

Sin embargo, muchas de las mejoras en la eficiencia de estos algoritmos se han logrado gracias a la detección y explotación del paralelismo inherente a estas tareas. En función de las características de cada aplicación, puede resultar más adecuado modelar un paralelismo de datos (aplicar varios algoritmos en paralelo a una misma imagen), un paralelismo de procesos (aplicar un mismo algoritmo a varias imágenes simultáneamente), o una combinación de ambos.

Así, cuando los algoritmos tradicionalmente expresados de manera secuencial para ejecutarse sobre procesadores de propósito general[14], consiguen formularse en paralelo y ejecutarse sobre un dispositivo hardware de propósito específico (por ejemplo una FPGA), se obtiene una mejora considerable de su rendimiento (aumento de velocidad, reducción de la cantidad de memoria necesaria, etc.). Sin embargo, no existe un procedimiento universal que permita optimizar, de manera general, el diseño paralelo de un algoritmo para su ejecución en cualquier plataforma hardware. De hecho, estas optimizaciones requieren un profundo conocimiento de la arquitectura hardware del dispositivo que se vaya a emplear, a fin de planificar el uso de los recursos disponibles de modo que se maximice su rendimiento. Además, la programación de estos dispositivos suele realizarse utilizando lenguajes de muy bajo nivel lo que dificulta en gran medida el desarrollo, modificación, validación y mantenimiento del código. Por ello, la decisión de utilizar dispositivos hardware específicos para mejorar la eficiencia y en particular la velocidad de respuesta de los VIPS, conlleva un aumento considerable en el tiempo que es necesario invertir en el desarrollo de estos productos. Es más, la codificación y optimización de los algoritmos necesarios puede requerir tanto tiempo que, al ritmo que evoluciona la electrónica, muy probablemente aparecerán en el mercado dispositivos mucho más potentes que el inicialmente seleccionado y sobre los que dichas optimizaciones tendrían un impacto prácticamente despreciable. Además, a pesar del abaratamiento de estos dispositivos, la complejidad de algunos algoritmos requiere emplear ciertas tecnologías que todavía resultan demasiado costosas para ser viables.

Por todo ello, cuando no es posible cumplir con los requisitos impuestos a un determinado VIPS utilizando únicamente un procesador convencional (configuración totalmente software), en lugar de optar por una configuración totalmente hardware,

[14] Los procesadores convencionales o microprocesadores son secuenciales (ejecutan una instrucción en cada ciclo de reloj). Sin embargo, actualmente la mayoría de estos procesadores incluyen juegos de instrucciones SIMD (*Single Instruction Multiple Data*) que permiten aumentar su rendimiento, sobre todo en operaciones multimedia (MMX o SSEx de Intel, 3DNow! de AMD, o AltiVec de Motorota).

lo más común suele ser abordar su construcción a partir de un diseño mixto hardware-software (HW/SW). De este modo se consigue combinar la potencia expresiva de los lenguajes de alto nivel y la flexibilidad del software, con la eficiencia del hardware.

En este tipo de configuraciones mixtas las tareas más complejas como las relacionadas con el procesamiento de imágenes de alto nivel, la extracción y clasificación de las características visuales, la gestión del control y las comunicaciones, etc. (componente software), suele llevarlas a cabo el microprocesador, mientras que determinados cálculos más simples correspondientes a funciones de más bajo nivel (componente hardware) se ejecutan simultáneamente y en paralelo en un dispositivo hardware de propósito específico.

Como ya se ha comentado con anterioridad cada tipo de VIPS debe cumplir distintos requisitos en función de su aplicación, anteponiéndose los más prioritarios a aquellos considerados menos relevantes en cada caso. Como no resulta sencillo conseguir el cumplimiento de varios requisitos de manera simultánea, ya que a veces éstos no son compatibles entre sí, se hace necesario en la mayoría de los casos alcanzar un cierto compromiso entre todos ellos. Además, debe tenerse en cuenta que una solución que teóricamente y a priori consiga balancear correctamente todos los requisitos, puede no resultar viable en la práctica. Actualmente, la única forma de comprobar que el diseño de un VIPS es viable y cumple con todos los requisitos inicialmente exigidos es llevar a cabo su implementación, lo que conlleva la adquisición de varios dispositivos y por lo tanto una considerable inversión económica. Si finalmente el diseño implementado no funciona como se esperaba, esta inversión habrá resultado inútil.

Cabe destacar el hecho de que entre los requisitos que típicamente guían el diseño de los VIPS no se suelan incluir ni la flexibilidad ni el grado de reutilización que serían deseables para estos sistemas, ya que ambos resultan incompatibles con el elevado grado de especialización necesario para conseguir VIPS eficientes.

A continuación, se describe el procedimiento que se sigue en la actualidad para la construcción de los VIPS. Como se verá, este proceso presenta una serie de deficiencias para las que se tratará de encontrar una solución en los apartados siguientes.

3.2. Procedimiento actual de desarrollo de VIPS: problemas asociados

De acuerdo con la siguiente definición ofrecida por el IEEE los VIPS pueden considerarse como sistemas intensivamente software.

> *"Un sistema intensivamente software es aquel en el que el componente software influye y contribuye de una manera esencial al diseño, construcción, desarrollo y evolución del sistema global." [IEEE-Std-1471].*

Definición 6. Sistema intensivamente software.

Sin embargo, a pesar de ello, éstos suelen construirse como sistemas muy centrados en el hardware. De hecho, la práctica habitual a la hora de construir un VIPS consiste en seleccionar, en primera instancia, la plataforma hardware sobre la que se llevará a

cabo su implementación. Para ello se siguen como únicos criterios (1) la experiencia previa del diseñador en la construcción de este tipo de sistemas y (2) las limitaciones impuestas por ciertos requisitos no funcionales tales como el coste máximo asumible, la velocidad de respuesta mínima exigida, etc. Una vez seleccionada la plataforma hardware correspondiente se procede a la implementación de la lógica software atendiendo al cumplimiento de los requisitos funcionales impuestos al VIPS.

La primera consecuencia negativa derivada del hecho de anteponer la elección del hardware a la implementación del software suele ser la adquisición de plataformas sobredimensionadas, tanto en coste como en capacidad de procesamiento, haciendo válido el dicho "más vale que sobre…". Además, la elección y adquisición de una plataforma hardware como primer paso en la construcción de un VIPS suele imponer el uso de determinados controladores (drivers) y librerías optimizadas para dicha plataforma lo que, en algunos casos, puede limitar enormemente la labor del programador (ver Figura 17).

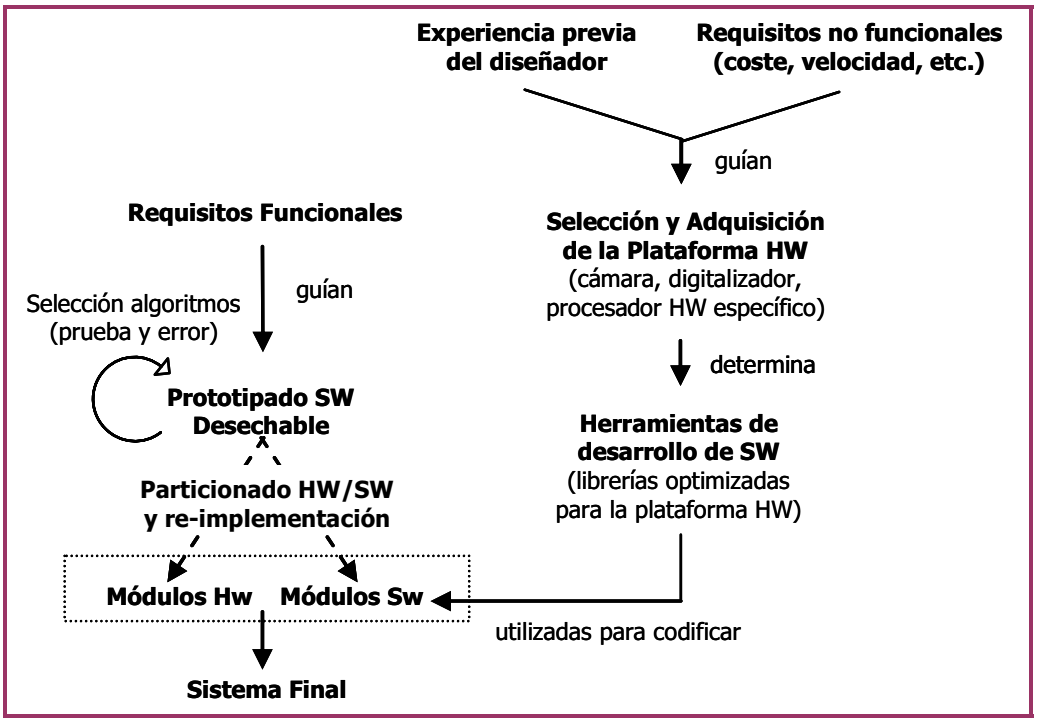


Figura 17. Fases de desarrollo de un VIPS siguiendo el enfoque tradicional.

Sin embargo, quizá la limitación más importante derivada de la subordinación del software al hardware, sea la dificultad de conseguir diseños flexibles y reutilizables. Por lo general, ante un problema específico los desarrolladores de VIPS prefieren implementar una solución eficiente y absolutamente a medida, antes que diseñar una más general con vistas a poder reutilizarla en la construcción de otros sistemas similares. En consecuencia, cada nuevo VIPS debe construirse prácticamente desde cero, aunque su lógica y su estructura sean casi idénticas a las de otros sistemas desarrollados previamente pero implementados sobre plataformas distintas a la seleccionada.

Una de las razones que podrían justificar la concepción hardware de los VIPS es que este tipo de aplicaciones suelen implementarlas expertos en visión, control y/o

automatización, con amplios conocimientos en programación pero, por lo general, ajenos al uso de metodologías propias del campo de la Ingeniería del Software. Por ello, en la mayoría de los casos, se procede directamente a la implementación del sistema, obviándose las fases de análisis y diseño, o bien llevándolas a cabo de forma poco sistemática. Además, no existe un procedimiento unificado, siendo cada empresa la que decide cómo desarrolla sus productos.

Ahora bien, puesto que los VIPS pueden considerarse sistemas intensivamente software, parece adecuado abordar su construcción utilizando alguna de las metodología[15] de desarrollo de software existentes para dar soporte a todas las fases de su ciclo de vida. Esta metodología podría servir, adicionalmente, para unificar los criterios de desarrollo de los distintos fabricantes de estos productos y lo que es más importante, permitiría aprovechar el conocimiento que se genera cada vez que se desarrolla un VIPS, ya que su diseño quedaría documentado de forma que podría ser reutilizado con posterioridad.

La implementación de los VIPS suele realizarse en dos etapas. La primera de ellas consiste en desarrollar un prototipo software que permita validar distintas combinaciones de algoritmos, generalmente seleccionados manualmente y de manera empírica (prueba y error) hasta dar con una solución válida (que cumpla con los requisitos funcionales del sistema), aunque ésta no resulte especialmente eficiente. Estos prototipos funcionales suelen implementarse utilizando lenguajes de muy alto nivel [MATLAB] [VISIQUEST] con los que es posible construir y validar aplicaciones de manera muy rápida e intuitiva. En la mayoría de los casos, una vez validado el prototipo inicial, éste se desecha siendo necesario re-implementar cada uno de los algoritmos seleccionados, bien en su versión software (haciendo uso de las librerías optimizadas para la plataforma hardware seleccionada en primera instancia), bien en su versión hardware (utilizando un lenguaje de descripción de hardware como VHDL). Así pues, la construcción de VIPS requiere explorar una gran cantidad de algoritmos y configuraciones Hw/Sw. De hecho, son varias las combinaciones de algoritmos que pueden satisfacer los requisitos funcionales impuestos al sistema; para cada una de estas soluciones pueden realizarse distintas particiones Hw/Sw y, a su vez, cada una de ellas puede implementarse utilizando distintas plataformas.

Como se comentó en el capítulo anterior, en la actualidad existen en el mercado una amplia variedad de herramientas que permiten realizar cada uno de los pasos involucrados en la implementación de un VIPS (prototipado funcional, particionado Hw/Sw, simulación y co-simulación, etc.). Sin embargo, ninguna de ellas es capaz de cubrir por sí sola todas las fases del ciclo de vida de estos productos siendo éstas además generalmente incompatibles entre sí. Así, como afirma Perrier [Perrier 04], "es *necesario desarrollar una nueva generación de aplicaciones que, de una parte, resuelva los vacíos existentes entre las herramientas actuales, y de la otra, permita la integración del software y del hardware de una forma más natural.*"

En el apartado siguiente se analizan las posibles soluciones a los problemas derivados del actual procedimiento de construcción de los VIPS.

[15] Utilizar una metodología de desarrollo de software conlleva documentar formalmente todas las fases que se deben seguir para desarrollar un sistema. En cada fase será necesario establecer, de manera precisa, cuáles son sus objetivos, qué resultados se precisan de las fases anteriores y cuáles producen. Para ello puede ser necesario elaborar cierta documentación especializada (formularios, diagramas, etc.) [PCM Encyclopedia].

3.2.1. Posibles soluciones a las deficiencias detectadas

De acuerdo con todo lo anterior, cabe preguntarse cómo y en qué medida puede resultar útil abordar la construcción de los VIPS desde una nueva óptica que permita mejorar su grado de flexibilidad y reutilización, teniendo en cuenta las restricciones que suelen guiar su construcción (eficiencia, bajo coste, alta fiabilidad, espacio y peso reducidos, corto tiempo de desarrollo, etc.) ya que, como indica W. S. Newman refiriéndose al diseño de productos mixtos Hw/Sw, "*el gran avance en este campo sólo llegará cuando se consiga dotar a estos sistemas de la suficiente flexibilidad y generalidad como para que puedan adaptarse a los continuos cambios del mercado sin necesidad de ser rediseñados cada vez desde cero*" [Newman 00].

Aunque cada vez más se tiende a desarrollar hardware flexible y de propósito general, es sin duda el componente software el más maleable en este tipo de productos mixtos Hw/Sw. Por ello, en este apartado se analiza cómo un enfoque de Ingeniería del Software puede ayudar a mejorar el proceso y los resultados obtenidos a la hora de construir este tipo de aplicaciones.

En el cuadro siguiente se recogen de manera esquemática los inconvenientes asociados al procedimiento actual de desarrollo de VIPS, indicándose en qué sentido debería buscárseles solución (ver Tabla 3).

Problema	Implicaciones	Solución
Especialización de las soluciones para conseguir eficiencia	Sw subordinado al Hw por lo que las soluciones resultan poco flexibles y reutilizables. Cada nuevo sistema debe construirse desde cero y los resultados parciales obtenidos en cada etapa no se aprovechan en las siguientes.	Desarrollar una nueva metodología centrada en el Sw que promueva la creación de soluciones más flexibles, retrasando todo lo posible el momento de la elección y adquisición del Hw. Deberá fomentarse al máximo la reutilización tanto del conocimiento como de los recursos generados, favoreciendo el prototipado evolutivo.
Eficiencia por medio de implementaciones muy dependientes del HW.		
Difícil reutilización del conocimiento generado.		
Uso de prototipos desechables		
Uso de múltiples herramientas durante las distintas fases del desarrollo, la mayoría de ellas muy dependientes del Hw y cuyo uso requiere amplios conocimientos de programación a medio y bajo nivel.	La integración de estas herramientas resulta difícil tanto en vertical (herramientas que cubren las distintas etapas del proceso), como en horizontal (utilizadas de manera conjunta durante una fase del proceso).	Desarrollar una nueva herramienta que dé soporte al ciclo de vida de los VIPS definido en la metodología anterior, integrando las herramientas ya existentes de manera que sean compatibles entre sí, y proporcionando un entorno de programación visual de alto nivel, sencillo y de manejo intuitivo.

Tabla 3. Problemas asociados al procedimiento actual de desarrollo de los VIPS.

Algunos de los problemas que se acaban de exponer están relacionados con carencias derivadas del actual procedimiento de desarrollo de los VIPS, mientras que otros lo están con las limitaciones impuestas por las herramientas disponibles. Los primeros dan lugar a desarrollos poco flexibles y reutilizables y su solución puede abordarse desde dos puntos de vista alternativos: (1) la construcción de VIPS de propósito más general, o (2) la definición de una nueva metodología más general, válida para construir los distintos tipos de VIPS.

La primera opción persigue el desarrollo de VIPS totalmente flexibles y reutilizables, capaces de realizar una amplia variedad de tareas cada una de ellas bajo requisitos muy diferentes, lo que puede implicar la necesidad de interactuar con distintos dispositivos y/o el empleo de distintas plataformas de desarrollo. Sin embargo, esto no sólo requeriría un esfuerzo colosal sino que, muy probablemente, complicaría sobremanera el manejo, configuración y mantenimiento de este tipo de sistemas, aumentando ineludiblemente su coste y probablemente reduciendo su rendimiento.

La segunda opción consistente en generalizar la metodología que se sigue a la hora de construir estos sistemas, en lugar de los sistemas en sí mismos, resulta a todas luces mucho más práctica y asequible. Para ello es necesario encontrar un procedimiento de validez general para desarrollar distintos tipos de VIPS, en lugar de abordar su construcción como si se tratara de productos únicos y totalmente distintos entre sí, esto es, evitando construir cada producto con un procedimiento a medida que, en general, no resulta válido para construir otros VIPS.

Para diseñar esta nueva metodología de desarrollo de VIPS en primer lugar se debe definir el ciclo de vida de estos productos, esto es, las etapas necesarias para construirlos, mantenerlos y hacerlos evolucionar cuando el mercado así lo requiera. La nueva metodología deberá cubrir todas estas etapas, favoreciendo la transición de unas a otras y permitiendo validar los resultados parciales obtenidos en cada paso, a fin de detectar cuanto antes los errores en los que se pudiera haber incurrido. Además, para que esta metodología sea efectiva resulta esencial seleccionar y/o desarrollar las herramientas adecuadas para dar soporte a todas las fases de su ciclo de vida. Por último, la metodología que se diseñe deberá ser la base de una comunicación efectiva entre todos los agentes involucrados en el proceso de construcción de los VIPS (analistas, diseñadores, programadores, usuarios, etc.).

En cuanto al diseño del ciclo de vida de los VIPS, hay que tener en cuenta la naturaleza especial de estos productos ya que se trata de sistemas mixtos Hw/Sw. Dado que se pretende centrar la metodología en el software para así favorecer la flexibilidad y la reutilización de estos productos, las etapas del ciclo de vida serán las clásicas del desarrollo de cualquier producto software, aunque habrá que añadir ciertas modificaciones para incorporar al proceso la posibilidad de evaluar distintas implementaciones del producto sobre una variedad de plataformas Hw a fin de seleccionar aquella que mejor se ajuste a los requisitos impuestos al sistema.

En la literatura es posible encontrar distintos tipos de metodologías de desarrollo de software. A continuación se muestra una posible clasificación[16] de las mismas atendiendo al tipo de proceso que siguen sus ciclos de vida:

[16] Esta clasificación no es única. También es posible clasificar las metodologías atendiendo a si son estructuradas o no, si están orientadas a procesos/datos/objetos, etc.

- Metodologías descendentes (*top-down*) en las que se busca manejar la creciente complejidad del software a través de abstracciones que permitan razonar acerca de la estructura y la lógica de los sistemas que se desea implementar. A partir de estas abstracciones se elaboran las soluciones que se van refinando y concretando sucesivamente hasta llegar a la implementación final. Este tipo de metodologías dan lugar a lo que se denomina desarrollo "para reutilización", siendo posible clasificarlas como sigue:
  - Metodologías en cascada (*waterfall*): Su ciclo de vida consta de una serie de etapas entre las que tradicionalmente se incluyen las siguientes: análisis, diseño, implementación, integración y mantenimiento, cada una de las cuales requiere un esfuerzo ingente de documentación. Estas etapas se realizan secuencialmente y de modo que hasta que no se completa una etapa no se pasa a la siguiente, aunque es posible volver atrás en el proceso para realizar algunas modificaciones (retroalimentación). Este tipo de metodologías suelen contemplar la posibilidad de validar los resultados intermedios utilizando prototipos ya sean evolutivos o desechables.
  - Metodologías en espiral [Boehm 88]: Se trata de un modelo inspirado en el anterior en el que se repiten todas las etapas del ciclo de vida tradicional del software de manera iterativa. Al finalizar cada etapa se produce un prototipo que, en cada iteración, se aproximará más a la solución final. Las metodologías en espiral suelen considerarse más flexibles aunque también más costosas de seguir que las anteriores.

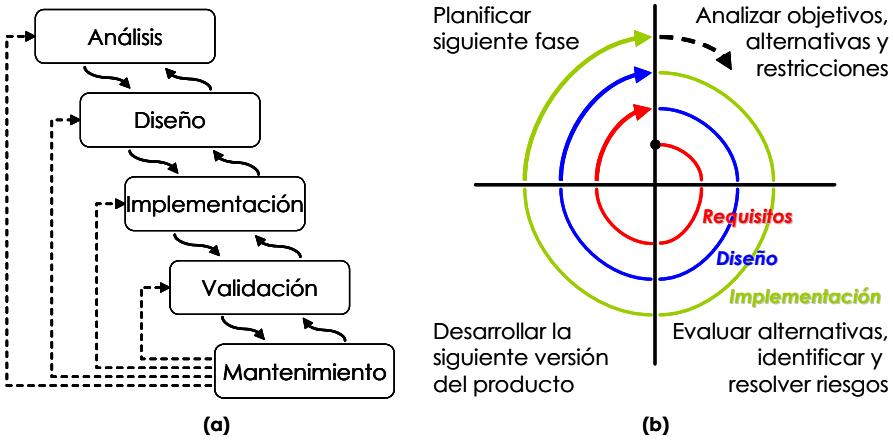


Figura 18. Ciclo de vida (a) en cascada y (b) en espiral.

- Metodologías ascendentes (*bottom-up*), en las que se elaboran las soluciones partiendo de pequeñas "piezas" de software, ya sean éstas de fabricación propia o adquiridas a terceros, que se van agregando de manera incremental hasta alcanzar la solución final, siguiendo un proceso similar al de una típica cadena de montaje industrial. Este tipo de metodologías dan lugar a lo que se denomina desarrollo "con reutilización", siendo el desarrollo de software basado en componentes (DSBC) uno de los máximos exponentes de este enfoque.

Por lo que sabemos, hasta la fecha no se ha descrito una metodología que aborde de manera integral el desarrollo de los VIPS, ni siquiera de un subconjunto representativo de ellos como por ejemplo los SIVA. La mayoría de los trabajos publicados con relación al desarrollo de este tipo de sistemas se limitan a proponer mejoras algorítmicas de algunas funciones o nuevas configuraciones hardware basadas en dispositivos reconfigurables de bajo coste [Benkrid 04] [Qureshi 05]. Sin embargo, las escasas contribuciones en las que se aborda el diseño de estos sistemas desde un punto de vista más amplio, se limitan a generalizar las tareas relacionadas con el procesamiento de las imágenes [Everding 94], dejando de lado otras de vital importancia como son el control, la sincronización y la comunicación entre los distintos procesos y dispositivos físicos del sistema.

Recientemente se han descrito algunas propuestas metodológicas que abordan el desarrollo de sistemas que guardan un cierto paralelismo con los VIPS (diseños distribuidos, orientados al hardware, con restricciones de tiempo real, etc.), como por ejemplo los sistemas de tele-operación [Ortiz 05], o de comunicaciones móviles [Muthig 04]. Sin embargo, en la mayoría de los casos, el núcleo software de estos sistemas es bastante más simple y reducido que en los VIPS, por lo que los resultados alcanzados resultan difíciles de extrapolar.

La Ingeniería del Software proporciona distintos enfoques válidos a priori para resolver cada uno de los problemas detectados en el procedimiento actual de construcción de VIPS. Sin embargo, como veremos en el último apartado de este capítulo en el que se definirá una nueva metodología, será necesario integrar varios de estos enfoques con el fin de solventar cada uno de los problemas de la manera más adecuada en cada caso. Así, por ejemplo, se verá cómo la falta de flexibilidad de los diseños actuales puede mejorarse mediante la descripción de la arquitectura software genérica de los VIPS, y cómo la incompatibilidad entre las distintas librerías que pueden requerirse durante la implementación puede resolverse utilizando un modelo de componentes software adecuado.

Integrar todos estos enfoques en una metodología única no resultará sencillo dada la falta de consenso existente respecto a qué modelos y notaciones emplear durante cada una de las etapas del ciclo de vida de los productos software. A esto habrá que sumar la dificultad de validar formalmente los modelos que se construyan y la de materializar el código final de las aplicaciones a partir de ellos. De hecho, todas estas dificultades asociadas a la integración de varias metodologías bajo un marco único de desarrollo hacen que en la actualidad éste sea un problema aún no resuelto.

En el siguiente apartado se realiza un repaso de algunos de los paradigmas actuales de desarrollo de software que pueden contribuir a mejorar el proceso de desarrollo de los VIPS, y que posteriormente se integrarán en la nueva metodología que se propone al final del capítulo.

3.3. Paradigmas actuales de desarrollo de software

En el contexto de las ciencias de la computación un paradigma es una forma de hacer, esto es, el enfoque que se adopta para resolver los problemas de desarrollo del software. No debe confundirse el término paradigma con el de metodología; de hecho una metodología puede basarse en uno o varios paradigmas y distintas

metodologías pueden sustentarse en el/los mismo/s paradigma/s.

Entre los paradigmas de desarrollo de software que actualmente cuentan con un mayor grado de aceptación cabe destacar algunos como los paradigmas de desarrollo de software orientados a objetos (OOSD, *Object-Oriented Software Development*), a aspectos (AOSD, *Aspect-Oriented Software Development*), los basados en componentes (CBSD, *Component-Based Software Development*), los basados en arquitecturas dirigidas por modelos (MDA, *Model-Driven Architectures*), los basados en líneas de productos (SPL, *Software Product Lines*) y el de Programación Generativa (PG).

A continuación se comentan aquellos que, por su utilidad para resolver los problemas planteados en el apartado anterior, se han seleccionado para formar parte de la nueva metodología que se propone.

3.3.1. Desarrollo de software basado en componentes (DSBC)

Desde sus comienzos, la Ingeniería del Software ha perseguido la idea de construir sistemas a partir de piezas ya existentes siguiendo el modelo de ensamblaje que utilizan muchas factorías industriales [Greenfield 04]. Para equiparar el proceso industrial con el de desarrollo de software sería necesario que los productos que han de ensamblarse estuvieran perfectamente especificados siguiendo unos ciertos estándares, tal y como ocurre en la industria. Sin embargo, los intentos de crear "factorías software" han fracasado reiteradamente por varios motivos. En primer lugar, el software en general no es un producto fácil de estandarizar, ya que existe una enorme variedad de metodologías de desarrollo, notaciones, lenguajes de programación, etc. Aunque la comunidad científica ha realizado varias propuestas tratando de definir estándares, por lo general éstos no han encontrado un apoyo decidido por parte de la industria; quizá la notación UML (*Unified Modeling Language*), estandarizada por el OMG (*Object Management Group*) sea una de las pocas excepciones a esta afirmación. Esta ausencia de estándares favorece que el software que se construye sea de naturaleza muy diversa de modo que los distintos productos obtenidos resultan, por lo general, difíciles de integrar entre sí.

Por otra parte, la creciente demanda de software cada vez más complejo obliga a las empresas a reducir a toda costa los precios y el tiempo de salida al mercado de sus productos a fin de mantener su competitividad. Para conseguir estos objetivos resulta imprescindible reutilizar al máximo cualquier software ya desarrollado, bien por la propia empresa en proyectos previos, bien el disponible de fuentes externas.

En este contexto, K. Wallnau [Wallnau 01] plantea dos visiones extremas acerca del proceso de desarrollo de software: la primera y quizá más cercana a la realidad actual es bastante "pesimista", mientras que la segunda, mucho más "optimista", muestra los ideales a los que se debería tender para aproximarnos al concepto de "factoría software".

La "visión pesimista" surge como consecuencia de considerar que:

- El software está compuesto de elementos (componentes) vistos como cajas negras que ocultan muchas de sus propiedades haciendo difícil acceder a la información que contienen y a su comportamiento interno.

- Cada fabricante utiliza su propia notación para definir sus esquemas de diseño y por lo tanto éstos no suelen ajustar a ningún estándar.
- Las herramientas disponibles sólo permiten el desarrollo de productos para una determinada plataforma, limitando los lenguajes de programación a aquellos que son compatibles con ella. Por ejemplo, el entorno .NET de Microsoft® permite integrar componentes implementados utilizando varias de las herramientas de programación existentes para esta plataforma (Visual Basic, Visual C++, etc.), si bien estos componentes no son directamente compatibles con los desarrollados utilizando CORBA, EJB, etc.
- Los componentes no suelen estar bien documentados lo que dificulta enormemente tanto su localización como su validación y, en consecuencia, su reutilización.

En contraposición, la "visión optimista" presupone que se cuenta con:

- Elementos software (componentes) definidos mediante interfaces claras y correctamente especificadas a modo de contratos.
- Esquemas de diseño estándar en los que los anteriores componentes encajan de manera sencilla.
- Repositorios de componentes ya implementados y listos para ser reutilizados y en los que es sencillo encontrar el componente que mejor se ajusta a las necesidades de un determinado proyecto.
- Herramientas de desarrollo estandarizadas que estén orientadas a la fabricación de soluciones mediante técnicas de composición y que permitan la integración de componentes heterogéneos (escritos utilizando distintos lenguajes de programación, para distintas plataformas, etc.).

Tratando de acercarse a esta visión optimista, distante aún de convertirse en realidad, ha surgido un nuevo paradigma de desarrollo de software basado en componentes. Éste se ha visto impulsado por la aparición de nuevos modelos tales como .NET, EJB o CCM, que se comentarán más adelante y que están teniendo un fuerte apoyo por parte de algunas de las más importantes empresas y organismos relacionados con el mundo del software (por ejemplo, Microsoft®, OMG, etc.). La principal limitación con la que se enfrenta este nuevo paradigma reside, nuevamente, en la falta de consenso respecto a los estándares que deben guiar el desarrollo de software. De hecho, como se analiza en el siguiente punto, ni siquiera existe acuerdo respecto a lo qué es o no un componente.

3.3.1.1. Definición de Componente Software

Etimológicamente, el término "componente" procede de la palabra latina *cumponere* que significa "poner junto" (cum "junto" y ponere "poner").

Como puede fácilmente deducirse de la consulta de la literatura especializada, existe diversidad de opiniones sobre lo que puede considerarse o no un Componente Software (CS). Esta disparidad de opiniones ha dado lugar a numerosas definiciones de este término, algunas de las cuales quedan recogidas a continuación.

"Un componente es una unidad binaria de composición que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio" [Szyperski 98]

"Un componente es una implementación opaca de una determinada funcionalidad (caja negra) que puede ensamblarse con otros componentes siguiendo las reglas establecidas por el modelo de componentes adoptado" [Bachmann 00].

"Un componente es una implementación que: (a) realiza un conjunto de funciones relacionadas, (b) puede ser independientemente desarrollado, entregado e instalado, (c) tiene un conjunto de interfaces para los servicios proporcionados y otro para los servicios requeridos, (d) permite tener acceso a los datos y al comportamiento sólo a través de sus interfaces, (e) opcionalmente admite una configuración controlada." [IBM-WEBSPHERE].

Definición 7. Componente Software según distintos autores.

Como puede observarse en las definiciones anteriores, todas ellas coinciden en el hecho de que los CS son unidades de implementación concebidas para ser reutilizadas, pudiendo ensamblarse con otros CS que, aún no siendo necesariamente de la misma procedencia, deben ser compatibles entre sí. La compatibilidad entre los CS requiere que éstos sean conformes con un determinado modelo de componentes que establecerá bajo qué condiciones pueden ensamblarse. El proceso de ensamblaje requiere que cada componente defina en su interfaz tanto los servicios que demanda (requisitos), como aquellos que ofrece al resto de componentes con los que interactúa.

El Desarrollo de Software Basado en Componentes (DSBC) puede considerarse una evolución del paradigma Orientado a Objetos (OO). La diferencia entre ambos enfoques parece no estar muy clara y a veces se tiende a confundir el concepto de componente con el de objetos. He aquí algunas claves para entender las diferencias entre ambos conceptos:

- La interfaz de un objeto exhibe sólo los servicios que éste proporciona, mientras que la interfaz de un componente debe incluir además los servicios requeridos para su correcto funcionamiento.
- Existen diversas formas de interactuar con un componente, mientras que la única forma de interactuar con un objeto es mediante la invocación de alguno de sus métodos públicos (paso de mensajes).
- Un componente puede contener uno o más objetos, clases, rutinas, e incluso otros componentes.
- Los componentes de una misma aplicación pueden estar escritos en distintos lenguajes de programación, incluyendo lenguajes orientados a objetos.
- Los componentes suelen estar empaquetados de manera más robusta que los objetos.

Según esto, un componente puede ser desde una subrutina de una librería matemática hasta una clase Java, un paquete Ada, un objeto COM, un JavaBean, o incluso una aplicación independiente con la que es posible interactuar desde otra a través de una determinada interfaz.

Es posible realizar distintas clasificaciones de los componentes atendiendo a su funcionalidad, visibilidad (caja negra, gris o blanca), coste (*freeware*, *shareware* o *fullprice*), modos de interacción, ámbito de aplicación, granularidad, etc. En particular, la granularidad y el ámbito de aplicación de un componente determinan en gran medida su utilidad, aplicabilidad y rendimiento. Así, por lo general, los componentes de grano grueso y de propósito general resultan de utilidad para la construcción de una amplia variedad de aplicaciones y consiguen un ahorro sustancial en los tiempos de desarrollo, dado el alto grado de reutilización que supone su incorporación. Sin embargo, este tipo de componentes suele proporcionar una funcionalidad a veces no requerida que puede complicar su integración y mermar la eficiencia global del sistema.

Por el contrario, cuanto más pequeño y específico es un componente mejor se ajustará a los requisitos que de él se demandan y más sencilla y eficiente resultará su integración en las distintas aplicaciones. Ahora bien, dada su especificidad, serán pocos los sistemas que puedan reutilizar este tipo de componentes así como tampoco supondrán un gran ahorro en el tiempo de desarrollo dado el poco código que se reutiliza.

3.3.1.2. Desarrollo de Software Basado en Componentes: Etapas

Tradicionalmente los ingenieros del software han seguido un enfoque descendente (*top-down*) para el desarrollo de sus sistemas [Page-Jones 88]. Sin embargo, el DSBC apuesta por un enfoque ascendente (*bottom-up*) en el que los productos se construyen mediante el ensamblaje e integración de componentes software preexistentes.

A diferencia de las metodologías top-down cuyo ciclo de vida comprende las etapas de análisis, diseño e implementación, el DSBC divide el proceso en las siguientes fases [Brown 99] [Iribarne 03]:

- Selección y evaluación de los componentes software que deberán satisfacer las necesidades del usuario y ajustarse al esquema de diseño de la aplicación.
- Adaptación de los componentes cuando sea necesario
- Ensamblaje de los componentes como parte de la solución final.
- Evolución del sistema si el usuario lo requiere.

En la siguiente figura se muestra un resumen comparativo de las fases comprendidas en el ciclo de vida del software siguiendo el enfoque tradicional y el basado en componentes.

	Etapas de desarrollo del software			
Ciclo de vida clásico (top-down)	**Análisis**	**Diseño**	**Implementación**	**Mantenimiento**
Ciclo de vida DSBC (bottom-up)	**Selección y evaluación**	**Adaptación**	**Ensamblaje**	**Evolución**

Figura 19. Etapas de los ciclos de vida top-down y DSBC

A continuación se comentan brevemente cada una de las etapas comprendidas en el ciclo de vida del software según el enfoque del DSBC.

- **Selección y evaluación de los componentes**

Esta etapa permite determinar qué componentes, de entre los disponibles, se ajustan mejor a las necesidades de una determinada aplicación. Para ello es necesario en primer lugar encontrarlos y, en segundo, evaluarlos a fin de comprobar que efectivamente cumplen con lo que se espera de ellos.

Encontrar el componente adecuado requiere establecer con precisión los criterios de búsqueda. Entre éstos, suelen incluirse los relativos a la funcionalidad (servicios que debe proporcionar) aunque también es común incluir otros relacionados con el coste, el fabricante, compatibilidad con una determinada plataforma, etc. Esta fase puede resultar tediosa debido a que la abundante información disponible no siempre está correctamente clasificada. Además, los resultados de la búsqueda pueden resultar difíciles de cuantificar [Iribarne 02].

La fase de evaluación tampoco resulta sencilla, si bien existen algunas técnicas relativamente maduras que pueden aplicarse al efecto [ISO/IEC-9126 91]. Estas técnicas de evaluación se basan en encuestas realizadas a los clientes y en la construcción de prototipos con los que validar, de manera efectiva, el comportamiento de los componentes seleccionados.

- **Adaptación de los componentes**

En el momento de crear nuevos componentes, los desarrolladores hacen ciertos supuestos sobre los sistemas en los que éstos se integrarán. Sin embargo, cuando estas suposiciones no se ajustan a las necesidades del cliente, como suele ocurrir, éste deberá realizar ciertas adaptaciones de forma que los componentes puedan encajar en el sistema. El procedimiento de adaptación depende de la accesibilidad del componente. Cuando éste se adquiere como una "caja negra" habrá que adaptar el sistema para permitir que encaje el componente, mientras que si el componente seleccionado es accesible, ya sea porque es de fabricación propia o porque se ha adquirido con ciertos privilegios para poder modificarlo, puede resultar más conveniente adaptar el componente al sistema.

Las adaptaciones que puede requerir un componente pueden ser de tres tipos: para limitar su funcionalidad, para extenderla, o para hacerlo compatible (con otros componentes, con la plataforma, etc.). Uno de los mecanismos de adaptación más utilizados consiste en construir un componente de tipo *wrapper* que envuelva al que se quiere adaptar de forma que lo limite, extienda, o modifique como convenga (ver Figura 20).

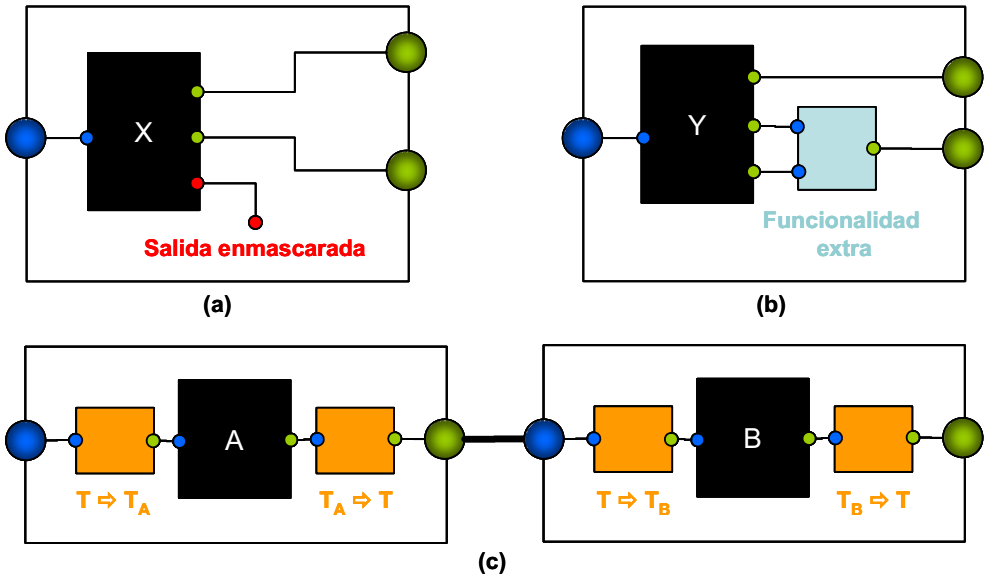


Figura 20. Mecanismos de adaptación de componentes

(a) Wrapper que oculta parte de la funcionalidad del componente X, (b) Wrapper que añade funcionalidad a la proporcionada por el componente Y, (c) Conexión de dos wrappers que encapsulan sendos componentes A y B que utilizan tipos de datos no compatibles: T_A y T_B. Ambos wrappers utilizan datos de tipo T que interiormente se transforma al tipo que es capaz de manejar cada componente interno.

➢ Ensamblaje o integración de los componentes

Como ya se ha comentado con anterioridad, el proceso de ensamblaje requiere de la existencia de un *modelo de componentes*, entendido como un conjunto de reglas que establecen qué componentes se pueden conectar con qué otros y cómo se comunican éstos entre sí. Dado que los componentes se comunican a través de sus interfaces, será necesario utilizar un lenguaje de definición de interfaces (IDL) que permita definirlas de una manera precisa. Además, cuando se precise conectar dos componentes situados en distintas localizaciones, será necesario proporcionar los mecanismos adecuados para su comunicación remota. Todo ello requiere además contar con una infraestructura o middleware que permita realizar el proceso de ensamblaje de los componentes. Entre los entornos *middleware* más utilizados cabe destacar algunos basados en tecnologías ORB (Object Request Broker) como .NET, EJB o CCM, que se comentarán más adelante.

➢ Evolución del sistema

A priori pudiera parecer que los sistemas construidos a partir de componentes deberían poder evolucionar de manera sencilla como ocurre en los procesos industriales de montaje en los que la sustitución de piezas o su actualización con otras más modernas resulta una tarea casi trivial. Sin embargo, cuando se trata de productos software, la sustitución de un componente por otro suele ser una tarea compleja y tediosa ya que además de que el nuevo componente debe someterse a un proceso minucioso de validación para comprobar que se ajusta a lo que se espera de él, puede ser necesario plantear una modificación drástica de la estructura del sistema para permitir su integración. Adicionalmente, debe tenerse en cuenta que la evolución de un sistema basado en componentes depende en gran medida de cómo los fabricantes decidan evolucionar dichos componentes. De hecho, puede ocurrir

que el fabricante no esté interesado en desarrollar una nueva versión del componente para cubrir la demanda de un cliente puntual o incluso que decida crear una nueva versión del producto dejando sin soporte la versión previa, lo que obligará al cliente a decidir si cambia de versión o de proveedor.

3.3.1.3. Modelos de Componentes Software comerciales

Un modelo de componentes define un conjunto de estándares y convenciones para el desarrollo, utilización y evolución de un conjunto de componentes [Crnkovic 02]. En este apartado se describen los modelos de componentes comerciales más utilizados en la actualidad tales como COM, DCOM, y .NET de Microsoft®, Enterprise JavaBeans (EJB) de Sun Microsystems®, o el modelo de componentes CORBA (CCM) propuesto por el OMG (Object Management Group).

- **Modelos de componentes de Microsoft®: COM, DCOM y .NET**

COM (*Component Object Model*) es el modelo de componentes comercializado por Microsoft® que proporciona un estándar para la definición de componentes en el que se incluye un lenguaje para especificar sus interfaces (MIDL, Microsoft® *Interface Description Language*) así como los mecanismos de comunicación entre los componentes a través de éstas. Este estándar es independiente del lenguaje de programación si bien requiere el uso de plataformas Microsoft®. Para COM, toda comunicación entre componentes se realiza a través de la invocación de alguna de las operaciones definidas en sus interfaces. De este modo, el usuario sólo necesitará invocar la operación que desea realizar, siendo transparente para él la ubicación del componente que le responde desde el mismo proceso/máquina u otro/a diferente.

DCOM surge como una evolución de COM para dar soporte a sistemas distribuidos. Esta tecnología presenta varias ventajas frente a su predecesora. En primer lugar, DCOM simplifica el desarrollo de nuevos componentes ya que genera de forma automática la implementación de las diversas interfaces necesarias para gestionar aspectos relativos al tiempo de vida del componente, su persistencia, las referencias a otros componentes, etc., que en el caso de COM deben implementarse manualmente. Además, DCOM proporciona un modelo más simple y robusto para registrar e instalar nuevos componentes, gracias al cual es posible llevar a cabo un riguroso control de versiones de las clases asociadas a cada paquete, lo que facilita enormemente el mantenimiento de las aplicaciones.

.NET es la apuesta más reciente de Microsoft® en el campo del desarrollo de software basado en componentes. Además de un modelo de componentes, .NET puede considerarse como un marco (*framework*) completo de desarrollo, en el que se ha conseguido eliminar la restricción de los modelos COM/DCOM, sólo válidos para plataformas Microsoft®.

El creciente desarrollo de aplicaciones de comercio electrónico (*e-commerce*) y la aparición de nuevas plataformas hardware (por ejemplo PDAs), hace necesario desarrollar software que permita acceder a estas aplicaciones desde distintas plataformas. El framework .NET, en respuesta a estas demandas, permite la interoperabilidad entre plataformas y aplicaciones, mientras que sus predecesoras COM/DCOM sólo soportan la interoperabilidad entre componentes.

- **Enterprise JavaBeans (EJB)**

Un JavaBean es un componente reutilizable del que puede hacerse uso desde cualquier entorno de desarrollo de aplicaciones Java; de hecho, los JavaBeans se implementan utilizando clases Java. Un JavaBean puede tener cualquier grado de complejidad y tener o no una representación gráfica. Por ejemplo, un JavaBean gráfico para modelar una ventana de mensajes resulta relativamente sencillo, mientras que un JavaBean tan complejo como el que modela el acceso a una base de datos puede no tener una representación gráfica asociada. Para incorporar un JavaBean a cualquier aplicación basta con "pegarlo" en un contenedor (objeto de la clase Java *Container*).

La tecnología Enterprise JavaBeans (EJB), definida como parte de la plataforma J2EE desarrollada por Sun Microsystems en colaboración con un consorcio de empresas (IBM, Oracle o BEA, etc.), proporciona una arquitectura para el desarrollo e implementación de aplicaciones basadas en componentes. Al contrario de lo que ocurre en los modelos de Microsoft®, el desarrollo de aplicaciones utilizando EJB sólo puede realizarse utilizando el lenguaje de programación Java lo que, como contrapartida, hace que estas aplicaciones sean independientes de la plataforma. Para utilizar un componente EJB, éste debe asociarse a un EJB Container (EJBC). A diferencia de los objetos *Container* en los que se ejecutan los JavaBeans, los EJBC constituyen el middleware responsable de implementar el contrato entre el componente y la arquitectura J2EE, especificando las características del entorno de ejecución, así como de proporcionar los servicios de seguridad, concurrencia, gestión de transacciones, gestión de nombres, etc.

- **Common Object Request Broker Architecture (CORBA)**

CORBA es un estándar desarrollado por el OMG (*Object Management Group*) que ofrece una arquitectura y una infraestructura, abiertas e independientes del fabricante, para la comunicación entre aplicaciones a través de la red [CORBA]. Esta propuesta aborda la interoperabilidad entre componentes en sistemas distribuidos y heterogéneos de manera totalmente transparente para el programador. Esto es, cualesquiera dos programas basados en CORBA podrán comunicarse entre sí, con independencia de sus fabricantes, del tipo de máquinas y sistemas operativos en los que se estén ejecutando, y de los lenguajes de programación empleados para su implementación.

CORBA forma parte de un modelo más extenso (OMA, *Object Management Architecture*) en el que se define un contexto abstracto de diseño donde los componentes pueden operar e interactuar de manera segura.

Los componentes CORBA son paquetes binarios que pueden residir en cualquier parte de la red de modo que cuando un componente quiere acceder a alguno de los servicios que ofrece otro componente, podrá hacerlo sin importarle dónde esté ubicado o en qué lenguaje de programación esté implementado. Esto es posible gracias a que la interfaz externa de los componentes se especifica utilizando un lenguaje de definición de interfaces (IDL) propio de CORBA que es independiente del lenguaje de programación empleado para implementar su código interno. El encargado de mediar entre los componentes CORBA, realizando la traducción y el transporte de los datos que éstos intercambian, es el denominado ORB (*Object*

Request Broker), pieza fundamental de la arquitectura CORBA. De este modo, CORBA puede considerarse como un middleware que permite conectar clientes y servidores heterogéneos tanto en lo que se refiere a los lenguajes de programación en los que están implementados como a la plataforma sobre la que se ejecutan.

La siguiente tabla resume de manera comparativa las características de los tres modelos de componentes expuestos con anterioridad.

	COM/DCOM, .NET	**EJB**	**CORBA**
Desarrolladores	Microsoft®	Sun Microsystems IBM, BEA, etc.	OMG (*Object Management Group*)
Componentes	Módulos con múltiples clases u otras implementaciones	Módulos que pueden contener múltiples clases	Módulos que contienen cualquier implementación
Interfaz	OLE IDL (interfaces como una colección de funciones)	Lenguaje Java	IDL propio de OMG
Conexión	Vía punteros a interfaces	Vía eventos y escuchadores	Vía IDL
Mecanismos de variabilidad	Genericidad, agregación, etc.	Herencia y agregación	Herencia y agregación
Plataforma	Windows (.NET es multi-plataforma)	Multi-plataforma, cualquier sistema con máquina virtual Java (JVM)	Multi-plataforma
Lenguaje de implementación	Cualquier lenguaje de una lista dada	Java	Cualquier lenguaje
Mecanismos de distribución	DCOM, Internet	EJB, Internet, RMI (Remote Method Invocation)	Un ORB de un proveedor

Tabla 4. Comparativa COM/DCOM, .NET, EJB y CORBA.

3.3.1.4. Componentes COTS

Los componentes COTS (*Commercial Off-The-Shelf*) son componentes software diseñados para ser vendidos en múltiples copias a numerosos clientes sin atender las necesidades particulares de ninguno de ellos. Los fabricantes de estos componentes son los únicos responsables de su actualización y mantenimiento sin que los clientes tengan ningún control sobre los cambios que sobre ellos se realizan. Por lo general, estos productos se venden sin su código fuente asociado y en algunos casos, además, carecen de una documentación técnica completa y/o precisa.

Si bien los procesos de selección, integración y mantenimiento de los sistemas basados en componentes COTS resultan más complejos que cuando se utilizan componentes de desarrollo propio [Iribarne 04], por lo general, los primeros consiguen

una mayor reducción de los tiempos de desarrollo y el coste final de los productos, por lo que la política de "comprar en lugar de construir" resulta cada vez más popular entre las empresas desarrolladoras de software. La Tabla 2 resume algunas de las ventajas e inconvenientes de los componentes COTS frente a los de desarrollo propio [Albert 02].

Componentes de desarrollo propio	Componentes COTS	Implicaciones del uso de componentes COTS
La persona encargada de su implementación conoce la arquitectura del sistema y el modelo de componentes subyacente, por lo que los construye para que sean compatibles con ellos.	Los fabricantes de componentes COTS desconocen la arquitectura de los sistemas concretos en los que posteriormente éstos se integrarán, aunque asumen algunas generalidades.	Puesto que los componentes COTS no suelen ser modificables, debe ser el sistema, y en particular su arquitectura, la que se acomode para permitir su integración.
Suministran exactamente la funcionalidad requerida ya que se construyen a medida	Por lo general proporcionan más funcionalidad de la estrictamente requerida por el cliente.	La funcionalidad extra debe ser convenientemente ocultada.
Evolucionan cuando y como convenga ya que el código fuente está disponible y puede ser fácilmente modificado.	Es el vendedor el único que decide cómo y cuándo éstos evolucionan ya que por lo general el usuario no tiene acceso al código fuente o, si lo tiene, su modificación no está exenta de riesgos.	A veces los clientes se ven obligados a actualizar los componentes sin desearlo. Otras, el vendedor no actualiza el componente como desearía el usuario.
Están ampliamente documentados y pueden validarse fácilmente.	Pueden no estar completa y/o correctamente documentados.	La validación debe ser cuidadosa y exhaustiva.

Tabla 5. Componentes de desarrollo propio frente a componentes COTS.

3.3.2. Desarrollo basado en Líneas de Productos Software (LPS)

Hoy en día, la complejidad y el tamaño de los productos software está creciendo rápidamente y los clientes demandan productos cada vez de más calidad y más ajustados a sus necesidades. Para cubrir esta demanda la mayoría de las organizaciones productoras de software desarrollan y mantienen más de un producto simultáneamente. Esto es aplicable tanto a las empresas que desarrollan sistemas a medida para clientes individuales, como para aquellas que desarrollan productos para un mercado más amplio. Incluso las organizaciones que dicen vender un único producto invierten gran parte de su presupuesto en mejorarlo o adaptarlo a las necesidades de cada uno de sus clientes individuales, por lo que finalmente deben dar soporte a un conjunto de variantes de su producto [Muthig 02].

Por lo general, las empresas productoras de software se especializan en un determinado rango o dominio de aplicaciones por lo que sus productos suelen tener muchas características en común. A este conjunto de aplicaciones similares pertenecientes a un mismo dominio de aplicación se le suele denominar *familia de productos*. A veces, erróneamente, se asimila este concepto al de *línea de productos*, aunque como queda recogido en las siguientes definiciones existen ciertos matices que permiten diferenciar ambos conceptos.

> *"Línea de productos: conjunto de productos que comparten una serie de características comunes y que responden a las necesidades particulares de un determinado sector del mercado" [Whitney 96].*

Definición 8. Línea de Productos.

Por ejemplo, una línea de productos de higiene puede incluir, entre otros, geles de ducha, espumas de afeitar, productos de maquillaje o perfumes, etc.

> *"Familia de productos: conjunto de productos pertenecientes a un mismo dominio de aplicación que pueden construirse a partir de un conjunto de elementos básicos comunes." [Whitney 96].*

Definición 9. Familia de Productos.

Siguiendo con el ejemplo anterior, podría crearse una familia de geles de ducha que incluyera distintas variedades, por ejemplo, para pieles sensibles, para mujeres, hombres, bebés, etc.

No obstante, una línea de productos puede diseñarse para desarrollar productos de una misma familia. Esto proporciona la ventaja de poder construir los distintos productos de la línea utilizando un conjunto de recursos comunes y válidos para todos ellos [Eisenecker 03]. Así, siguiendo con los ejemplos anteriores, tendría sentido hablar de una línea de productos de la familia "gel de ducha", de forma que todos los geles podrían fabricarse a partir de la misma base jabonosa o utilizando los mismos envases, diferenciándose unos de otros en el perfume añadido al jabón o en el color del etiquetado.

En el caso de las líneas de productos software (LPS) éstas siempre se utilizan para desarrollar sistemas de una misma familia o dominio de aplicación, tal y como queda recogido en la siguiente definición, en la que se ha resaltado en negrita las peculiaridades de éstas frente a las utilizadas para desarrollar otros tipos de productos (ver Definición 10).

> *"Una línea de productos software es un conjunto de sistemas* ***intensivamente software*** *que comparten una serie de características comunes y que satisfacen las necesidades de un segmento particular del mercado,* ***pudiendo desarrollarse a partir de un conjunto de recursos comunes de acuerdo con un plan de producción previamente establecido****." [Clements 02].*

Definición 10. Línea de Productos Software (LPS).

En este caso puede sugerirse, a modo de ejemplo, una LPS para el desarrollo de herramientas CAD (*Computer Aided Design*) en la que se podrían utilizar como recursos comunes y válidos para todas ellas, ciertas primitivas básicas de dibujo, visualización bidimensional (2D) y tridimensional (3D), coloreado y texturizado, etc. Otros recursos tales como las bases de datos de materiales a emplear, o ciertas primitivas de dibujo avanzadas, serán específicas de cada tipo de CAD. Así, por ejemplo, mientras que un CAD de diseño electrónico deberá incluir primitivas para dibujar los distintos tipos de chips, otro para arquitectura se dotará con primitivas para representar vigas, paredes o ventanas.

Describir aquello que queda fuera de la definición de un concepto resulta a menudo tanto o más útil que la propia definición en sí. En este sentido, las líneas de productos software no deben confundirse con: (1) aplicaciones que emplean, de manera casual y esporádica, mecanismos de reutilización de grano fino, (2) sistemas construidos reutilizando porciones de código extraídas a conveniencia de otros sistemas, (3) aplicaciones construidas a partir de una arquitectura genérica o utilizando una metodología de desarrollo basada en componentes y (4) un conjunto de versiones desarrolladas a partir de un mismo producto [Clements 02]. Así, si bien el desarrollo de software basado en líneas de productos puede considerarse una suma, entre otros, de varios de los conceptos aquí mencionados (reutilización, arquitectura software, componentes, etc.), éstos no se utilizan de manera casual, opcional, parcial o interesada sino que, muy al contrario, todos ellos resultan esenciales y de su correcta integración, siempre previamente planificada, dependerá el éxito final de la LPS.

El desarrollo de software basado en líneas de productos requiere llevar a cabo dos actividades, la primera de ellas relacionada con lo que se ha dado en llamar *ingeniería del dominio* y la segunda relacionada con la denominada *ingeniería de las aplicaciones* (ver Figura 21).

La *ingeniería de dominio* está inspirada en el concepto de "implementar *para* reutilizar". Esta disciplina se encarga de analizar las características comunes (*commonalities*) y diferenciales (*variabilities*) de las aplicaciones para, a partir de ellas, construir la infraestructura de la LPS consistente en un repositorio de artefactos reutilizables. Entre éstos cabe destacar la especificación de los requisitos de las aplicaciones, su arquitectura software y el conjunto de componentes a partir de los que se construirán los distintos productos de la línea. El manejo de la variabilidad propia de las LPS requiere de tecnologías que den soporte a la identificación, creación y adaptación de este núcleo de artefactos reutilizables.

A diferencia de la anterior, la *ingeniería de aplicaciones* se inspira en el concepto de "implementación con reutilización" y es la responsable de la construcción de los distintos productos a partir de los artefactos desarrollados por la primera. Cada uno de estos productos se construirá (1) seleccionando aquellos recursos que resulten de utilidad, adaptándolos como convenga mediante los mecanismos de variación previamente establecidos (parametrización, especialización mediante herencia, etc.), (2) desarrollando nuevos recursos cuando sea necesario, y (3) ensamblándolos de forma que encajen en la arquitectura del sistema siguiendo las reglas establecidas en el plan de producción. De esta manera, construir un nuevo producto o sistema se convierte más en una tarea de ensamblaje o integración que de programación.

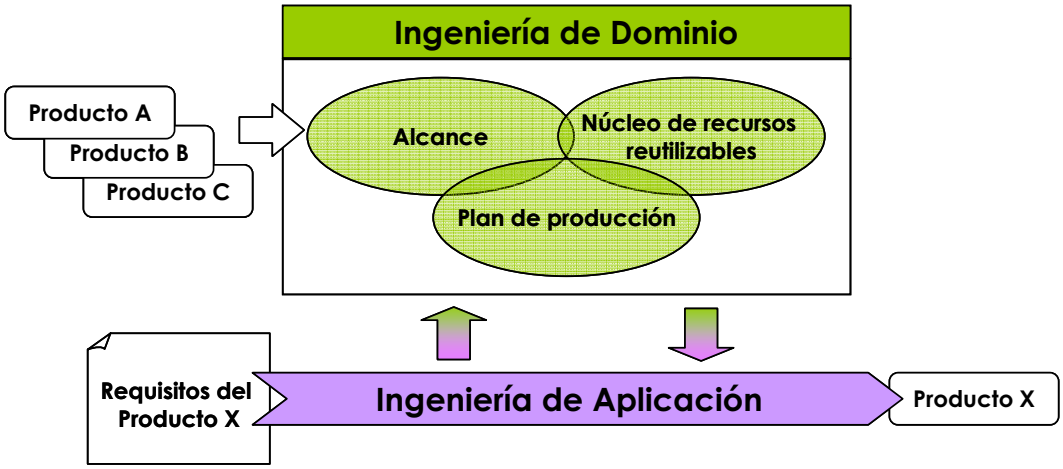


Figura 21. Esquema de los procesos de ingeniería de dominio y de aplicaciones.

Con relación a la ingeniería de dominio es necesario hacer hincapié en las tres etapas que ésta comprende y que se comentan a continuación: (1) la definición del alcance de la LPS, (2) el desarrollo[17] del núcleo de recursos reutilizables, y (3) la especificación del plan de producción.

3.3.2.1. Definición del alcance

El alcance de una LPS define qué aplicaciones pueden derivarse a partir de ella, especificando qué características tienen en común y cuáles las diferencian. De este modo, para cada aplicación incluida en el alcance de la LPS, se deberá especificar qué características (funcionalidades) debe ésta incorporar de manera obligatoria, opcional o alternativa (característica obligatoria seleccionada de entre un conjunto de opciones válidas).

Por lo general el alcance de una LPS suele presentarse utilizando una matriz bidimensional de productos vs. características a la que suele adjuntarse un diagrama de variabiliad en el que quedan recogidas todas las características identificadas así como las relaciones existentes entre ellas [Savolainen 01]. Existen diversas notaciones gráficas para representar estos diagramas de variabilidad pudiendo encontrarse una recopilación de algunas de ellas en [Trigaux 03].

La correcta definición del alcance y, en particular, el modelado sistemático de la variabilidad resultan actividades esenciales para el éxito de cualquier LPS ya que de ellas depende su capacidad para anticipar y poder reaccionar, de manera adecuada, ante los posibles cambios que pudieran introducirse en el futuro [van Gurp 01].

3.3.2.2. Desarrollo del núcleo de recursos reutilizables (core assets)

El núcleo de recursos esenciales es la base de toda LPS. Estos recursos incluyen, entre otros, la arquitectura, los componentes software y hardware, los modelos de dominio, los requisitos, la documentación y las especificaciones que se van

[17] Desarrollar estos recursos puede entenderse como implementarlos (ya sea desde cero o a partir de versiones previas), adquirirlos a terceros como productos COTS (con o sin adaptaciones posteriores), o encargar su fabricación a otra empresa.

generando, los modelos de rendimiento, la planificación temporal de las actividades, el presupuesto, los casos de prueba, etc. Todos estos recursos ofrecen distintos niveles de reutilización durante el desarrollo del producto y quizá, de entre ellos, los más importantes sean la arquitectura y los componentes software. Puesto que a éstos últimos ya se ha dedicado uno de los apartados anteriores, parece conveniente realizar en este punto algunas consideraciones sobre las arquitecturas software.

➢ Arquitectura Software (AS)

En la literatura es posible encontrar numerosas definiciones para el término Arquitectura Software (AS) entre las que, a continuación, se muestran las propuestas por Garlan y Bass:

> *"La arquitectura software de un sistema establece la estructura organizativa de sus componentes incluyendo cómo éstos se relacionan entre sí, así como los principios y líneas maestras que gobiernan su diseño y evolución" [Garlan 95].*
>
> *"La arquitectura de un sistema consiste en una vista abstracta que elimina los detalles de implementación y se concentra en el comportamiento e interconexión de los componentes vistos como cajas negras." [Bass 98].*

Definición 11. Arquitectura Software (AS) según Garlan y Bass.

En las definiciones anteriores, el término componente no debe entenderse como la "*unidad binaria de composición*" a la que se hizo referencia al hablar del desarrollo de software basado en componentes. En el caso de las AS, un componente es una entidad abstracta que sirve para modelar el sistema visto como un todo, los subsistemas que lo componen, los bloques funcionales en los que éstos pueden dividirse, etc. Así, la especificación de la AS posee distintos niveles de abstracción tal y como se muestra en la Figura 22.

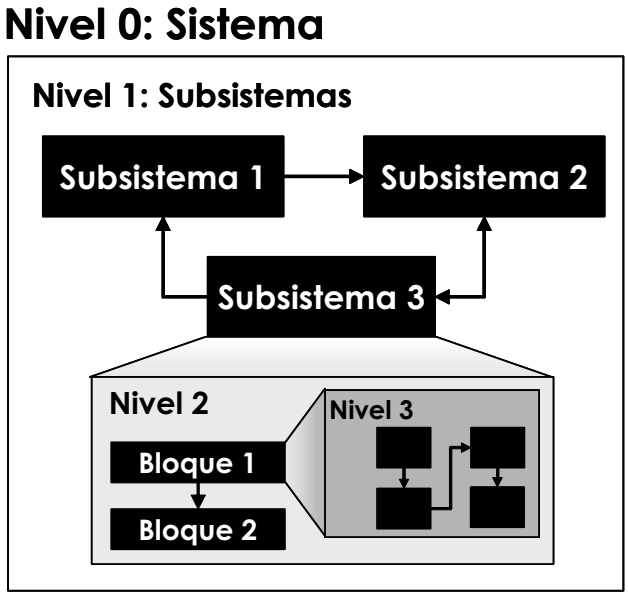


Figura 22. Distintos niveles de abstracción de una AS.

A la hora de diseñar una AS es posible acudir a ciertos *patrones o estilos arquitectónicos* previamente establecidos [Gamma 95]. Por ejemplo, el patrón denominado "*pipes and filters*" modela las aplicaciones utilizando únicamente dos tipos de elementos que se encargan, unos de procesar la información (*filters*) y otros de transmitirla (*pipes*). El patrón establece además que sólo es posible conectar entre sí elementos de distinto tipo. En particular, existen algunos patrones de diseño específicos para definir AS destinadas a la construcción de productos de un determinado dominio, también denominadas *arquitecturas específicas de dominio* o *arquitecturas de referencia*. A esta categoría pertenecen las AS de las LPS.

A la hora de especificar la AS de una LPS se deberá recoger la variabilidad inherente a los distintos productos que se pretenden diseñar, esto es, la AS deberá capturar las similitudes que subyacen en la estructura y la lógica de todos ellos y, simultáneamente, ser lo suficientemente flexible como para incluir sus peculiaridades. Además, deberá evolucionar de manera sencilla a partir de los requisitos y ser independiente de la plataforma sobre la que se desarrollen las implementaciones particulares.

3.3.2.3. Plan de producción de una LPS

El plan de producción describe cómo los distintos productos se construyen a partir del núcleo de recursos definido en la fase anterior. Para ello, cada uno de estos recursos deberá conocer cuál es su aportación al proceso de desarrollo (planes de producción parciales) y, adicionalmente, deberá definirse un plan de producción global en el que se establezca qué recursos de entre todos los disponibles son necesarios y cómo encajarlos para conseguir el producto final deseado.

Cada uno de los elementos que forman parte del plan de producción de una LPS, esto es, el conjunto de planes parciales y el plan global, se presentarán en un formato adecuado al tipo de usuario al que vayan destinados (analistas, diseñadores, integradores, clientes, etc.). Así, los planes de producción pueden variar desde especificaciones técnicas detalladas sobre el modelo de interoperabilidad de los componentes software disponibles, hasta documentos más informales que describan cómo utilizar los casos de prueba para validar cada producto.

3.3.3. Programación generativa

La Programación Generativa (PG) es un paradigma de desarrollo de software de reciente aparición que, como en el caso de las LPS, trata de desarrollar productos software pertenecientes a una misma familia. Para ello, este enfoque persigue que a partir de la especificación de unos ciertos requisitos, contemplados como opciones de configuración, y utilizando una serie de componentes de implementación elementales y reutilizables, se pueda derivar automáticamente un producto optimizado, ya sea éste intermedio o final [Czarnecki 00].

Según esta definición, el uso de técnicas de PG consigue, una vez que el programador de aplicaciones ha establecido en términos abstractos (utilizando un lenguaje textual o gráfico) las características del producto que desea, que sea un generador el que se encargue de construirlo automáticamente, enlazando

adecuadamente los componentes necesarios. Para conseguir este objetivo tanto tiempo anhelado por la Ingeniería del Software es necesario (1) diseñar los componentes de implementación para que encajen en la arquitectura, (2) modelar el *proceso de generación* o traducción de los requisitos abstractos (*espacio del problema*) a un conjunto específico de componentes (*espacio de la solución*), y (3) conseguir que esta traducción la implemente automáticamente un generador.

Tal y como se muestra en la Figura 23, pasar del espacio del problema al de la solución requiere dotar al generador de cierta información relativa a los recursos con los que cuenta y a las reglas de composición que le permitirán combinarlos de manera que consiga, de entre los resultados posibles, el mejor. Esta información deberá estar especificada de la manera más precisa y formal que sea posible, de modo que el generador pueda procesarla adecuadamente [Czarnecki 00]. La ventaja de utilizar especificaciones formales es que a partir de ellas el generador puede, aplicando una serie de reglas sencillas, llegar a una solución optimizada, mientras que esta misma tarea resultaría enormemente compleja y costosa para un programador. Como contrapartida debe considerarse el esfuerzo que supone formalizar la especificación del problema, lo que generalmente conlleva el uso de lenguajes formales como Maude [MAUDE] [Lucas 04] derivado de OBJ u OOZE [Alencar 91] [Nicolás 96] basado en la notación algebraica Z.

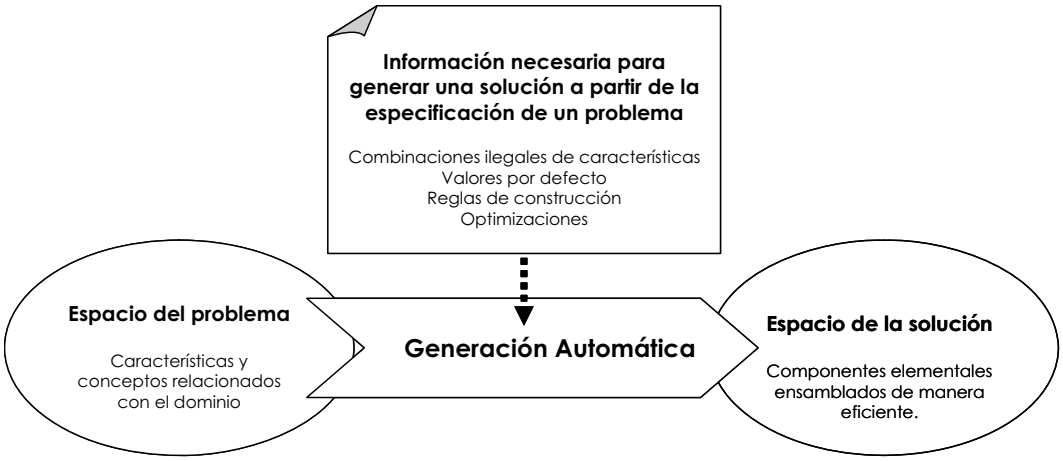


Figura 23. Elementos de la PG.

Desde cierto punto de vista, las herramientas de PG pueden considerarse compiladores de modelos o herramientas de meta-programación. De manera similar a como en lingüística se utiliza un meta-lenguaje para discutir, describir o analizar otro lenguaje, un meta-programa puede definirse como aquel que se utiliza para generar otros programas. Desde este punto de vista, la PG puede considerarse una técnica de meta-programación [Štuikys 02].

La generación automática de software no es un concepto nuevo ni exclusivo de la PG. Por ejemplo, los compiladores generan automáticamente código ejecutable a partir de código fuente, así como algunas herramientas CASE (*Computer Aided Software Engineering*) son capaces de generar automáticamente esqueletos de aplicaciones a partir de modelos UML.

Del mismo modo, otros de los conceptos que aparecen íntimamente ligados a la PG tampoco son nuevos, como por ejemplo el desarrollo de software basado en

objetos, componentes y aspectos, la ingeniería de dominio, la programación intencional (IP, *Intentional Programming*), etc. La principal ventaja de la PG reside en que este paradigma de desarrollo de software, por primera vez, lejos de intentar diferenciarse de sus predecesores busca incorporar todo lo que cada uno de ellos tiene de positivo. De entre todos ellos, quizá los que más aportan a la PG sean el paradigma de desarrollo de software basado en LPS y el de programación automática [Balzer 85]; de hecho puede considerarse a la PG como una herramienta que permite, a partir de la especificación de una LPS, derivar distintas aplicaciones pertenecientes a un mismo dominio (el recogido en la LPS) de manera (semi-)automática.

Lo reciente de la PG determina que este paradigma aún no haya alcanzado un grado madurez comparable al de sus predecesores. Por otra parte, es preciso tener en cuenta que, como en el caso de las LPS, su aplicación está limitada al desarrollo de familias de productos, lo que restringe el número de usuarios potenciales y, por lo tanto, su nivel de implantación. Hasta donde sabemos, actualmente sólo existen disponibles un número muy reducido de herramientas entre las que cabe destacar AHEAD (*Algebraic Hierarchical Equations for Application Design*) [AHEAD], desarrollada por Don Batory.

3.3.4. Programación visual

La Programación Visual (PV) se basa en el desarrollo de aplicaciones utilizando lenguajes o notaciones multi-dimensionales [Burnett 99]. La semántica de un Lenguaje de Programación Visual (LPV) queda definida por un conjunto de expresiones visuales, siendo cada una de ellas un vector multidimensional que refleja distintos aspectos o dimensiones del lenguaje. Entre otras, los LPV suelen incluir varias de las siguientes dimensiones: textual (en el sentido de los lenguajes de programación clásicos), espacial (añadiendo información gráfica bi-, tri-, o multi-dimensional), temporal (añadiendo relaciones "antes de", "después de", o "a la vez que"), etc.

La PV persigue, entre otros, los siguientes objetivos [Green 95]:

- Facilitar la tarea de programar aplicaciones proporcionando mecanismos para (1) reducir el número de conceptos que es necesario manejar para programar, (2) representar las relaciones entre objetos de forma explícita y por lo tanto más sencilla de entender, y (3) permitir visualizar de manera inmediata las modificaciones realizadas sobre el programa.
- Mejorar la corrección de las aplicaciones que se desarrollan, ya que es el entorno de PV el que se encarga de traducir a código máquina los modelos visuales diseñados por el usuario.
- Reducir el tiempo empleado en el desarrollo de aplicaciones.

Obviamente, los lenguajes de programación tradicionales tales como C/C++, Pascal, Java, etc., no pueden considerarse LPV ya que se utilizan para crear un flujo unidimensional de datos (programa) que será utilizado como entrada por un compilador para generar el código ejecutable o interpretable correspondiente. A pesar de su nombre, algunas herramientas como Visual C++ o Visual Basic tampoco pueden considerarse visuales ya que no dejan de ser herramientas de programación

textual a las que se ha dotado de un entorno gráfico de desarrollo; así, estas herramientas pueden considerarse "gráficas" pero en ningún caso "visuales".

Los LPV pueden clasificarse atendiendo a distintos criterios [del Fabbro 00], por ejemplo, en función del paradigma de programación en el que estén inspirados pueden dividirse en orientados a objetos, basados en reglas, imperativos, funcionales, etc. Del mismo modo, atendiendo al tipo de sintaxis visual que utilizan, éstos pueden agruparse en lenguajes diagramáticos (las funciones se representan como objetos gráficos que se conectan entre sí a través de enlaces que representan un intercambio de datos), icónicos (utilizan figuras diferentes para representar distintos conceptos) o basados en secuencias pictóricas que describen una sucesión de acontecimientos relacionados. Por último, los LPV también pueden clasificarse atendiendo al dominio de aplicación en el que suelen emplearse; así, por ejemplo, existen LPV para desarrollar aplicaciones de control [LABVIEW] [SIMULINK], sistemas de procesamiento de imágenes [VISIQUEST], etc.

A pesar de que los LPV resultan más sencillos e intuitivos de aprender y manejar que los lenguajes textuales, en la actualidad su grado de implantación es mucho más reducido que el de éstos lo que puede atribuirse, entre otros, a los siguientes factores:

- Aunque más sencillos de programar que los textuales, los LPV pueden no proporcionar una semántica lo suficientemente rica, haciendo que en este caso resulte difícil, cuando no imposible, programar la solución a los problemas que se plantean.
- Como algunos programadores aducen, los LPV no producen resultados tan eficientes como los que pueden alcanzarse programando "manualmente" y "a bajo nivel" mediante un lenguaje textual. En consecuencia, los LPV suelen utilizarse únicamente para desarrollar prototipos rápidos que, una vez validados, suelen volver a implementarse utilizando un lenguaje textual que permita mejorar su eficiencia.
- Algunos autores consideran que los LPV, a pesar de ser cada vez más potentes, nunca conseguirán desbancar a los lenguajes textuales clásicos como C/C++ o Cobol [Green 95]. Sin embargo, la PV empieza a consolidarse como uno de los paradigmas de programación de mayor implantación en algunos sectores como los relacionados con los sistemas de control o los de procesamiento de imágenes.

3.3.5. Aportaciones de estos paradigmas a la construcción de VIPS

A continuación se comentan brevemente las ventajas e inconvenientes asociados al uso de los paradigmas antes analizados para, posteriormente, demostrar cómo su integración bajo un marco de desarrollo unificado permitirá potenciar sus cualidades, minimizando simultáneamente sus limitaciones.

En primer lugar, como ya se ha comentado anteriormente, el uso de Líneas de Productos Software (LPS) conlleva una serie de ventajas evidentes. Entre ellas cabe destacar las que se derivan del modelado conjunto de varios sistemas pertenecientes a una misma familia o dominio de aplicación, así como las que resultan de la posibilidad de construirlos a partir de una serie de recursos comunes. La aplicación de

este paradigma de desarrollo de software permite reducir considerablemente el tiempo de desarrollo de los distintos productos, así como su coste, todo ello gracias al fomento de atributos como la flexibilidad o la reutilización. Sin embargo, la ausencia de herramientas que soporten de una manera integral todo el ciclo de vida de las LPS conlleva que la conexión entre los modelos de diseño abstractos y la implementación final de los productos concretos resulte extremadamente compleja y que tenga que realizarse manualmente en la mayoría de los casos.

En el caso particular de los VIPS, el uso de LPS permitiría resolver los problemas asociados a la falta de flexibilidad de los diseños actuales (muy dependientes del hardware) gracias a la definición de la arquitectura software genérica de estos productos. Del mismo modo, el desarrollo de VIPS basado en LPS permitiría conseguir una mejor reutilización tanto del conocimiento adquirido en desarrollos anteriores como de los recursos previamente generados.

Por su parte, el Desarrollo de Software Basado en Componentes (DSBC) trata también de explotar al máximo el concepto de reutilización aunque, en este caso, desde un enfoque mucho más próximo a la implementación. En la práctica, el DSBC suele centrarse en resolver problemas como la selección y la validación de los componentes software, la especificación formal de su semántica, o su compatibilidad e interconexión. Sin embargo, el DSBC no ha sido concebido con la idea de desarrollar familias de sistemas limitándose, por lo general, al desarrollo de aplicaciones específicas a partir de componentes concretos.

Volviendo al caso particular de los VIPS, conviene recordar que estos sistemas suelen implementarse utilizando distintas herramientas (algunas de ellas de tipo COTS) que por lo general suelen ser incompatibles entre sí. La integración de estas herramientas utilizando los mecanismos de adaptación e interconexión promovidos por el DSBC, puede considerarse la aportación más importante de este paradigma al proceso de construcción de los VIPS.

La Programación Generativa (PG), por su parte, trata de facilitar el desarrollo de sistemas pertenecientes a un mismo dominio de aplicación, automatizando la transición entre los diseños abstractos y la implementación de cada uno de los productos concretos. Con ello se consigue corregir la principal limitación de las LPS. Adicionalmente, el hecho de utilizar modelos formales de reglas para (1) construir un conjunto de componentes reutilizables y fácilmente conectables entre sí y (2) derivar automáticamente la solución a partir de ellos, permite mejorar la corrección de las aplicaciones a la vez que facilita enormemente las tareas de diseño e implementación. La aportación de la PG al desarrollo de VIPS es pues evidente en cuanto que resuelve muchas de las limitaciones de los paradigmas LPS y DSBC, a la vez que facilita su integración.

Por último, la Programación Visual (PV) permite simplificar la especificación de los sistemas haciéndola más sencilla e intuitiva para el programador. En particular, la PV resulta especialmente adecuada para el DSBC, ya que el modelado jerárquico de un sistema, visto como un componente de componentes, resulta más sencillo de representar como un conjunto de bloques gráficos enlazados mediante líneas, que utilizando un lenguaje textual clásico. Este hecho puede considerarse una aportación importante de la PV al DSBC y, en particular, al proceso de diseño e implementación de los VIPS.

3.4. Una nueva metodología multi-paradigma para el desarrollo de VIPS

La nueva metodología que se propone en esta Tesis trata de resolver los problemas asociados al procedimiento actual de construcción de los VIPS (ver apartado 3.2) mediante la integración de varios de los paradigmas de desarrollo de software más actuales (ver apartado 3.3).

Frente al procedimiento actual, muy centrado en el hardware, la metodología que aquí se propone se centra preferentemente en el software lo que permitirá mejorar aspectos como la flexibilidad o la reutilización sin olvidar otros como la eficiencia.

Como se tratará de demostrar, el hecho de integrar varios paradigmas de desarrollo de software en esta propuesta consigue un efecto sinérgico ya que, además de aprovechar las ventajas que ofrece cada uno de ellos, su complementariedad permite reducir considerablemente las limitaciones que presentan cuando se aplican de manera individual.

La nueva metodología que se propone se basa en la definición de una línea (de familias) de productos denominada LPS-VIPS que resulta de aplicación para el desarrollo de productos en el domino de los Sistemas de Procesamiento de Información Visual. Como parte del núcleo de recursos reutilizables de la línea LPS-VIPS, además de los artefactos comúnmente incluidos (modelos requisitos del dominio, modelos de análisis, arquitectura software, etc.), se incorporarán dos elementos adicionales (ver Figura 24):

- Una librería de componentes software para procesamiento de imágenes que integrará y compatibilizará la funcionalidad ofrecida por varias de las librerías existentes en la actualidad (en su mayoría productos COTS).
- La herramienta IP-CoDER que permitirá: (1) crear y añadir nuevos componentes a la librería anterior, (2) combinarlos y componerlos gráficamente mediante un lenguaje de programación visual, y (3) generar, a partir de dichos componentes y de manera totalmente automática, prototipos ejecutables de aplicaciones de procesamiento de imágenes.

Como se observa en la Figura 24, el nuevo proceso que se propone para el desarrollo de VIPS parte de la línea LPS-VIPS que deberá haber sido previamente definida (ver Capítulo 4). A partir del núcleo de recursos reutilizables que ésta proporciona y de los requisitos del producto que se desea construir (familia a la que pertenece el VIPS, funcionalidad que debe proporcionar, requisitos no funcionales relacionados con la velocidad de respuesta, la precisión, el coste, etc.) se procederá a crear una instancia de la línea LPS-VIPS, configurando las opciones de variabilidad ofrecidas por el modelo de análisis y la arquitectura software de la línea de productos.

Como resultado de este proceso se obtendrá un "esqueleto" de aplicación con parte de la funcionalidad ya implementada (por ejemplo la relativa a la interfaz de usuario, la comunicación entre los distintos subsistemas del VIPS, etc.), pero con algunas de sus características aún por especificar. Este es el caso de la funcionalidad asociada al procesamiento de imágenes cuyo enorme grado de variabilidad (dependiendo de la aplicación se utilizan distintas secuencias de algoritmos de procesamiento de imágenes) hace que resulte imposible modelarla como parte del

núcleo de recursos reutilizables de la línea LPS-VIPS.

Para completar la parte del esqueleto correspondiente al procesamiento de imágenes del VIPS se utilizará la herramienta IP-CoDER con la que, de manera sencilla, se podrán crear distintas configuraciones a partir de los componentes de procesamiento de imágenes previamente creados y almacenados en la librería (ver Figura 24). Una vez seleccionada una configuración, la herramienta IP-CoDER generará automáticamente un prototipo ejecutable que permitirá al usuario validar su diseño utilizando distintos datos de entrada. Si el prototipo produce los resultados deseados, su código se incorporará al esqueleto de la aplicación completándose así la funcionalidad correspondiente al procesamiento de imágenes; en caso contrario, se deberá modificar y volver a validar el diseño hasta encontrar una configuración válida de componentes.

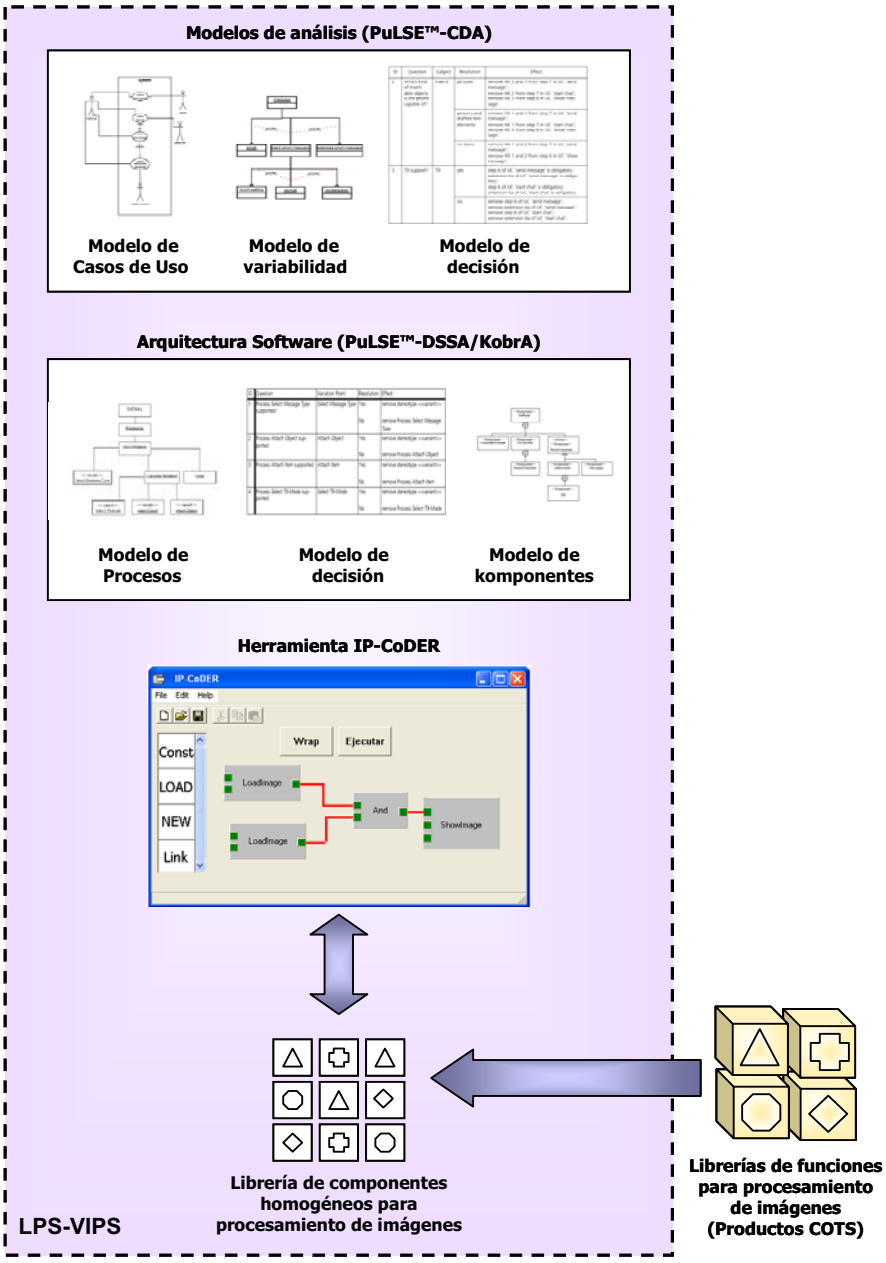


Figura 24. Núcleo de recursos reutilizables de la línea LPS-VIPS.

Los siguientes capítulos recogen la definición de la línea LPS-VIPS (Capítulo 4) y los detalles de cómo se han implementado tanto la librería de componentes software para procesamiento de imágenes como la herramienta de soporte IP-CoDER (Capítulo 5). Para la definición de la línea LPS-VIPS se han empleado las metodologías PuLSE™ y Kobra, ambas concebidas para la especificación de líneas de productos software y documentadas en el Apéndice I al final de esta memoria.

Esta página se deja en blanco intencionadamente

Capítulo 4

Definición de la Línea LPS-VIPS

En este capítulo se define la línea de productos LPS-VIPS a partir de la que se podrán derivar distintos VIPS de las familias descritas en el Capítulo 2. Para ello será necesario realizar un análisis exhaustivo y sistemático de las características que tienen en común todos estos sistemas así como de los aspectos en los que se diferencian unos de otros (análisis de la variabilidad). Este estudio permitirá desarrollar un núcleo de recursos reutilizables a partir de los que se podrán construir los distintos VIPS específicos que se desee siguiendo la metodología propuesta en el capítulo anterior.

4.1. Definición de LPS: Trabajos Previos

El desarrollo de software basado en líneas de productos puede considerarse una de las tendencias más actuales y en boga dentro del campo de la Ingeniería del Software. Sin embargo, son pocas las propuestas metodológicas que abordan la definición de Líneas de Productos Software (LPS) de manera completa, rigurosa y pragmática. Probablemente esto es debido, por una parte, a lo relativamente reciente de este enfoque, y por la otra, al hecho de que éste exhibe sus mejores resultados sólo cuando se trata de desarrollar varios productos de una cierta envergadura y pertenecientes a un mismo dominio de aplicación.

Entre los trabajos descritos en la literatura relacionados con la descripción de LPS, cabe destacar las propuestas realizadas por el SEI [Bass 00] [Clements 02] [Chastek 02], las del Fraunhofer Institute of Experimental Software Engineering (IESE) [Atkinson 01] [Kettemann 03] [PULSE] [KOBRA], la de H. Gomma [Gomma 04], o la de K. C. Kang [Kang 02]. De entre ellas, las dos primeras son las que están más documentadas, tanto

de manera teórica como con casos prácticos, siendo las más referenciadas por el resto de trabajos publicados en la actualidad en relación con el desarrollo de LPS.

4.1.1. Metodologías seleccionadas para definir la línea LPS-VIPS

Para describir la línea LPS-VIPS de manera rigurosa, se ha optado por utilizar una combinación de las metodologías PuLSE™ [PULSE] y KobrA [KOBRA] [Atkinson 01], ambas desarrolladas por el Fraunhofer Institute of Experimental Software Engineering (IESE) y cuyo uso está relativamente extendido en la actualidad. Dado que este capítulo resultaría excesivamente extenso de incluirse en él una descripción detallada de estas metodologías, se ha optado por recoger los aspectos más relevantes de cada una de ellas en el Apéndice I para facilitar la mejor comprensión del contenido de este capítulo.

Algunos ejemplos del uso combinado de estas metodologías pueden encontrarse en los casos de estudio desarrollados por J. Bayer –definición de una LPS para sistemas de gestión bibliotecaria [Bayer 01]– y D. Muthig –definición de una LPS para sistemas de telefonía móvil [Muthig 04]. Tanto estos casos de estudio como el resto de los documentados en la literatura (ya sea utilizando éstas u otras metodologías) describen líneas de productos software pertenecientes a un determinado domino de aplicación. Sin embargo, en este trabajo lo que se pretende especificar no es una línea de productos sino una línea de familias de productos. Esto supone incorporar un grado más de abstracción al problema y dado que, hasta donde sabemos, ninguna metodología contempla este supuesto, ha sido necesario introducir ciertas adaptaciones en algunos de los procesos y herramientas utilizados por las metodologías PuLSE™ y KobrA. Así, dado el gran número y variedad de aplicaciones (agrupadas en familias) que se pretende poder derivar a partir de de la línea LPS-VIPS, en ocasiones resultará imposible realizar ciertas especificaciones con el mismo nivel de detalle que el presentado en los casos de estudio antes mencionados.

El mayor grado de abstracción requerido en este caso a la hora de especificar la línea LPS-VIPS conllevará una mayor abstracción también en los resultados que se obtengan al aplicar las metodologías PuLSE™ y KobrA. Así, será necesario realizar un cierto esfuerzo adicional durante el proceso de instanciación de la línea LPS-VIPS para configurar ciertos detalles de la aplicación concreta que se quiera desarrollar. Este esfuerzo se verá considerablemente paliado gracias al soporte que proporciona la herramienta IP-CoDER al proceso de instanciación/configuración de aplicaciones, tal y como queda recogido en el siguiente capítulo.

4.1.2. Esquema del proceso de definición de la línea LPS-VIPS

En este apartado se presenta el esquema del proceso que se ha seguido para definir la línea LPS-VIPS y que se desarrollará a lo largo de los distintos apartados de este capítulo. En cada etapa de este esquema se indica qué fase de la metodología PuLSE™ y/o KobrA se ha empleado así como las herramientas utilizadas en cada caso (tablas, plantillas, diagramas, etc.). Para una mejor comprensión de este esquema y si no se está familiarizado con las metodologías PuLSE™ y KobrA conviene leer previamente la descripción que de ellas se hace en el Apéndice I.

1. **Definición del alcance de la línea LPS-VIPS (PuLSE™-Eco)**

   1.1. Descripción de los productos que se pretende poder derivar de la línea LPS-VIPS. En este apartado se describirán las seis familias de VIPS presentadas en el capítulo 2 utilizando para ello una plantilla que permitirá recoger sus características principales.

   1.2. Identificación y descripción de los dominios más relevantes:

   1.2.1. Definición de la tabla de características: a partir de las descripciones previas se extraerán las principales características (funcionalidades) proporcionadas por los distintos tipos de VIPS agrupándolas por dominios (áreas funcionales).

   1.2.2. Descripción de los dominios identificados: cada dominio identificado en la fase previa se describirá utilizando una plantilla diseñada para recoger sus características principales.

   1.2.3. Definición del mapa de relaciones entre dominios: utilizando la información recogida en las descripciones anteriores se elaborará un diagrama que reflejará las relaciones de dependencia y jerarquía existentes entre los distintos dominios.

   1.2.4. Definición del mapa inicial de productos en el que se recogerá la funcionalidad externa e interna de las distintas familias de VIPS agrupándola en los dominios previamente definidos. Esta tabla será una versión extendida y reordenada de la tabla de características definida en 1.2.1.

   1.3. Valoración y selección de dominios:

   1.3.1. Estudio de la relevancia y aportación funcional de los dominios: se elaborará una tabla que recogerá las valoraciones realizadas por varios expertos en relación a la relevancia y la aportación funcional de cada dominio a la hora de construir los distintos tipos de VIPS.

   1.3.2. Estudio del potencial de reutilización de los dominios: como en el caso anterior se elaborará una tabla con las valoraciones realizadas por expertos en relación al potencial de reutilización de cada uno de los dominios.

   1.3.3. Selección de dominios y definición final del alcance: se evaluará el coste/beneficio de incorporar la funcionalidad ofrecida por cada uno de los dominios al núcleo de recursos reutilizables de la línea LPS-VIPS. Como resultado de esta evaluación se determinará qué dominios quedan dentro del alcance de la línea de productos y cuáles, por el contrario, quedan excluidos.

2. **Análisis del dominio (PuLSE™-CDA)**

   2.1. Modelado de Casos de Uso (CU): Se identificarán y describirán los CU asociados a cada uno de los dominios previamente seleccionados como resultado de la fase PuLSE™-Eco. Los distintos CU se especificarán de forma gráfica (utilizando una extensión de la notación UML para permitir el modelado de elementos variantes) y textual (utilizando una plantilla de descripción de CU).

2.2. Modelado de la variabilidad: para cada uno de los CU descritos en la fase anterior se elaborará un diagrama de variabilidad utilizando la notación FODA (Feature-Oriented Domain Analisys) [Kang 90].

2.3. Modelo de decisión: a partir de las descripciones de los CU y de sus modelos de variabilidad asociados, se elaborará una tabla de decisión que recogerá las distintas alternativas posibles para cada uno de los elementos modelados como variantes.

3. **Definición de la arquitectura genérica de los VIPS (PuLSE™-DSSA/KobrA).** Partiendo de los resultado obtenidos previamente durante la fase de análisis del dominio (PuLSE™-CDA) se establecerá el contexto en el que típicamente operan los VIPS. Para ello, se *especificará* un primer Komponente[18] que recogerá las características externamente visibles de un VIPS genérico. A continuación, se *definirá* este Komponente identificando sus Komponentes internos y estableciendo cómo éstos realizan la funcionalidad previamente descrita en la *especificación*. Una vez completada la especificación y la definición del contexto, este Komponente inicial pasará a ser la raíz de un árbol cuyos descendientes (Komponentes internos) deberán a su vez ser especificados y definidos siguiendo un proceso recursivo que concluirá cuando se alcance el máximo grado de descomposición posible. Cuando alguno de los Komponentes identificados presente algún tipo de variabilidad externa y/o interna deberá acompañarse su especificación y/o definición con la correspondiente tabla de decisión en la que se recogerán las distintas alternativas de configuración posibles. Finalmente, la arquitectura genérica de la línea LPS-VIPS vendrá dada por el árbol de Komponentes resultante de todo este proceso.

En los siguientes apartados se detallan los procesos realizados y los resultados obtenidos en cada una de las etapas señaladas en el esquema anterior.

4.2. Definición del alcance de la LPS-VIPS (PuLSE™-Eco)

La definición del alcance de una LPS establece qué aplicaciones pueden derivarse a partir de ella, qué características tienen éstas en común y cuáles las diferencian. Como ya se comentó en el capítulo anterior, la correcta definición del alcance y, en particular, el modelado sistemático de la variabilidad, resultan actividades esenciales para el éxito de toda LPS.

La definición del alcance de la línea LPS-VIPS se realizará siguiendo las indicaciones marcadas por la metodología PuLSE™ en su fase Eco (*Economic Scoping*). Como ya se ha comentado en el apartado anterior, esta fase comprende cuatro etapas: (1) descripción de los productos (en este caso familias de productos) que se desea poder derivar de la línea, (2) identificación y descripción de los dominios (áreas funcionales) identificados como más relevantes en uno de los tipos de VIPS, (3) descripción de las relaciones existentes entre estos dominios, y (4) definición final del alcance mediante la selección de aquellos dominios que deben formar parte del núcleo de recursos

[18] Komponente es la denominación que utiliza KobrA para designar a los componentes de la arquitectura software de una línea de productos.

reutilizables a partir de los que se construirán los distintos VIPS específicos. En los siguientes apartados se recogen los resultados obtenidos en cada una de estas etapas.

4.2.1. Descripción de las distintas familias de productos VIPS

En este apartado se describen las características de los VIPS que se desea poder derivar de la línea LPS-VIPS, agrupándolos en las seis familias presentadas en el Capítulo 2: Sistemas de Inspección Visual Automatizada (SIVA), Sistemas BIOmétricos (SBIO), Sistemas de Imagen MÉDica (SIMED), Sistemas de Navegación (SNAV), Interfaces PERceptuales (IPERC), y Sistemas de Información Geográfica (SIG).

Para describir cada una de estas familias de VIPS se empleará la plantilla recogida a continuación en el cuadro de la Figura 26. Esta plantilla es una adaptación de la propuesta por K. Schmid en [Schmid 01].

P1 - Nombre: Nombre de la familia de VIPS.

P2 - Descripción: Breve descripción.

P3 - Estado actual: Seleccionar una o más opción de la siguiente lista:
Hipotético | Planificado | En producción | Mantenimiento | Preexistente

P4 - Funcionalidad principal que proporcionan:

FP_1 Funcionalidad Principal 1
...
FP_n Funcionalidad Principal N

En ocasiones también proporcionan:

FO_1 Funcionalidad Opcional 1
...
FO_n Funcionalidad Opcional N

P5 - Segmento del mercado al que va dirigido el producto:

- **Segmento de precios**.
- **Segmento de usuarios**.
- **Segmento funcional**.

P6 - Características que lo diferencian de otros productos:

CDif Características diferenciales

P7 - Sistemas individuales: descripción del proceso de producción (a medida, en serie, varias copias para un mismo cliente, etc.)

P8 - Posibles adaptaciones necesarias: posibles fuentes de variabilidad.

P9 - Restricciones no funcionales

RNF_1 Restricción no funcional 1
...
RNF_n Restricción no funcional N

Figura 25. Plantilla para la descripción de las familias de productos.

A continuación, utilizando la plantilla mostrada en la Figura 25 se describen las seis familias de VIPS antes mencionadas.

► *Descripción de los productos de la familia SIVA*

P1 - Nombre: Sistemas de Inspección Visual Automatizada (SIVA).

P2 - Descripción: Sistemas para la inspección visual automática de la calidad de todo tipo de productos industriales. Permiten detectar uno o más tipos de defectos en el 100% de la producción. Por lo general interactúan con distintos tipos de sensores (cámaras, sensores de paso, de temperatura, de presión, etc.) y actuadores (motores, electro-válvulas, etc.) siendo común que el sistema lleve a cabo operaciones de control y sincronización de los procesos y dispositivos.

P3 - Estado actual: Hipotético | Planificado | **En producción** | Mantenimiento | **Preexistente**
Varios SIVA en fase de desarrollo y otros ya construidos por el grupo de investigación DSIE [Fernández 99] [Fernández 01] [Martínez 04] [Vicente 04b].

P4 - Funcionalidad principal que proporcionan:

- **FP_1** Arrancar y parar el SIVA y los dispositivos/sistemas asociados.
- **FP_2** Iniciar/detener el proceso de inspección.
- **FP_3** Calibrar de forma asistida y/o automática los sensores y/o actuadores.
- **FP_4** Configurar off-line ciertos parámetros del SIVA de forma manual o (semi-) automática.
- **FP_5** Gestionar las alarmas del sistema: falta/exceso de producto, fallo de un dispositivo, fallo del control o las comunicaciones, etc.

En ocasiones también proporcionan:

- **FO_1** Controlar el acceso al sistema de los distintos usuarios (gestión de permisos).
- **FO_2** Configurar on-line los parámetros del SIVA de forma manual o (semi-) automática.
- **FO_3** Generar informes: productos defectuosos frente a productos inspeccionados, productos inspeccionados a la hora, etc.

P5 - Segmento del mercado al que va dirigido el producto:

- **Segmento de precios:** medio - alto.
- **Segmento de usuarios:** industrias de muy diversos sectores que precisan asegurar un alto nivel de calidad en sus productos (siempre que ésta pueda determinarse de forma visual).
- **Segmento funcional:** inspección visual de la calidad.

P6 - Características que lo diferencian de otros productos:

- **CDif** Dado que los SIVA operan en entornos industriales (sucios, ruidosos, etc.) deben implementarse utilizando elementos robustos; por ejemplo, las comunicaciones (si las hay) suelen implementarse sobre sistemas especialmente concebidos para su uso en este tipo de entornos (por ejemplo PROFIBUS o PROFINET).

P7 - Sistemas individuales. Por lo general se diseña y construye un SIVA específico para cada producto; a veces incluso uno para cada cliente (sistemas a medida). Algunas empresas requieren la instalación en paralelo de varios SIVA idénticos para aumentar la velocidad de inspección.

P8 - Posibles adaptaciones necesarias. Existen varias fuentes de variabilidad: (1) tipo de implementación (HW, SW o mixto) según el rendimiento mínimo exigido y el coste máximo asumible, (2) tipo de sensores y actuadores requeridos, (3) tipo de iluminación empleado, (4) tipo de imágenes obtenidas (color, escala de grises, B/N), etc.

P9 - Restricciones no funcionales:

- **RNF_1** Elevada velocidad de respuesta: no deben retrasar el ritmo de producción.
- **RNF_2** Robustez (capacidad para reaccionar ante situaciones anormales).
- **RNF_3** Facilidad de uso e interpretación del estado del sistema.
- **RNF_4** En ocasiones se requiere una alta precisión.

► ***Descripción de los productos de la familia SBIO***

P1 - Nombre: Sistemas de Seguridad Basados en Biométricas (SBIO).

P2 - Descripción: Sistemas utilizados para identificar[19] o validar la identidad[20] de un individuo mediante el análisis de uno o más de sus rasgos biométricos. En la mayoría de los casos estos sistemas se emplean para restringir el acceso a ciertos lugares o recursos (ficheros, cuentas bancarias, etc.). También pueden considerarse sistemas de este tipo los dedicados a identificar ciertos comportamientos considerados sospechosos en determinados contexto (persona parada largo tiempo en una estación de metro, persona corriendo en un supermercado, etc.).

P3 - Estado actual: **Hipotético** | Planificado | En producción | Mantenimiento | Preexistente
En contacto con dos empresas del sector: Vision Base Ltd. y Landa Digital Vision S.L.

P4 - Funcionalidad principal que proporcionan:

- **FP_1** Calibrar de forma asistida y/o automática el sistema.
- **FP_2** Iniciar el proceso de identificación/verificación.
- **FP_3** Obtener registros de identidad de una base de datos.
- **FP_4** Gestionar alarmas: intento de acceso ilegal, persona sospechosa, etc.

En ocasiones también proporcionan:

- **FO_1** Arrancar y parar el SBIO y los dispositivos/sistemas asociados.
- **FO_2** Detener el proceso de identificación cuando éste sea continuo.
- **FO_3** Configurar ciertos parámetros del SBIO: horarios de acceso permitido/ restringido/prohibido, cambiar permisos de ciertos usuarios, etc.
- **FO_4** Controlar el acceso al sistema de los distintos administradores.
- **FO_5** Alta de nuevos registros en la base de datos de identidades.
- **FO_6** Generar informes: lugar/recurso accedido, intentos ilegales, etc.

[19] Identificación biométrica: averiguar la identidad de una persona en función de uno o más de sus rasgos biométricos comparándolos con los registrados en una base de datos. Este proceso puede requerir o no la interacción y/o el consentimiento de la persona que se trata de identificar.

[20] Validación biométrica: averiguar si una persona es quien dice ser analizando uno o más de sus rasgos biométricos. El individuo deberá identificarse ante el sistema bien por medio de dispositivos habilitados a tal efecto (tarjeta magnética, teclado, etc.), o bien utilizando un entorno perceptual (por ejemplo, un reconocedor de voz).

P5 - Segmento del mercado al que va dirigido el producto:

- **Segmento de precios:** medio - alto.
- **Segmento de usuarios:** grandes y medianas empresas (públicas o privadas) que requieran un control de las personas que acceden a sus instalaciones y/o recursos.
- **Segmento funcional:** seguridad.

P6 - Características que lo diferencian de otros productos:

CDif Uso de canales de comunicación y protocolos seguros.

P7 - Sistemas individuales. Estos sistemas suelen desarrollarse a medida del cliente pudiendo éste adquirir varias copias del mismo para instalarlas en sus distintas sedes. Existen algunas soluciones comerciales disponibles para el público en general a modo de COTS.

P8 - Posibles adaptaciones necesarias. Existen varias fuentes de variabilidad (1) tipo de implementación (HW, SW o mixta), (2) sensores y actuadores requeridos en función de la/s biométrica/s que sea necesario analizar y del tipo de interacción con el usuario, (3) necesidad o no del uso de una base de datos de identidades; de resultar ésta necesaria su tamaño puede variar considerablemente así como el modelo de gestión empleado (centralizado o distribuido), (4) nivel de seguridad requerido (medio, alto o crítico), etc.

P9 - Restricciones no funcionales:

RNF_1 Elevada fiabilidad y robustez. En ocasiones un falso negativo (rechazar el acceso a alguien con permiso) puede resultar admisible. Por el contrario, un falso positivo (permitir el acceso a alguien sin permiso) resulta inaceptable en la mayoría de los casos.

► ***Descripción de los productos de la familia SIMED***

P1 - Nombre: Sistemas de Imagen Médica (SIMED)

P2 - Descripción: Sistemas que utilizan imágenes obtenidas de pacientes empleando muy diversas técnicas (CT, RMI, microscopía electrónica, etc.). En la mayoría de los casos estos sistemas se emplean con fines diagnósticos, ofreciendo al especialista la información visual obtenida del paciente (previamente procesada o no) así como ciertas medidas inferidas a partir de las imágenes (distancias entre puntos significativos, conteo de regiones, etc.). Algunos de estos sistemas permiten reconstruir de manera virtual y tridimensional aquellas partes/órganos del cuerpo que se están explorando a fin de mejorar la comprensión de su morfología, función, etc.

P3 - Estado actual: **Hipotético** | Planificado | En producción | Mantenimiento | Preexistente

P4 - Funcionalidad principal que proporcionan:

FP_1 Iniciar el proceso de captura y análisis de las imágenes.

FP_2 Calibrar de forma asistida y/o automática el sistema.

En ocasiones también proporcionan:

FO_1 Arrancar y parar el sistema y los dispositivos asociados.

FO_2 Detener el proceso de captura de imágenes bajo demanda.

FO_3 Configurar ciertos parámetros: zona/órgano analizado, tiempo e intensidad de exposición, tipo de contraste/tintura empleado, etc.

FO_4 Controlar el acceso al sistema de los distintos especialistas.

FO_5 Generar informes: copia de la imagen, datos del paciente y de la prueba, datos útiles para el diagnóstico, notas del especialista, etc.

FO_6 Obtener y/o actualizar la historia de un paciente de una base de datos.

FO_7 Gestionar las alarmas: funcionamiento incorrecto de sensores y/o actuadores, etc.

FO_8 Reconstrucción virtual tridimensional de la región analizada.

P5 - Segmento del mercado al que va dirigido el producto:

- **Segmento de precios:** medio - alto.
- **Segmento de usuarios:** especialistas médicos, laboratorios de análisis clínicos, centros médicos y hospitales, centros de investigación y enseñanza de la medicina.
- **Segmento funcional: medicina.**

P6 - Características que lo diferencian de otros productos:

CDif Utilizan técnicas/dispositivos propios (a veces exclusivos) de la medicina.

P7 - Sistemas individuales. Dado su elevado coste suelen fabricarse en serie para varios clientes.

P8 - Posibles adaptaciones necesarias. Existen varias fuentes de variabilidad: (1) tipo de implementación (HW, SW o mixto), (2) tipo de sensores y/o actuadores requeridos en función de las técnicas diagnósticas empleadas, (3) tipo de imágenes obtenidas (color o escala de grises), (4) necesidad o no de relacionar la información visual obtenida (por ejemplo, construcción de modelos 3D a partir de imágenes 2D), etc.

P9 - Restricciones no funcionales:

RNF_1 Requisitos asociados con la seguridad y el bienestar del paciente: control del nivel de radiación, facilidad de acceso para personas con baja movilidad, etc.

RNF_2 Fiabilidad y robustez resultan esenciales ya que de la corrección del proceso y de los resultados depende la del diagnóstico.

RNF_3 Algunos de estos sistemas requieren una alta precisión (por ejemplo, los empleados para guiar intervenciones quirúrgicas).

RNF_4 Facilidad de uso e interpretación de los resultados.

► ***Descripción de los productos de la familia SNAV***

P1 - Nombre: Sistemas de Ayuda a la Navegación (SNAV).

P2 - Descripción: Sistemas que emplean la información visual que perciben del entorno para guiar el desplazamiento de un determinado vehículo. En la mayoría de los casos estos sistemas emplean fuentes de información que complementan la de tipo visual (ultrasonidos, GPS, etc.). Algunos SNAV como los implementados en robots móviles (vehículos no tripulados que normalmente se desenvuelven en entornos estructurados) permiten la navegación autónoma (sin supervisión ni intervención externa). Sin embargo, en el caso de vehículos tripulados (automóviles, embarcaciones, aviones, etc.), los SNAV suelen emplearse como una guía o ayuda para el conductor quien, en última instancia, es quien controla el vehículo.

P3 - Estado actual: Hipotético | **Planificado** | En producción | Mantenimiento | Preexistente
Se han realizado algunos estudios sobre navegación autónoma de robots en entornos estructurados empleando *landmarks* naturales.

P4 - Funcionalidad principal que proporcionan:

FP_1 Arrancar y parar el SNAV y los dispositivos/sistemas asociados.

FP_2 Iniciar/detener el proceso de navegación.

FO_3 Calibrar de forma asistida y/o automática los sensores y/o actuadores.

FO_4 Configurar parámetros del SNAV de forma manual o (semi-) automática.

FP_4 Gestionar las alarmas: impacto con un obstáculo, fallo de un dispositivo, situación de bloqueo, fallo de energía, etc.

En ocasiones también proporcionan:

FO_1 Controlar el acceso al sistema de los distintos usuarios (gestión de permisos y configuraciones personales).

FP_2 Configurar on-line ciertos parámetros del SNAV: velocidad, ruta, cambio de destino, etc.

FO_3 Generar informes: paso junto a una determinada marca o señal, tiempo en alcanzar el destino, alarmas surgidas a lo largo del recorrido, etc.

FO_4 Reconstruir virtualmente el entorno por el que se navega (2D o 3D).

P5 - Segmento del mercado al que va dirigido el producto:

- **Segmento de precios:** medio - alto.
- **Segmento de usuarios:** empresas fabricantes de todo tipo de vehículos.
- **Segmento funcional:** navegación (semi)-autónoma de vehículos.

P6 - Características que lo diferencian de otros productos:

CDif La principal diferencia de estos sistemas frente a otros VIPS es su carácter móvil. Junto con los SIVA son probablemente los VIPS en los que mayor peso tiene el componente de control. Los SNAV como los IPERC son sistemas que interactúan con su entorno, por lo que deben percibir no sólo lo que éste contiene sino lo que ocurre en él.

P7 - Sistemas individuales. Suelen fabricarse e instalarse en serie para cada tipo/marca de vehículo.

P8 - Posibles adaptaciones necesarias. Existen varias fuentes de variabilidad: (1) tipo de implementación (HW, SW o mixto) según el rendimiento, coste, peso, consumo energético requerido etc., (2) tipo de sensores y actuadores necesarios, (3) necesidad o no de combinar distintas fuentes de información (imágenes, ultrasonidos, láser, GPS, etc.), (4) distintos modos de navegación (automática, semi-automática o manual), etc.

P9 - Restricciones no funcionales:

RNF_1 Velocidad de respuesta adecuada a la velocidad de navegación: cuanto más rápido se desplace el sistema, más velozmente deberá reaccionar, sobre todo en situaciones de riesgo.

RNF_2 Robustez y fiabilidad (sobre todo en sistemas instalados en vehículos tripulados).

RNF_3 En ocasiones se requiere una alta precisión (por ejemplo, cuanto el entorno es peligroso).

RNF_4 En ocasiones el peso y el consumo energético del sistema resultan críticos (por ejemplo, en el caso de los robots de exploración espacial).

► *Descripción de los productos de la familia IPERC*

P1 - Nombre: Interfaces Perceptuales (IPERC)

P2 - Descripción: Sistemas que facilitan la interacción hombre-máquina utilizando dispositivos y procedimientos más sencillos e intuitivos de aprender y manejar que los tradicionales (pantalla, ratón, teclado, etc.). En particular, se desarrollarán IPERC que utilicen información visual como principal fuente de información y/o salida del sistema, pudiendo ésta combinarse (de manera opcional) con otras de tipo auditivo, táctil, etc.

P3 - Estado actual: Hipotético | Planificado | **En producción** | Mantenimiento | **Preexistente**
Dos IPERC ya construidos para (1) reconocimiento de expresiones faciales [García 00] [García 01a] [García 01b] y (2) seguimiento de caras en imágenes de vídeo [García 02] [Vicente 02]. Otro IPERC en fase de desarrollo para video-conferencia en tiempo real en un entorno virtual animado**.**

P4 - Funcionalidad principal que proporcionan:

FP_1 Iniciar/detener el proceso de interacción perceptual hombre-máquina.

FP_2 Calibrar de forma asistida y/o automática los sensores y/o actuadores.

En ocasiones también proporcionan:

FO1 Controlar el acceso al sistema de los distintos usuarios (gestión de permisos y configuraciones personales).

FO_2 Configurar los parámetros del IPERC de forma manual o (semi-) automática.

FO_3 Gestionar las alarmas del sistema (fallo en los dispositivos y/o en las comunicaciones).

FO_4 Generar informes (dispositivos convencionales utilizados a la vez que la interfaz perceptual para interactuar con el sistema, etc.).

FO_5 Reconstrucción virtual del entorno con el que se interactúa.

P5 - Segmento del mercado al que va dirigido el producto:

- **Segmento de precios:** bajo-medio.
- **Segmento de usuarios:** usuarios particulares, centros de enseñanza, etc.
- **Segmento funcional:** interacción hombre-máquina (HCI).

P6 - Características que lo diferencian de otros productos:

CDif A diferencia de otros VIPS, la mayoría de los IPERC requieren de dispositivos especiales de interacción con el usuario (gafas de visualización 3D, cámaras portátiles o de movimiento controlado, etc.). Estos sistemas pueden requerir la aplicación de técnicas de visión artificial de alto nivel que permitan percibir el entorno 3D en el que se desenvuelve el usuario.

P7 - Sistemas individuales. Suelen fabricarse en serie para el público en general.

P8 - Posibles adaptaciones necesarias. Existen varias fuentes de variabilidad: (1) tipo de implementación (HW, SW o mixta) según el rendimiento mínimo exigido y el coste máximo asumible, (2) tipo de sensores requeridos, (3) necesidad o no de combinar distintas fuentes de información (por ejemplo, imágenes y voz), etc.

P9 - Restricciones no funcionales:

- **RNF_1** Elevada velocidad de respuesta de modo que la interacción con el usuario sea fluida.
- **RNF_2** Facilidad de uso y configuración.
- **RNF_3** Cuando se trata de IPERC portátiles factores como el peso o el consumo energético pueden resultar críticos.

► ***Descripción de los productos de la familia SIG***

P1 - Nombre: Sistemas de Información Geográfica (SIG).

P2 - Descripción: En la mayoría de los casos se trata de sistemas distribuidos de tipo cliente-servidor. Las aplicaciones servidor incorporan una base de datos en la que se almacenan las imágenes (preprocesadas o no) que se reciben desde una o más fuentes (satélites por lo general). Los clientes (aplicaciones de tipo VIPS) acceden de forma local o remota (por ejemplo, a través de una página web) a la base de datos de imágenes del servidor. Por lo general, es la aplicación cliente la que se encarga de procesar las imágenes y extraer de ellas la información relevante en cada caso (localización de objetivos, seguimiento de fenómenos meteorológicos, construcción de mapas tridimensionales, etc.).

P3 - Estado actual: **Hipotético** | Planificado | En producción | Mantenimiento | Preexistente

P4 - Funcionalidad principal que proporcionan:

- **FP_1** Iniciar la obtención y el procesamiento de las imágenes.
- **FP_2** Configurar ciertos parámetros del SIG de forma manual o (semi-) automática (resolución, velocidad de refresco, selección del satélite y/o base de datos que proporciona las imágenes, etc.).

En ocasiones también proporcionan:

- **FO_1** Controlar el acceso al sistema de los distintos usuarios (gestión de permisos).
- **FO_2** Generar informes (eventos críticos detectados: hora, características y persona que los atendió; informes de tendencias detectadas en secuencias de imágenes, etc.).
- **FO_3** Obtener registros de bases de datos (imágenes vía satélite).
- **FO_4** Gestionar las alarmas del sistema (evento crítico detectado, fallo en el canal de comunicaciones o en la BD, etc.).
- **FO_5** Reconstrucción virtual 2D o 3D (por ejemplo, mapas orográficos o perfiles de ciudades).

P5 - Segmento del mercado al que va dirigido el producto:

- **Segmento de precios:** todos.
- **Segmento de usuarios:** organismos públicos y privados de carácter civil o militar. Últimamente, algunas empresas han comenzado a proporcionar versiones simplificadas de estos productos de forma pública y gratuita a través de la Internet (por ejemplo, Google™ Earth [GEARTH]).
- **Segmento funcional:** teledetección, meteorología, urbanismo, etc.

P6 - Características que lo diferencian de otros productos:

CDif La mayoría de estos sistemas se nutren de imágenes obtenidas vía satélite que son almacenadas en enormes bases de datos para su posterior consulta. En consecuencia, una correcta gestión de las comunicaciones y de las bases de datos de imágenes resulta esencial (tareas centrales en los SIG que no lo son en otros VIPS).

P7 - Sistemas individuales. Suelen fabricarse en serie y venderse a clientes corporativos.

P8 - Posibles adaptaciones necesarias. Existen varias fuentes de variabilidad: (1) tipo de implementación (HW, SW o mixto) según el rendimiento mínimo exigido y el coste máximo asumible, (2) Tipo de acceso a las bases de datos de imágenes (local/remoto, público/restringido, en tiempo real/en diferido, etc.), (3) necesidad o no de relacionar la información visual obtenida de una o más fuentes (por ejemplo, elaboración de mapas 3D a partir de imágenes 2D), o de combinar las imágenes con otras fuentes de información (mapas de temperatura, presión atmosférica, etc.).

P9 - Restricciones no funcionales:

RNF_1 Facilidad para localizar la información requerida y para interpretar los resultados.

RNF_2 En ocasiones se precisa poder manejar imágenes de muy alta resolución o de resolución variable (el usuario selecciona el detalle con el que quiere ver las imágenes).

RNF_3 En aplicaciones de teledetección de eventos críticos (incendios, huracanes, etc.) se requiere una velocidad de respuesta elevada.

4.2.2. Descripción de los principales dominios o áreas funcionales

Durante esta etapa se utilizan las descripciones anteriores para identificar y describir la funcionalidad que ofrecen los distintos tipos de VIPS, entendida como el conjunto de servicios que proporcionan externamente. Para ello, se elaborará una primera versión del mapa de productos en el que se recogerán las características descritas en cada familia de VIPS, agrupándolas por dominios o áreas funcionales. Este mapa permitirá identificar a simple vista cuáles de estas características aparecen en todos, en varios o sólo en algunos de los productos.

A continuación, se procederá a describir detalladamente cada uno de los dominios utilizando para ello una plantilla que permitirá recoger no sólo la funcionalidad externa previamente identificada en las descripciones de los productos sino también la siguiente información:

- Funcionalidad interna que ofrece cada dominio al resto de dominios identificados en la fase anterior. A veces, como resultado de la especificación de esta funcionalidad pueden identificarse nuevos dominios internos. También puede ocurrir que si la funcionalidad interna de un domino es muy extensa resulte conveniente repartirla en varios subdominios de modo que se mantenga, en la medida de lo posible, el principio de máxima cohesión y mínimo acoplamiento entre ellos. Tanto los nuevos dominios internos identificados como los nuevos subdominios que se creen deberán describirse también utilizando la misma plantilla.

- Información relativa a cómo se relacionan los distintos dominios entre sí. Esta información permitirá elaborar un mapa de la estructura funcional de las aplicaciones que se van a derivar de la LPS y posteriormente servirá para identificar los subsistemas que forman parte de su arquitectura software.

Por último, a partir de la descripción de los dominios, deberá establecerse cuáles de ellos se desarrollarán como parte del núcleo de recursos reutilizables, cuáles se incorporarán al mismo desde sistemas previamente existentes (bien de desarrollo propio o adquiridos a terceros) y cuáles quedarán fuera del alcance de la LPS. Esta decisión se tomará en función de criterios tales como la cantidad de productos que incorporan la funcionalidad de un determinado dominio (grado de reutilización), experiencia previa y/o existencia de recursos ya desarrollados en un área funcional, etc.

El hecho de excluir un dominio del alcance de una LPS no implica que las aplicaciones que se construyan a partir de ella no puedan incorporar su funcionalidad, pero sí que ésta deberá desarrollarse como una extensión o adaptación específica sólo para aquellas aplicaciones que lo requieran. Suele ocurrir, de hecho con relativa frecuencia, que a media que la LPS evoluciona se van detectando e incorporando nuevos dominios a su alcance, entre los que pueden encontrarse algunos de los inicialmente descartados.

En los siguientes aparatados se presentan los resultados obtenidos en cada uno de estos procesos.

4.2.2.1. Tabla de características (agrupadas por dominios)

A partir de las descripciones elaboradas en el apartado 4.2.1 se ha construido el siguiente mapa de productos en el que se incluyen las características identificadas para en cada una de las familias de la línea LPS-VIPS. Nótese que en los mapas de características elaborados siguiendo la metodología PuLSE™ cada una de las casillas aparece o bien marcada (cuando el producto ofrece dicha característica) o bien vacía (en caso contrario). Sin embargo, dado que en este caso la línea LPS-VIPS es una línea de familias de productos, tiene sentido decir que una característica aparece con una mayor o menor probabilidad en los productos de una determinada familia por lo que se ha optado por utilizar distintos símbolos para reflejar este hecho.

		Tipos de VIPS					
Dominio	**Característica**	**SIVA**	**SBIO**	**SIMED**	**SNAV**	**IPERC**	**SIG**
Operativa Básica	Arrancar VIPS	■		○	■		
	Parar VIPS	■		○	■		
	Iniciar proceso	■	■	■	■	■	■
	Detener proceso[21]	■	○	○	■	■	○
	Calibrar sistema	■	■	■	■	■	○
	Identificar usuario	◘	◘	◘	◘	◘	◘
Gestión de Alarmas	Iniciar alarma de usuario[22]	■		○	■		
	Detener alarma sistema/usuario	◘	◘	◘	◘	◘	◘
Gestión de Informes	Crear informes	◘	◘	◘	◘	◘	◘
	Imprimir informes	◘	◘	◘	◘	◘	◘
	Guardar informes	◘	◘	◘	◘	◘	◘
	Cargar informes previos	◘	◘	◘	◘	◘	◘
Gestión de bases de datos	Buscar registros		■	◘			◘
	Crear nuevo registros		◘	◘			
	Modificar registros			○			
	Eliminar registros						
Virtualización	Iniciar/detener entorno virtual			◘	○	◘	◘
Gestión de configuración	Configurar proceso	◘	◘	◘	◘	◘	◘
	Configurar alarmas	◘	◘				◘
	Configurar informes	○	○	○			○
	Configurar acceso a BD		○	○			○
	Configurar virtualización			○	○	○	○

Convención gráfica utilizada:
■ Funcionalidad ofrecida por la mayoría de los VIPS de una familia.
◘ Funcionalidad ofrecida por varios de los VIPS de una familia.
○ Funcionalidad ofrecida sólo en algunos de los VIPS de una familia.
Casilla vacía: funcionalidad típicamente no disponible.

Tabla 6. Mapa inicial de características de las distintas familias de VIPS.

Como se observa en el mapa de características mostrado en la Tabla 6, los usuarios de las distintas familias de VIPS tienen acceso a diversas operaciones (características) que se han agrupado en seis dominios o categorías funcionales: operativa básica, gestión de alarmas, gestión de informes, gestión de bases de datos, virtualización y gestión de configuración.

[21] "*Detener proceso*" sólo está disponible cuando el proceso iniciado es continuo (por ejemplo cuando se procesa vídeo procedente de una cámara o de un fichero). En los casos en los que las imágenes se procesan individualmente el proceso se detiene por sí solo.

[22] El sistema puede generar alarmas como respuesta a ciertos errores detectados internamente. Sin embargo en esta tabla sólo se recogen las características funcionales a las que puede acceder directamente el usuario (no la funcionalidad interna del VIPS). El usuario podrá detener tanto las alarmas generadas por él o por otro usuario como las generadas por el sistema.

Dado que este mapa inicial de características sólo recoge la funcionalidad externamente visible de los distintos VIPS (operaciones a las que tiene acceso el usuario), una posible forma de interpretar esta tabla es considerar cada dominio como uno de los menús ofrecidos por el VIPS y cada una de sus características como una de las operaciones disponibles en dicho menú. De este modo, por ejemplo, un SIVA típico ofrecería al usuario cuatro menús correspondientes a los cuatro dominios en los que hay alguna característica marcada (se excluyen los menús de gestión de bases de datos y de virtualización). Siguiendo con el ejemplo, al seleccionar el menú de operativa básica, al usuario se le ofrecerá la posibilidad de arrancar/parar el sistema y sus dispositivos asociados, iniciar/detener el proceso de inspección y, en algunos casos (dependiendo del SIVA), la opción de calibrar el sistema y/o la de identificarse para acceder a ciertos recursos restringidos.

4.2.2.2. Descripción de los dominios identificados

Una vez descrita y clasificada la funcionalidad externa proporcionada por los distintos tipos de VIPS, en esta fase se describen detalladamente cada uno de los dominios identificados. Estas descripciones seguirán el esquema de la plantilla que se muestra a continuación en el cuadro de la Figura 26.

P1 - Nombre: Nombre del dominio

P2 - Descripción: Breve descripción.

P3 - Funcionalidad ofrecida y demandada por el dominio:
- **Entradas** (funcionalidad demandada)
- **Salidas** (funcionalidad ofrecida)

P4 - Dominios de orden superior: dominios de los que depende.

P5 - Dominios de orden inferior: dominios que dependen de éste.

P6 - Subdominios:

P7 - Funcionalidad proporcionadas por el dominio:

F1	**Funcionalidad_1**
…	**…**
F_n	**Funcionalidad_N**

P8 - Datos manejados/almacenados en el dominio:
- **Datos de entrada:** datos que recibe el dominio.
- **Datos de salida:** datos que envía el dominio.
- **Datos almacenados:** datos almacenados en el dominio.

P9 - Recursos preexistentes.

P10 - Productos en los que se implementará la funcionalidad de este dominio.

Figura 26. Plantilla para la descripción de dominios.

Como ya se ha comentado con anterioridad, estas descripciones permitirán acotar la funcionalidad (interna y externa) asociada a cada dominio[23] así como determinar las relaciones de dependencia y jerarquía existentes entre ellos. A lo largo de este proceso pueden surgir nuevos dominios internos[24] no identificados previamente, así como algunos subdominios resultantes de particionar la funcionalidad de alguno de los dominios previamente descritos. En ambos casos estos nuevos dominios deberán también describirse utilizando la misma plantilla.

Para evitar alargar en exceso este capítulo y facilitar su lectura aquí sólo se recoge, a modo de ejemplo, la descripción del dominio de gestión de la operativa básica del VIPS. La descripción del resto de los dominios puede encontrarse al final de la memoria en el Apéndice II.

► ***Descripción del dominio de gestión de la operativa básica del VIPS***

P1 - Nombre: Gestión de operativa básica.

P2 - Descripción: Proporciona la funcionalidad básica del VIPS (arranque y parada del sistema, inicio y detención del proceso de análisis de la información visual, etc.).

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde la interfaz de usuario se reciben los distintos comandos asociados al menú de operativa básica del VIPS.
  - Desde el *gestor central del VIPS* (dominio interno) se reciben los resultados de los procesos internos: procesamiento de la información visual, control, comunicaciones y alarmas.
- **Salidas:**
  - Al *gestor central del VIPS* se le envían las órdenes recibidas del usuario para que éste las atienda convenientemente (por ejemplo, cuando el estado del sistema lo permita).
  - A la interfaz de usuario se le envían los resultados de los procesos internos y las alarmas. En función de la configuración de la aplicación y/o de los permisos que tenga el usuario activo algunos de estos resultados y/o alarmas pueden no enviarse (filtrado).

P4 - Dominios de orden superior (de los que éste depende): Interfaz de usuario.

P5 - Dominios de orden inferior (que dependen de éste): Gestor central del VIPS.

P6 - Subdominios: ---

[23] Siguiendo con el ejemplo anterior en el que las características de cada dominio podían verse como opciones asociadas a los distintos menús de un VIPS, cabría identificar la funcionalidad interna asociada a un dominio como el conjunto de operaciones que se desencadenan en el sistema al seleccionar una de las opciones correspondientes al menú asociado a ese dominio.

[24] Para facilitar la identificación de los dominios internos éstos se escribirán en cursiva.

P7 - Funciones / características proporcionadas por el dominio:

- **F_1 Identificar usuario**: verificar la identidad de un usuario que trata de acceder al sistema; si la verificación es correcta establecer la configuración y los permisos correspondientes a dicho usuario.
- **F_2 Arrancar VIPS**: se inicializa el sistema cargando los parámetros de configuración actuales y se envía orden de arranque al *gestor central del VIPS* para que inicialice los dispositivos externos (sensores, actuadores, controladores, dispositivos de comunicaciones, etc.).
- **F_3 Parar VIPS**: se envía orden de parada al *gestor central del VIPS* para que detenga los dispositivos externos; una vez recibida la confirmación de que se ha completado esta operación se finaliza la aplicación.
- **F_4 Iniciar proceso**: se envía orden al *gestor central del VIPS* de que comience el procesamiento de la información visual; mientras no se detenga este proceso se visualizarán los resultados seleccionados por el usuario a medida que vayan llegando desde el *gestor central del VIPS*.
- **F_5 Detener proceso**: se envía orden al *gestor central del VIPS* de que detenga el procesamiento de la información visual.
- **F_6 Calibrar sistema:** se envía orden al *gestor central del VIPS* para que se realice la calibración de los sensores y/o actuadores del sistema.
- **F_7 Iniciar alarma de usuario**: Si el usuario detecta cualquier situación anómala podrá disparar en cualquier momento una señal de alarma.
- **F_8 Detener alarma**: El usuario detiene la alarma activa (notificada por el sistema o por otro usuario) de lo que se informa al *gestor central del VIPS*.
- **F_9 Configurar proceso**: en función de sus permisos el usuario puede modificar ciertos parámetros del VIPS tanto externos (por ejemplo, resultados que desea visualizar) como internos (por ejemplo, parámetros de control, comunicaciones, etc.).

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  - Desde la interfaz de usuario: parámetros de configuración del VIPS (internos y externos).
  - Desde el *gestor central del VIPS*: resultados de las solicitudes enviadas (información solicitada a los gestores de: informes, bases de datos, procesamiento visual, control y comunicaciones).
  - Desde el *gestor de alarmas*: notificaciones de alarmas activadas.
- **Datos de salida:**
  - A la interfaz de usuario: resultados solicitados (correctamente formateados) y notificaciones de alarmas activadas.
  - Al *gestor central del VIPS*: parámetros de configuración del VIPS (internos y externos) y notificación de las alarmas de usuario.
- **Datos almacenados:** usuario activo y nivel de permisos.

P9 - Recursos preexistentes: ---

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1, F_8 y F_9 con frecuencia en todos los VIPS.
- F_2, F_3 y F_7 en los SIVA y los SNAV; ocasionalmente también en los SIMED.
- F_4 en todos los VIPS.
- F_5 con frecuencia en los SIVA, SNAV e IPERC; ocasionalmente también en el resto de los VIPS.
- F_6 en todos los VIPS salvo en los SIG donde aparece sólo ocasionalmente.

A partir de la descripción exhaustiva de los dominios, en los apartados siguientes se extraerán aquellos aspectos más relevantes a la hora de decidir cuáles de todos ellos se desarrollarán como parte del núcleo de recursos comunes de la línea LPS-VIPS, para cuáles sólo se proporcionará una interfaz que permita implementar posteriormente su funcionalidad (a medida o a partir de productos ya existentes), y cuáles quedarán fuera del alcance de la línea de productos.

4.2.2.3. Estructura funcional de los VIPS

A partir de las descripciones elaboradas en el apartado anterior se ha elaborado el mapa de dominios que se muestra en la Figura 27, en el que se han recogido los dominios y subdominios identificados así como las relaciones existentes entre ellos. Los dominios coloreados en verde (dominios externos) aglutinan la funcionalidad directamente accesible por el usuario, mientras que los coloreados en naranja (filas inferiores) recogen la funcionalidad interna del VIPS; por último, los tres dominios dibujados sobre fondo violeta representan los subdominios identificados en el *gestor del procesamiento visual*.

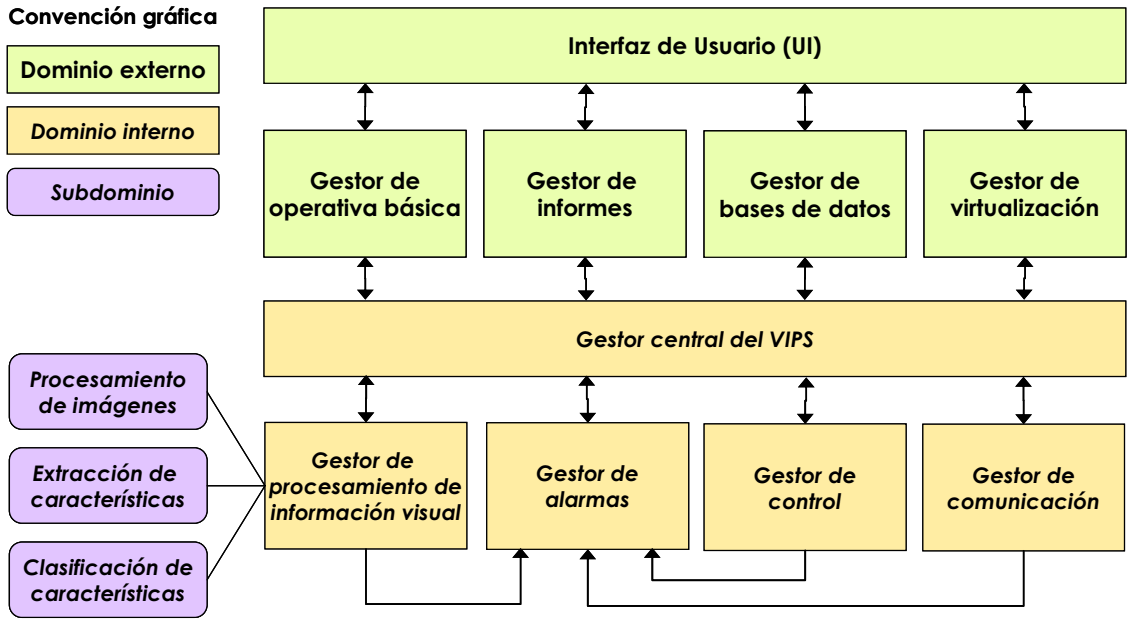


Figura 27. Mapa de estructura de dominios.

Como se observa en la figura anterior, toda la estructura funcional del sistema gira en torno al *gestor central del VIPS* que actúa como mediador entre el resto de los dominios. La elección de este patrón de diseño (patrón mediador [Gamma 95])

responde, de una parte, a la necesidad de independizar la funcionalidad ofrecida por cada dominio de la del resto (el mediador minimiza el acoplamiento entre los distintos dominios) y de otra, a la posibilidad de tener que añadir nueva funcionalidad a la línea LPS-VIPS en el futuro (el uso de este patrón favorece la extensibilidad ya que la inclusión de un nuevo dominio afectaría -al menos ostensiblemente- sólo al mediador).

Sin embargo, se ha permitido una excepción a la regla de que todo dominio debe utilizar el mediador para comunicarse con el resto y es la relacionada con la notificación de situaciones de alarma por parte de los dominios internos que gestionan el procesamiento de la información visual, el control y las comunicaciones. La razón de que se permita a estos dominios notificar los posibles fallos detectados directamente al *gestor de alarmas*, en lugar de hacerlo a través del *gestor central del VIPS*, responde fundamentalmente a una cuestión de seguridad ya que, en la mayoría de los casos, las alarmas generadas en estos dominios tienen que ver con fallos en los dispositivos externos que gestionan (cámaras, digitalizadores, sensores, actuadores, etc.). Este tipo de fallos suelen ser graves o críticos por lo que resulta esencial que el sistema responda lo más rápidamente posible.

El *gestor central del VIPS* soporta una gran carga de tareas y, en ocasiones, sólo puede comprobar la llegada de nuevas notificaciones de alarma de manera síncrona cada cierto tiempo (por ejemplo, cada vez que se termina de procesar una imagen). Así, permitiendo que estas notificaciones lleguen directa y asíncronamente (sin esperar a que se complete ninguna operación) al *gestor de alarmas*, se consigue reducir sensiblemente el tiempo de respuesta y, en consecuencia, se evitan posibles daños que podrían resultar muy costosos de subsanar (por ejemplo, la rotura de algunos de los sensores y/o actuadores).

Una posible solución a este problema que evitaría violar la estructura del patrón mediador, consistiría en gestionar las alarmas mediante un mecanismo de excepciones. Sin embargo, esta solución presenta dos inconvenientes fundamentales; en primer lugar, las notificaciones de alarma que llegaran al *gestor central del VIPS* interrumpirían el proceso que éste estuviera llevando a cabo en ese momento y no siempre resultaría sencillo (a veces incluso sería imposible) recuperar el estado anterior a la excepción. En segundo lugar, no parece adecuado optar por una solución tan dependiente de la implementación (no todos los lenguajes proporcionan mecanismos de gestión de excepciones), más aun cuando en este punto apenas se está empezando a perfilar el diseño de los VIPS.

4.2.2.4. Mapa inicial de productos

La descripción previa de los dominios identificados (ver 4.2.2.2), utilizada en el apartado anterior para confeccionar el mapa de la estructura funcional de los VIPS (ver Figura 27), permite además elaborar el mapa de productos mostrado a continuación en la Tabla 7) en el se recoge la funcionalidad (tanto externa como interna) ofrecida por los distintos tipos de VIPS.

		Tipos de VIPS					
Dominio	**Característica**	**SIVA**	**SBIO**	**SIMED**	**SNAV**	**IPERC**	**SIG**
Gestión de operativa básica	Identificar usuario	◘	◘	◘	◘	◘	◘
	Arrancar VIPS	■		○	■		
	Parar VIPS	■		○	■		
	Iniciar proceso	■	■	■	■	■	■
	Detener proceso	■	○	○	■	■	○
	Calibrar sistema	■	■	■	■	■	○
	Iniciar alarma de usuario	■		○	■		
	Detener alarma	◘	◘	◘	◘	◘	◘
	Configurar proceso	◘	◘	◘	◘	◘	◘
Gestión de informes	Crear informe puntual	◘	◘	◘	◘	◘	◘
	Crear informe continuo	◘	○	○	◘	◘	○
	Imprimir informe	◘	◘	◘	◘	◘	◘
	Guardar informe	◘	◘	◘	◘	◘	◘
	Cargar informe	◘	◘	◘	◘	◘	◘
	Configurar informes	○	○	○			○
Gestión de bases de datos	Buscar registro		■	◘			◘
	Crear nuevo registro		◘	◘			
	Modificar registro			○			
	Eliminar registro						
	Iniciar alarma de acceso a BD		◘	◘			◘
	Configurar acceso a BD		○	○			○
Virtualización	Iniciar reconstrucción virtual			◘	○	◘	◘
	Detener reconstrucción virtual			○	○	○	○
	Configurar reconstrucción virtual			○	○	○	○
Gestor de procesamiento visual	Arrancar sistema de visión	◘	◘	◘	◘	◘	
	Detener sistema de visión	◘	◘	◘	◘	◘	
	Calibrar cámaras	◘	◘	◘	◘	◘	
	Procesar información visual	■	■	■	■	■	■
	Iniciar alarma del sistema de visión	◘	◘	◘	◘	◘	◘
	Configurar procesamiento visual	◘	◘	◘	◘	◘	◘
Procesamiento De imágenes	Procesar imagen	■	■	■	■	■	■
	Configurar proces. de imágenes	◘	◘	◘	◘	◘	◘
Extracción de características	Extraer características	■	■	■	■	■	■
	Configurar extracción característic.	◘	◘	◘	◘	◘	◘
Clasificación características	Clasificar características	■	■	■	■	■	■
	Configurar clasificación	◘	◘	◘	◘	◘	◘
Gestión del control	Arrancar sistema de control	■	○	○	■	○	
	Detener sistema de control	■	○	○	■	○	
	Calibrar sistema de control	■	○	○	■	○	
	Regular VIPS	■	○	○	■	○	
	Configurar control	◘			◘		
Gestión de la comunicación	Enviar datos	◘	◘	○	○	○	○
	Recibir datos	◘	◘	◘	○	○	◘
	Configurar comunicación	○	○		○	○	◘
Gestión de Alarmas	Iniciar alarma del sistema	■	■	■	■	■	■
	Iniciar alarma de usuario	◘	◘				
	Detener alarma	◘	◘	◘			◘
	Configurar alarmas	◘	◘				◘

Convención gráfica utilizada:
- ■ Funcionalidad ofrecida por la mayoría de los VIPS de una familia.
- ◘ Funcionalidad ofrecida por varios de los VIPS de una familia.
- ○ Funcionalidad ofrecida sólo en algunos de los VIPS de una familia.
- Casilla vacía: funcionalidad típicamente no disponible.

Tabla 7. Mapa inicial de productos

Este mapa inicial de productos permite visualizar y comparar las características de las distintas familias de VIPS de manera sencilla e intuitiva. Como se puede comprobar, este mapa es una versión extendida y reorganizada del mostrado en la Tabla 6, en el que sólo se recogía la funcionalidad externa de los distintos tipos de VIPS.

4.2.3. Valoración y selección de dominios

En este apartado se valorará el coste y el beneficio que supone invertir en el desarrollo de cada uno de los dominios previamente identificados, considerando las expectativas de producción que se plantean para la línea LPS-VIPS a corto, medio y largo plazo. Tras este análisis, los dominios mejor valorados se incluirán como parte del núcleo de recursos reutilizables de la línea de productos.

El proceso de evaluación y selección de dominios se realizará en tres etapas: (1) evaluación de la relevancia y aportación de cada dominio para el desarrollo de los distintos tipos de VIPS, (2) evaluación de su potencial de reutilización en el ámbito de la línea LPS-VIPS, y (3) selección de los dominios más rentables y definición final del alcance la línea de productos.

4.2.3.1. Relevancia y aportación de cada dominio a los distintos VIPS

A la vista del mapa inicial de productos mostrado en la Tabla 7 y de las descripciones de los dominios realizadas en el apartado 4.2.2.2, resulta sencillo realizar una primera aproximación del peso que tiene cada uno de los dominios para los distintos tipos de VIPS. Por ejemplo, resulta evidente que en el caso los SIVA tiene mucho más peso el dominio de gestión del control que el de gestión de bases de datos, al contrario de lo que ocurre en el caso de los SIG.

En la siguiente tabla se muestra una valoración de la relevancia y el aporte funcional que proporciona cada dominio al desarrollo de los VIPS (ver Tabla 8). Para elaborar estas tablas se ha consultado a varios expertos en el desarrollo de distintos tipos de VIPS de modo que el resultado que se muestra es la media de todas sus valoraciones.

Tipo de VIPS / Dominio	SIVA	SBIO	SIMED	SNAV	IPERC	SIG	Relevancia y aportación del dominio
Operativa básica	9	7	7	7	9	7	7,7
Gestión de informes	7	6	8	3	0	7	5,2
Gestión de bases de datos	5	9	8	3	3	9	6,2
Virtualización	0	0	6	5	7	7	4,2
Gestión procesam. visual	8	8	9	8	8	9	8,3
Procesamiento de imágenes	8	8	8	7	7	9	7,8
Extracción de características	6	9	8	7	8	8	7,7
Clasificación características	7	9	5	7	8	6	7,0
Gestión del control	9	7	7	9	5	0	6,2
Gestión de la comunicación	3	7	5	4	5	9	5,5
Gestión de alarmas	7	9	7	9	5	7	7,3

Tabla 8. Valoración de la relevancia y aportación los dominios a los distintos VIPS.

Como se deduce de los resultados mostrados en la tabla anterior, los dominios más relevantes y que mayor funcionalidad aportan a la hora de construir nuevos VIPS son los relacionados con el procesamiento de la información visual (y sus subdominios), la gestión del control y la gestión de la operativa básica y las alarmas.

4.2.3.2. Potencial de reutilización de cada uno de los dominio

En este apartado se complementa el estudio anterior analizando el potencial de reutilización de cada uno de los dominios. Para ello, se han tenido en cuenta factores tales como:

- Su grado de variabilidad: aquellos dominios en los que la implementación de gran parte de su funcionalidad varía sensiblemente de un producto a otro son peores candidatos que aquellos que proporcionan una base funcional común y válida para varios de los productos de la línea.
- El grado de madurez y estabilidad del dominio: deberá tenerse en cuenta si se el dominio abarca áreas de conocimiento incipientes, relativamente asentadas o ya bien establecidas, así como la velocidad a la que evolucionan los conocimientos, las tecnologías y las herramientas relacionadas con el dominio en cuestión.
- El grado de cohesión dentro del dominio y de acoplamiento con otros dominios: aquellos dominios cuya funcionalidad dependa excesivamente de la proporcionada por otros dominios resultarán más difíciles de reutilizar y por lo tanto no resultarán buenos candidatos para formar parte del núcleo de recursos comunes de la línea de productos.
- La existencia de recursos previamente desarrollados que puedan reutilizarse sin necesidad de excesivas adaptaciones.

Como en el caso anterior, se ha solicitado a varios expertos tanto en el desarrollo de VIPS como en las áreas de conocimiento individuales relacionadas con cada uno de los dominios (visión artificial, control y automatización, informática, telecomunicación, etc.), que valoraran su potencial de reutilización atendiendo a los factores anteriormente expuestos. En la Tabla 9 se muestran los resultados obtenidos en la valoración de los distintos dominios (valores medios de las distintas evaluaciones realizadas por los expertos).

Como en el caso anterior las mejores valoraciones las obtienen los dominios y relacionados con el procesamiento de la información visual, algo que no es de extrañar considerando lo siguiente:

- En cuanto a la variabilidad en el dominio, aunque entre los distintos tipos de VIPS (e incluso entre los VIPS de una misma familia) existen diferencias considerables en el diseño de las tareas relacionadas con el procesamiento de la información visual (éstas pueden realizarse de forma continua o puntual, síncrona o asíncrona, etc.), existe una gran base común de algoritmos utilizados en la mayoría de los VIPS (algoritmos de umbralización, detección de bordes, segmentación por regiones o por contornos, reconocimiento de patrones, etc.).

- Las áreas de conocimiento implicadas (visión artificial, procesamiento de imágenes, reconocimiento de patrones, etc.) están bastante consolidadas en la actualidad y evolucionan a buen ritmo sin ser éste vertiginoso.
- En cuanto al grado de cohesión en el dominio éste suele ser muy alto si bien, cuando el procesamiento de la información visual debe sincronizarse con la gestión del control (como ocurre con frecuencia en los SIVA o los SNAV) existe un cierto acoplamiento entre estos dominios cuyo efecto puede minimizarse utilizando, por ejemplo, un patrón mediador como el propuesto en la Figura 27).
- Existen múltiples herramientas tanto software como hardware que facilitan la implementación de la funcionalidad de este dominio.

Dominio	Potencial de Reutilización
Operativa básica	5
Gestión de informes	5
Gestión de bases de datos	6
Virtualización	3
Gestión procesamiento visual	7
Procesamiento de imágenes	9
Extracción de características	9
Clasificación características	9
Gestión del control	6
Gestión de la comunicación	6
Gestión de alarmas	5

Tabla 9. Valoración del potencial de reutilización de los distintos dominios.

Por todo lo anterior, el dominio relacionado con el procesamiento de la información visual así como sus tres subdominios parecen candidatos adecuados a formar parte del núcleo de recursos comunes de la línea LPS-VIPS. Sin embargo, antes de tomar esta decisión deberá completarse la tercera fase del análisis cuyos resultados se muestran en el siguiente apartado.

4.2.3.3. Selección de dominios y definición final del alcance

La decisión final sobre qué dominios incluir como parte del núcleo de recursos reutilizables de la línea LPS-VIPS se realizará combinando y ponderando los resultados obtenidos en los análisis anteriores. Para ello, será necesario establecer:

- El factor o factores de ponderación que deben aplicarse al beneficio conjunto derivado del análisis de la relevancia y el grado de reutilización de cada dominio. En este caso se utilizará como factor de ponderación una valoración realizada por el equipo de personas involucradas en la especificación y posterior utilización de la línea LPS-VIPS sobre su experiencia previa en cada uno de los dominios (información recogida en las descripciones previas de los dominios).
- El coste que supondría desarrollar la funcionalidad de cada dominio si éste se excluyera del alcance de la LPS-VIPS, esto es, lo que costaría implementarlo de manera individualizada para cada nuevo producto. Como en el caso anterior,

la valoración de estos costes la ha realizado el equipo de personas involucradas en la especificación y la posterior utilización de la línea LPS-VIPS.

- Un criterio que permita clasificar los distintos dominios atendiendo a la relación beneficio/coste que se haya calculado. Para definir este criterio se ha utilizado como punto de partida el propuesto en el caso de estudio elaborado por D. Muthig para definir su línea de productos de dispositivos de comunicaciones móviles [Muthig 04]. Sin embargo, dado que los criterios de valoración y selección de un dominio pueden variar considerablemente de una línea de productos a otra, ha sido necesario realizar algunas adaptaciones para adecuar la regla de selección a la línea LPS- VIPS.

En el cuadro que se muestra en la Figura 28 se recogen los detalles del método empleado para valorar los distintos dominios, así como el criterio establecido para seleccionar aquellos que pasarán a formarán parte del alcance de la línea LPS-VIPS.

Valoración del dominio Di (Vi) = $\dfrac{\mathbf{Ri * Pi * Fi}}{\mathbf{3 * Ci}}$

Ri = Relevancia del dominio Di
Pi = Potencial de reutilización del dominio Di
Fi = Factor de ponderación aplicado al dominio Di
Ci = Coste de implementar el dominio para una aplicación individual

(Ri * Pi * Fi)/3 representa el beneficio de implementar el dominio como parte del núcleo común de recursos de la línea de productos

Criterio de selección de dominios en función de la valoración obtenida (Vi):

Decisión	**1**	**2**	**3**	**4**	**5**
Vi	**> 20**	**15 – 20**	**10 - 15**	**5 - 10**	**0-5**

1) Debe realizarse una implementación genérica del dominio.

2) Recomendable implementación genérica del dominio.

3) El dominio no se desarrollará como parte del núcleo de recursos comunes pero debe implementarse una interfaz que permita incorporar su funcionalidad sin alterar la arquitectura de la LPS.

4) El dominio no se desarrollará como parte del núcleo de recursos comunes aunque se recomienda implementar una interfaz que permita incorporar su funcionalidad sin alterar la arquitectura de la LPS.

5) El dominio queda fuera del alcance de la línea de productos. De ser necesario incorporar su funcionalidad en algún VIPS ésta deberá implementarse de manera específica para él.

Figura 28. Criterios de valoración y selección de dominios.

A continuación, en la Tabla 10, se muestran los resultados de aplicar este método de valoración y selección de dominios en el que se incorporan, entre otros, los resultados de los análisis previos relativos a la relevancia y el grado de reutilización de cada uno de los dominios.

	Ri	Pi	Fi	$\frac{Ri * Pi * Fi}{3}$	Ci	Vi	Decisión
Operativa básica	7,7	5	5	64,2	5	12,8	3
Gestión de informes	5,2	5	3	26,0	5	5,2	4
Gestión de bases de datos	6,2	6	3	37,2	7	5,3	4
Virtualización	4,2	3	3	12,6	9	1,4	5
Gestión procesamiento visual	8,3	7	8	154,9	8	19,4	2
Procesamiento de imágenes	7,8	9	9	210,6	6	35,1	1
Extracción de características	7,7	9	9	207,9	6	34,7	1
Clasificación características	7,0	9	8	168,0	7	24,0	1
Gestión del control	6,2	6	8	99,2	6	16,5	2
Gestión de la comunicación	5,5	6	5	55,0	8	6,8	4
Gestión de alarmas	7,3	5	5	60,83	5	12,2	3

Implementación genérica requerida	- Procesamiento de imágenes - Extracción de características - Clasificación de características
Implementación genérica recomendable	- Gestión del procesamiento visual - Gestión del control
Interfaz requerida	- Operativa básica - Gestión de alarmas
Interfaz recomendable	- Gestión de informes - Gestión de bases de datos - Gestión de la comunicación
Dominios excluidos	- Virtualización

Tabla 10. Resultados de la valoración y selección de dominios.

Como se muestra en la tabla anterior los dominios mejor valorados son los correspondientes al procesamiento de la información visual y a la gestión del control (categorías 1 y 2), seguidos de los dominios relacionados con la operativa básica y la gestión de alarmas (categoría 3). Para los primeros se deberá proporcionar una implementación genérica que permita su reutilización a la hora de construir nuevos VIPS; para los segundos bastará con proporcionar una interfaz en la que se recoja la funcionalidad que ofrece cada uno de ellos de modo que, posteriormente, ésta pueda implementarse de manera específica para cada nuevo VIPS. En ambos casos, tanto las implementaciones genéricas de los primeros como las interfaces definidas para los segundos, pasarán a formar parte del núcleo común de recursos reutilizables de la línea de productos.

Respecto a los dominios relacionados con la gestión de los informes, las bases de datos y las comunicaciones (categoría 4), se ha optado por no incluirlos en esta primera versión de la línea de productos. La razón principal que justifica esta decisión es que, al menos en el corto y medio plazo, no está previsto desarrollar aplicaciones que incorporen esta funcionalidad como parte esencial de su lógica (como es el caso de los SBIO, los SIMED y los SIG). Sin embargo, en versiones posteriores de la línea LPS-VIPS no se descarta incorporar estos dominios mediante la definición de las

correspondientes interfaces o incluso realizando una implementación genérica de los mismos si una reevaluación del criterio de selección así lo aconsejase.

Con esto concluye la etapa de definición del alcance de la línea LPS-VIPS correspondiente a la etapa Eco de la metodología PuLSE™. A continuación, en el apartado siguiente, se muestran los resultados de la etapa PuLSE™-CDA en la que se lleva a cabo el análisis del dominio de la línea LPS-VIPS.

4.3. Análisis del dominio (PuLSE™-CDA)

Como ya se comentó al comienzo de este capítulo al presentar el esquema del proceso que se iba a seguir para definir la línea LPS-VIPS, la etapa correspondiente al análisis del dominio (segunda del proceso) consta a su vez de tres etapas: (1) modelado de los Casos de Uso (CU) identificados en los dominios seleccionados durante la fase PuLSE™-Eco, (2) modelado de la variabilidad y (3) modelado de la toma de decisiones. En los tres apartados siguientes se comenta cómo se ha llevado a cabo cada una de estas etapas así como los resultados que se han obtenido.

4.3.1. Modelado de Casos de Uso (CU)

El modelado de Casos de Uso (CU) permite capturar, de manera sencilla y efectiva, los requisitos funcionales de un sistema visto desde la perspectiva de los actores externos con los que éste interactúa (usuarios u otros sistemas) [Jacobson 92]. Así, un modelo de CU recoge todos los posibles usos que se le pueden dar a un sistema o, dicho de otro modo, cada CU recoge una de las funcionalidades que ofrece el sistema a quien lo usa.

La especificación de CU suele realizarse de manera textual utilizando algún tipo de plantilla documental en la que típicamente se recoge, entre otra información, las pre- y post-condiciones asociadas al CU, la secuencia de eventos y acciones que se suceden en una ejecución normal del mismo, las posibles alternativas a esta secuencia en caso de error, etc. Dado que estas descripciones pueden resultar largas y farragosas de leer, sobre todo en sistemas de una cierta envergadura con muchos CU, suele resultar difícil extraer de ellas una visión global de la funcionalidad del sistema. Por ello, resulta común acompañar estas especificaciones de uno o más diagramas de CU elaborados utilizando alguna notación gráfica como, por ejemplo, UML™ [UML] (Unified Modeling Language™). En estos diagramas se representan los CU identificados así como las relaciones que éstos establecen entre sí y con los actores externos del sistema, lo que proporciona una vista mucho más esquemática y fácil de interpretar a simple vista que las correspondientes descripciones textuales.

El modelado de CU se ha aplicado con éxito en el desarrollo de numerosos aplicaciones concebidas como sistemas individuales. Sin embargo, ni las plantillas textuales ni los diagramas de CU empleados para especificar la funcionalidad de estos sistemas permiten capturar ningún tipo de variabilidad. Por lo tanto, para poder modelar líneas de productos software mediante CU resulta necesario extender las herramientas de especificación existentes de modo que permitan capturar la variabilidad funcional inherente a todas ellas.

En el cuadro de la Figura 29 se muestra una plantilla típica de especificación textual de CU a la que se han añadido ciertas extensiones (señaladas en cursiva) que permiten modelar los posibles aspectos variantes en un CU. Esta plantilla se ha extraído del caso de estudio presentado por de D. Muthig [Muthig 04] y será la que se utilice para describir textualmente los CU de la línea LPS-VIPS.

P1 - Nombre: nombre del CU. La utilización del estereotipo *<<VARIANT>>* antes del nombre del CU indicará que éste puede aparecer o no en los productos de la línea (equivalente a añadir una alternativa nula (∅) a cada uno de los pasos de la secuencia de éxito y a sus alternativas,).
P2 - Actor principal: Actor que desencadena la ejecución del CU.
P3 - Dominio: área funcional a la que pertenece el CU.
P4 - Nivel: Vista desde la que se describe el CU (global, usuario o subsistema).
P5 - Precondiciones: Estado del sistema y condiciones que deben cumplirse antes de iniciarse la ejecución del CU.
P6 - Postcondición en un escenario de éxito: Estado en el que quedará el sistema una vez completado el CU si no se produce ningún fallo.
P7 - Principal secuencia de éxito: Se describen las etapas de una ejecución típica del CU en la que no ocurre ningún error. Aquellas etapas que se ejecuten sólo de manera opcional se etiquetarán con el estereotipo *<<OPT>>*; aquellas en las que existan distintas alternativas de ejecución se etiquetarán con el estereotipo *<< ALT i >>*.
P8 - Extensiones: Situaciones de fallo que pueden ocurrir durante la ejecución de cada una de las etapas de la secuencia de éxito.
P9 - Requisitos no funcionales asociados al caso de uso.

Figura 29. Plantilla para la descripción de CU extendida con variabilidad.

En cuanto a la notación gráfica seleccionada para elaborar los diagramas de CU de la línea LPS-VIPS, se ha optado por utilizar UML [UML] dado que actualmente es considera el estándar *de facto* para la especificación del software orientado a objetos. Extender esta notación para que sea posible representar la variabilidad de los distintos elementos de un diagrama de CU resulta tan sencillo como añadir a su definición el estereotipo *<<variant>>* tal y como se muestra en el ejemplo de la Figura 30. Como en el caso anterior esta extensión también se ha extraído de la propuesta en el caso de estudio presentado por D. Muthig en [Muthig 04].

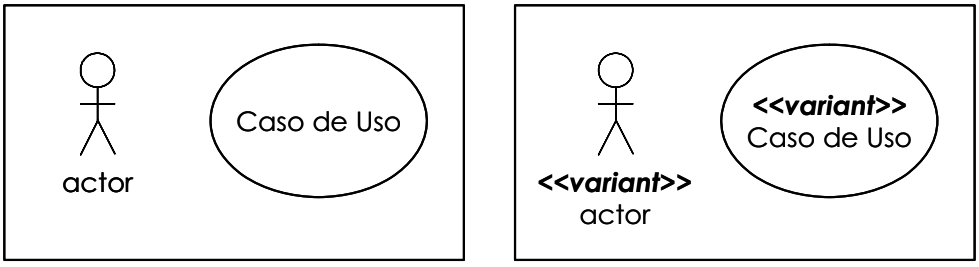


Figura 30. Extensión de UML para incluir variabilidad en actores y CU.

Una vez introducidas las modificaciones necesarias en las herramientas de especificación para poder expresar variabilidad en los CU, se está en disposición de realizar el análisis de los dominios seleccionados para la línea de productos LPS-VIPS. Para ello, en primer lugar se presentará el diagrama de CU elaborado utilizando la notación UML extendida (ver Figura 31) para, a continuación, describir cada uno de los CU del diagrama utilizando la plantilla definida previamente en la Figura 29.

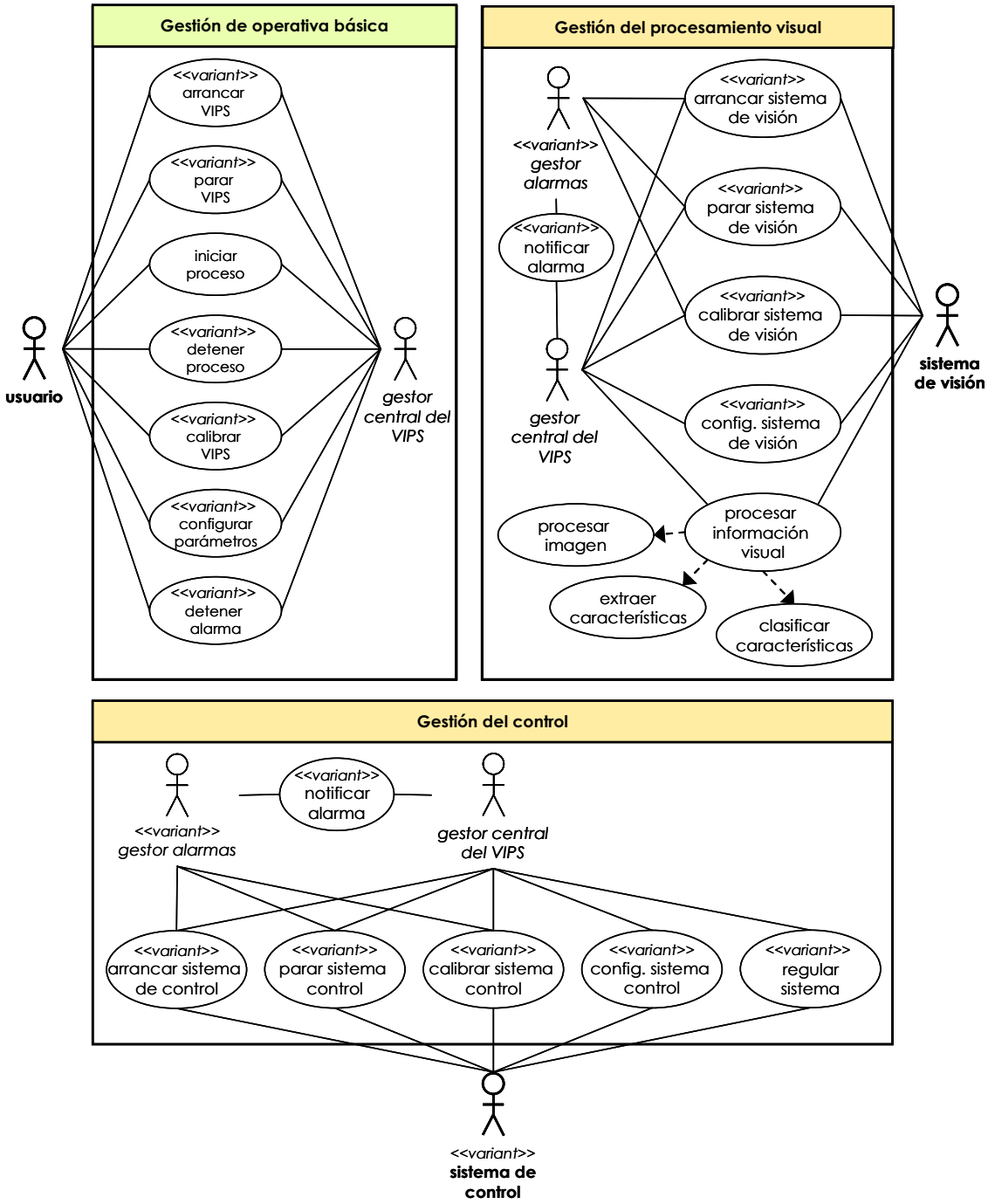


Figura 31. Diagramas de CU de los dominios seleccionados.

Como se observa en los diagramas de CU de la figura anterior, la mayoría de los CU identificados a partir de las descripciones de los dominios se han estereotipado como variantes ya que algunos VIPS no incluyen dicha funcionalidad. Por el mismo

motivo también se ha incluido el estereotipo <<*variant*>> en el sistema de control (actor externo), ya que algunos VIPS (por ejemplo, los SIG y la mayoría de los SIMED e IPERC) no interactúan con este tipo de sistemas (ver mapa de productos de la Tabla 7). Nótese que el hecho de que un CU no se esté etiquetado como <<variant>> no significa que su descripción no contenga uno o más elementos variantes sino simplemente que su funcionalidad aparece invariablemente en todos los VIPS.

Uno de los aspectos que puede llamar más la atención en los diagramas de CU anteriores es el hecho de haber representado el *gestor central del VIPS* y el *gestor de alarmas* como actores dentro de cada uno de los dominios. Sin embargo, considerando estos dos dominios como subsistemas del VIPS, ambos pueden verse como entidades externas que interactúan con el resto de los dominios y por lo tanto cobra sentido el hecho de representarlos como actores externos para ellos, aunque formen parte de la especificación interna del VIPS.

En el diagrama de CU correspondiente al dominio de gestión del procesamiento visual, cabe destacar también la especial relación existente entre el CU procesar información visual y los CU asociados a los subdominios internos: *procesar imagen*, *extraer características* y *clasificar características*. Estas relaciones, representadas mediante flechas de trazos discontinuos en el diagrama[25], indican que el CU en el que se origina la flecha incorpora toda la funcionalidad ofrecida por el CU destino; de hecho, los CU destino pueden considerarse subfunciones que se invocan desde el CU origen.

Una vez comentados los diagramas anteriores, a continuación se presentan las descripciones textuales de los CU identificados, utilizando para ello la plantilla extendida previamente definida en la Figura 29. Para evitar alargar en exceso este capítulo y facilitar así su lectura, en este apartado sólo se recoge, a modo de ejemplo, la descripción de los CU *arrancar SIVA* e *iniciar proceso*, ambos correspondiente al dominio de gestión de la operativa básica. La descripción del resto de CU puede encontrarse al final de la memoria en el Apéndice II.

► *Caso de Uso: arrancar VIPS (Gestión de operativa básica)*

P1 - Nombre: <<*VARIANT*>> arrancar VIPS
Sólo los SIVA, los SNAV y ocasionalmente los SIMED ofrecen al usuario (de manera explícita) esta funcionalidad.

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones:

- El sistema muestra activada la opción "arrancar VIPS" en el menú de gestión de operativa básica.
- Los dispositivos externos del VIPS están correctamente conectados.

[25] Generalmente este tipo de relaciones se representan utilizando una flecha de trazo continuo etiquetada con el estereotipo <<extends>>. En este caso, por motivos de espacio en el dibujo se ha sustituido esta representación por una flecha de trazo discontinuo.

P6 - Postcondición en un escenario de éxito: El sistema muestra activada la opción "parar VIPS" en el menú de gestión de operativa básica y todos los subsistemas y dispositivos externos están arrancados y funcionando.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "arrancar VIPS" del menú de gestión de operativa básica.
2. El sistema carga los parámetros de configuración del arranque y los envía al *gestor central del VIPS* mediante un mensaje solicitándole la inicialización de los distintos subsistemas y dispositivos externos.
3. El sistema espera la notificación del gestor central del VIPS indicándole que las operaciones de arranque e inicialización se han completado con éxito.
4. El sistema desactiva la opción "arrancar VIPS" y activa la opción "parar VIPS" en el menú de gestión de operativa básica.
5. El sistema notifica al usuario que la operación ha concluido con éxito mediante una ventana de mensaje.

P8 - Extensiones:

3a) Se recibe una notificación de alarma indicando que durante el arranque se ha producido un fallo en alguno de los dispositivos externos que se trataban de inicializar. El sistema notifica la alarma al usuario en una ventana de mensajes.

P9 - Requisitos no funcionales:

***** Una vez que el usuario ha activado la opción "arrancar VIPS" se le deberá notificar el éxito o el fallo en la operación en un tiempo limitado (depende de los dispositivos externos que tenga el VIPS).

► ***Caso de Uso: iniciar proceso (Gestión de operativa básica)***

P1 - Nombre: iniciar proceso
Todos los VIPS ofrecen esta funcionalidad.

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones: El sistema está arrancado y funcionando (no hay pendiente ninguna alarma).

P6 - Postcondición en un escenario de éxito: El sistema está arrancado y funcionando (no hay alarmas pendientes). Al menos se ha realizado una operación de procesamiento de información visual.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "iniciar proceso" del menú de gestión de operativa básica. Una vez seleccionada esta opción deja de estar activa en el menú.
2. En las opciones de configuración del VIPS ¿está activada la opción de mostrar los resultados del procesamiento de la información visual?
   <OPT> El gestor de operativa básica crea una ventana de resultados con el formato indicado en las opciones de configuración, esto es, con los

elementos gráficos necesarios para visualizar toda la información que se genera durante el proceso.

3. El sistema envía un mensaje al *gestor central del VIPS* solicitándole que inicie el procesamiento de la información visual. Si se creó una ventana para visualizar los resultados, en la solicitud se adjuntará una referencia a la misma (se difiere al *gestor central del VIPS* la tarea de actualizarla con los resultados).
4. ¿Qué tipo de información visual se va a procesar?
   <ALT 1> Una imagen individual (SBIO, SIMED y SIG). En este caso el sistema esperará a que el gestor central del VIPS le notifique que la operación ha concluido con éxito y volverá a activar la opción de "iniciar proceso" en el menú.
   <ALT 2> Una secuencia de vídeo almacenada en un fichero (SIG, SBIO) o un flujo de vídeo en vivo que se procesará durante un intervalo de tiempo prefijado de antemano (SIMED) o de manera indefinida (SIVA, SNAV, IPERC). En este caso el sistema activará la opción "detener proceso" y esperará a que, o bien el usuario seleccione esta opción, o bien el gestor central del VIPS le informe de que la operación ha concluido con éxito. Cuando ocurra alguna de estas dos situaciones, se volverá a activar la opción de "iniciar proceso" en el menú.

P8 - Extensiones:

4a) Se recibe una notificación de alarma indicando que durante el proceso se ha producido algún error ocasionado por alguna de las operaciones realizadas por los subsistemas o dispositivos implicados en el proceso.

P9 - Requisitos no funcionales:

* En muchos VIPS, la velocidad con la que se procesa la información visual es un factor crítico (por ejemplo en los SIVA y en los SNAV); en algunos de estos VIPS existen fuertes restricciones de tiempo real.
* En ciertos casos resulta imprescindible asegurar la robustez y fiabilidad de este proceso. Por ejemplo, en el caso de los SNAV, éstos deben ser capaces de reaccionar de forma segura aún cuando las condiciones de visibilidad sean pésimas.
* En ocasiones la precisión también resulta esencial, por ejemplo en algunos SIMED.

4.3.2. Modelado de la variabilidad

Durante esta fase, para cada uno de los CU descritos en el apartado anterior en los que se haya detectado algún elemento variante, se elaborará un diagrama de variabilidad utilizando la notación FODA (Feature-Oriented Domain Analisys) [Kang 90]. En estos diagramas se representarán los actores, las acciones y los eventos de la secuencia de éxito y de la secuencia extendida etiquetados como alternativos u optativos. A continuación se muestran los diagramas de variabilidad correspondiente a los CU *arrancar SIVA* e *iniciar proceso* (ver Figura 32), ambos descritos en el apartado anterior. Los diagramas correspondientes al resto de CU pueden encontrarse al final de la memoria en el Apéndice II junto con sus respectivas descripciones.

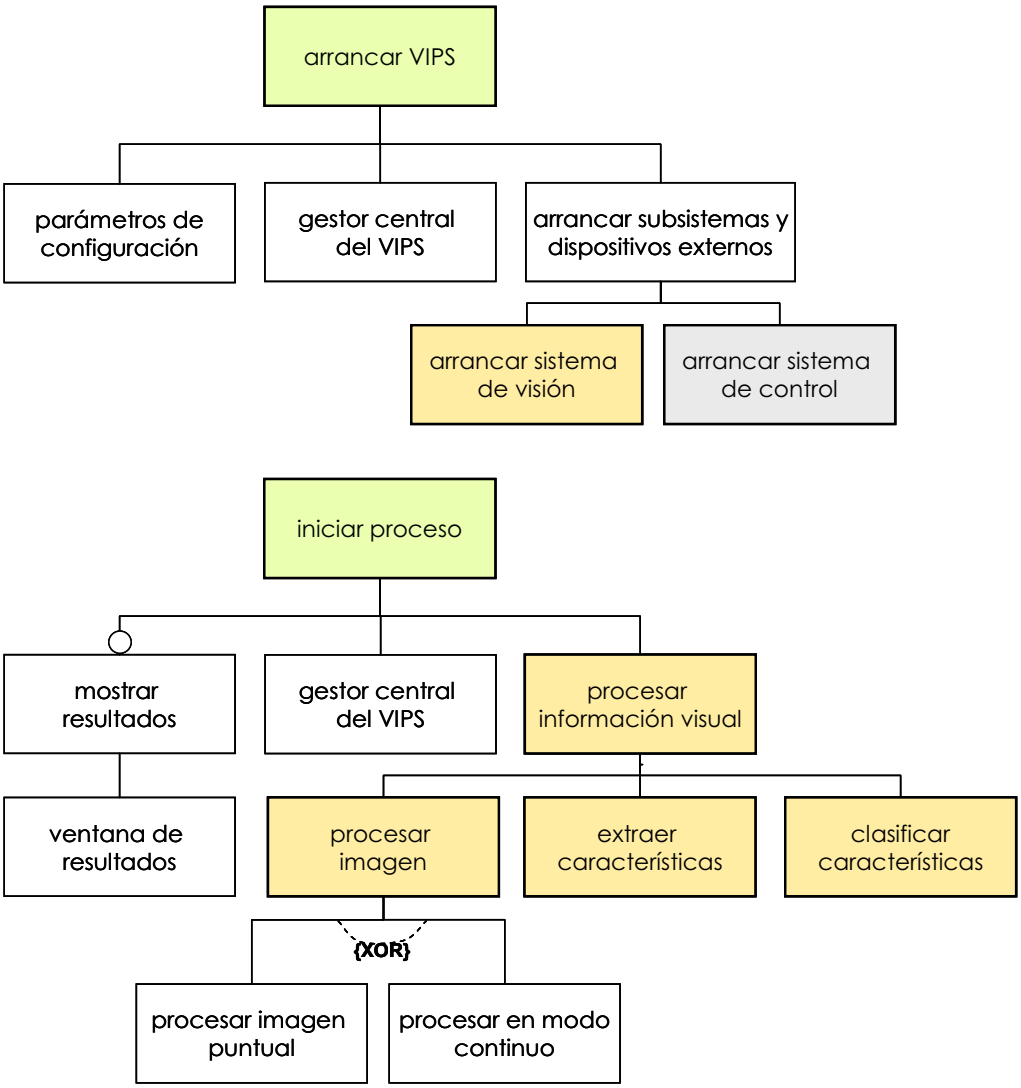


Figura 32. Diagramas de variabilidad de los CU *arrancar VIPS* e *iniciar proceso*.

Como se observa en la figura anterior, el uso de la notación FODA [Kang 90] permite indicar qué elementos del CU son optativos (aquellos en los que se dibuja un circulo en la parte superior) y cuáles son alternativos (todas las alternativa se recogen dentro de un semicírculo punteado etiquetado con la palabra XOR).

4.3.3. Modelado de la toma de decisiones

Para poder instanciar los modelos de variabilidad anteriores resulta necesario contar con los correspondientes modelos de decisión. Estos modelos recogen, en forma de tabla, las distintas configuraciones posibles para todos los elementos variantes indicando, en cada caso, qué implicaciones tiene en el modelo de análisis el hecho de que un determinado producto incluya (o no) un elemento optativo o, en el caso de los elementos alternativos, qué conlleva la elección de cada una de las opciones disponibles. A continuación se muestra la tabla de decisión correspondientes a los elementos variantes identificados en el CU *iniciar proceso* (ver Tabla 11).

ID	Pregunta	Elemento variante	Resolución	Efecto
1	¿Se oferta la opción "mostrar resultados"?	Opción de menú "mostrar resultados"	Sí	Paso 2 del CU *iniciar proceso* es obligatorio
			No	Eliminar paso 2 del CU *iniciar proceso*
2	¿Qué tipo de procesamiento se realiza?	iniciar procesamiento visual	Puntual	Eliminar <ALT 2> en el paso 4 del CU *iniciar proceso*
			Continuo	Eliminar <ALT 1> en el paso 4 del CU *iniciar proceso*

Tabla 11. Modelo de decisiones asociado al CU *iniciar proceso*.

El CU *arrancar SIVA* no tiene asociada ninguna tabla de decisión ya que, como se observa en su descripción y en el correspondiente diagrama de variabilidad, éste no contiene ninguna característica optativa o alternativa que configurar. Ahora bien, volviendo a su descripción, puede observarse que este CU está estereotipado como <<VARIANT>>. Esto significa que este CU modela una funcionalidad que es opcional en algunos de los VIPS que se derivan de la línea LPS-VIPS. Dado que este tipo de variabilidad no puede quedar recogida en una tabla de decisión asociada a un CU concreto, deberá elaborarse un modelo de decisión global (a nivel de producto) que recoja las posibles opciones/alternativas funcionales de los distintos productos. Esta tabla de decisión, junto con las elaboradas para el resto de CU variantes, puede encontrarse al final de la memoria en el Apéndice II.

A continuación, y partiendo de los resultados obtenidos durante esta fase de modelado del dominio (PuLSE™-CDA), en el apartado se siguiente se definirá la arquitectura software genérica de los VIPS utilizando para ello la metodología KobrA (ver Apéndice I, apartado I.2).

4.4. Arquitectura genérica de la línea LPS-VIPS (KobrA)

Para definir la arquitectura genérica de los VIPS, en lugar de seguir la metodología PuLSE™ como hasta ahora se venía haciendo, se utilizará la metodología KobrA [KOBRA]. Como se explica en el Anexo I de esta memoria, KobrA es una particularización orientada a objetos de PuLSE™ lo que facilita la integración de ambas en este punto.

La definición de la arquitectura software genérica de una línea de productos siguiendo la metodología Kobra, requiere elaborar tres modelos con sus correspondientes diagramas:

- El primero de ellos, denominado diagrama de entidades, muestra el conjunto de actores internos (subsistemas) y externos identificados durante la fase previa de análisis.
- El segundo, denominado diagrama de procesos, recoge las distintas actividades identificadas en las secuencias de éxito de los CU.
- Por último, el denominado diagrama del árbol de Komponentes[26], recoge el aspecto estructural de la arquitectura genérica de la línea LPS-VIPS. Este diagrama se completa con los obtenidos durante el proceso de especificación y definición de los distintos Komponentes.

Estos tres diagramas: el de entidades, el de procesos y el del árbol de komponentes, se muestran a continuación en las figuras 33, 34 y 35 respectivamente. Los diagramas de especificación y definición de los Komponentes pueden encontrarse en el Apéndice II al final de esta memoria.

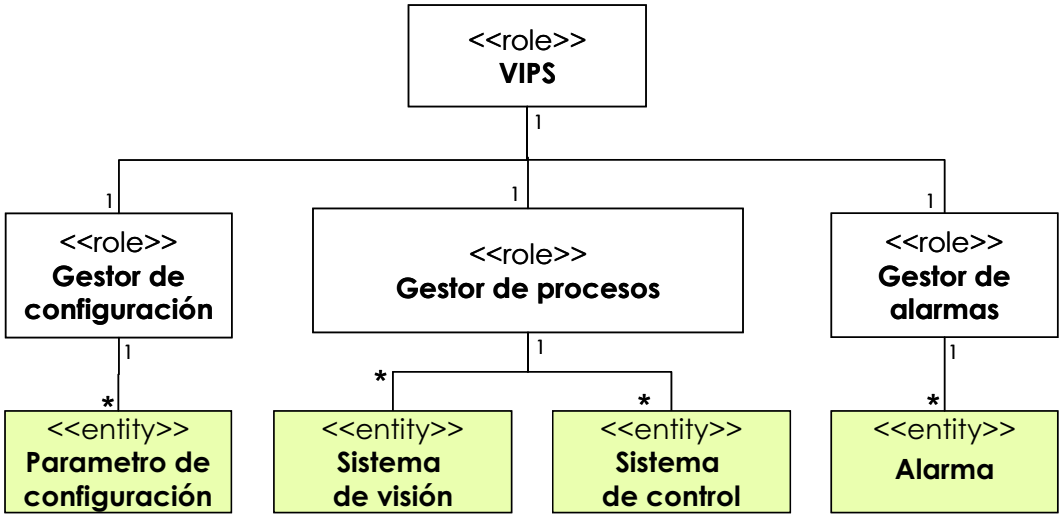


Figura 33. Diagrama de entidades.

Como se muestra en la figura anterior, el diagrama de entidades recoge dos tipos de elementos: las entidades externas con los que interactúa el VIPS (coloreadas en verde y etiquetadas con el estereotipo <<entity>>), y las entidades internas (subsistemas) que lo componen (etiquetadas con el estereotipo <<role>>). En el primer caso no debe confundirse entidad externa (fuente o destino de la información que maneja el sistema) con actor o sistema externo. De hecho, algunos actores externos (como por ejemplo en este caso el usuario) no aparecen en el diagrama como entidades externas y, viceversa, no todas las entidades externas tienen por qué ser actores o sistemas externos (como ocurre en este caso con las alarmas o los parámetros de configuración).

[26] Con el término komponente se hace referencia a un componente KobrA.

A continuación, en la Figura 34, se muestra el diagrama de procesos en el que se recogen las distintas actividades identificadas en las secuencias de éxito de los distintos CU.

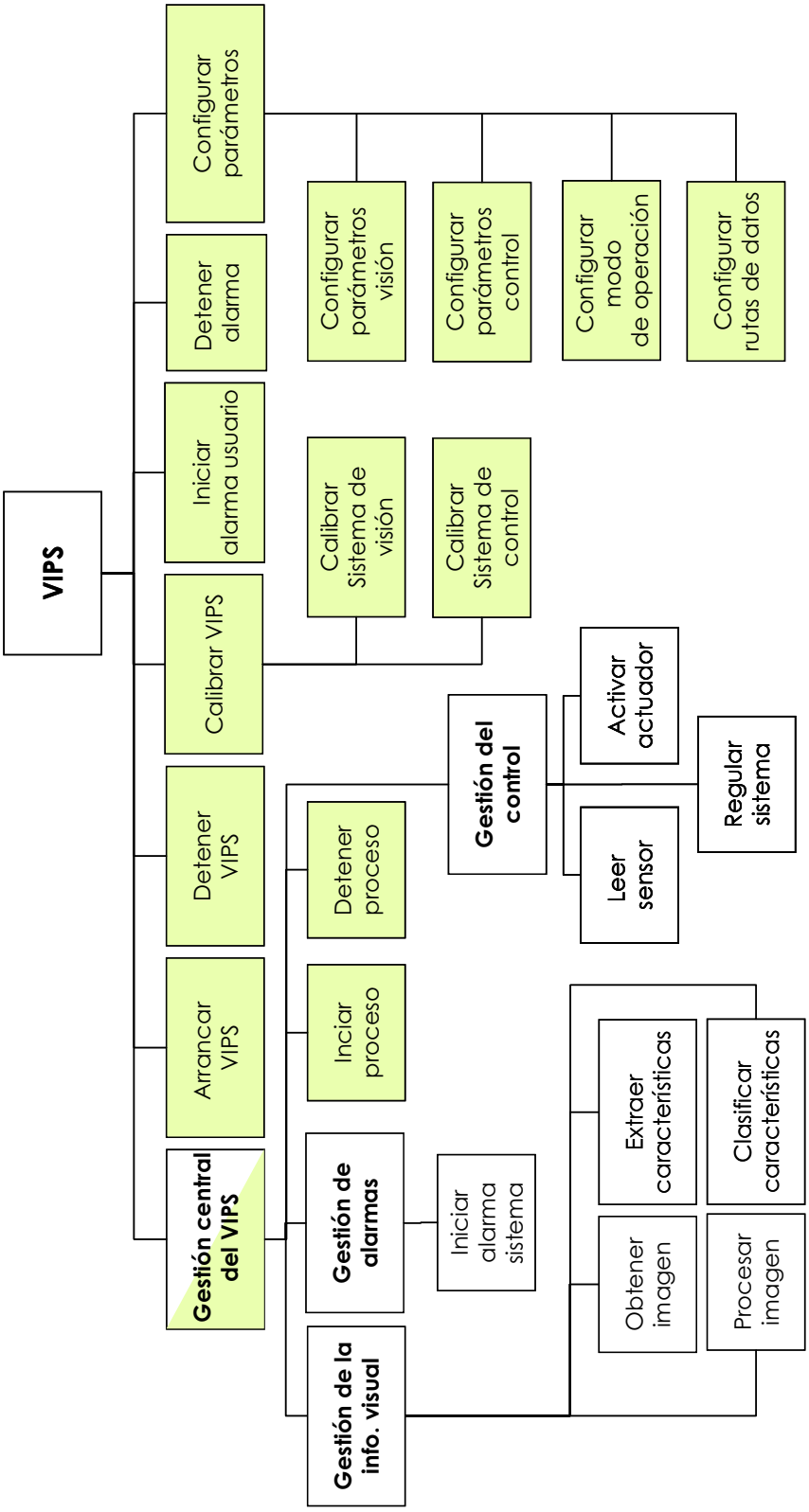


Figura 34. Diagramas de Procesos.

El diagrama de la Figura 34 muestra los procesos que se llevan a cabo en un VIPS, bien como consecuencia de una orden directa del usuario (por ejemplo, *calibrar VIPS*), o bien porque los desencadena algún otro proceso (por ejemplo, *iniciar alarma sistema*). Los primeros (coloreados en verde) muestran los procesos que puede lanzar el usuario probablemente a través de la selección de una opción en el menú de la aplicación, mientras que entre los segundos (de fondo blanco) aparecen algunos etiquetados en negrita y otros no; los primeros se corresponden con procesos complejos que invocan a los segundos (más simples), a veces sólo una vez y otras de manera repetida utilizando, por ejemplo, una estructura de control de tipo bucle.

Nótese que el proceso de Gestión Central del VIPS se ha coloreado parcialmente en verde; esto es debido a que sólo parte de los procesos que éste incluye puede lanzarlos directamente el usuario (iniciar y detener proceso), mientras que el resto son desencadenados por el propio sistema (por ejemplo, el proceso de Gestión de la Información Visual se lanza automáticamente cuando el usuario ejecuta la opción de iniciar proceso).

Por último, en la Figura 35, se muestra el diagrama que recoge el árbol de Komponentes que conforman la arquitectura genérica de la línea LPS-VIPS.

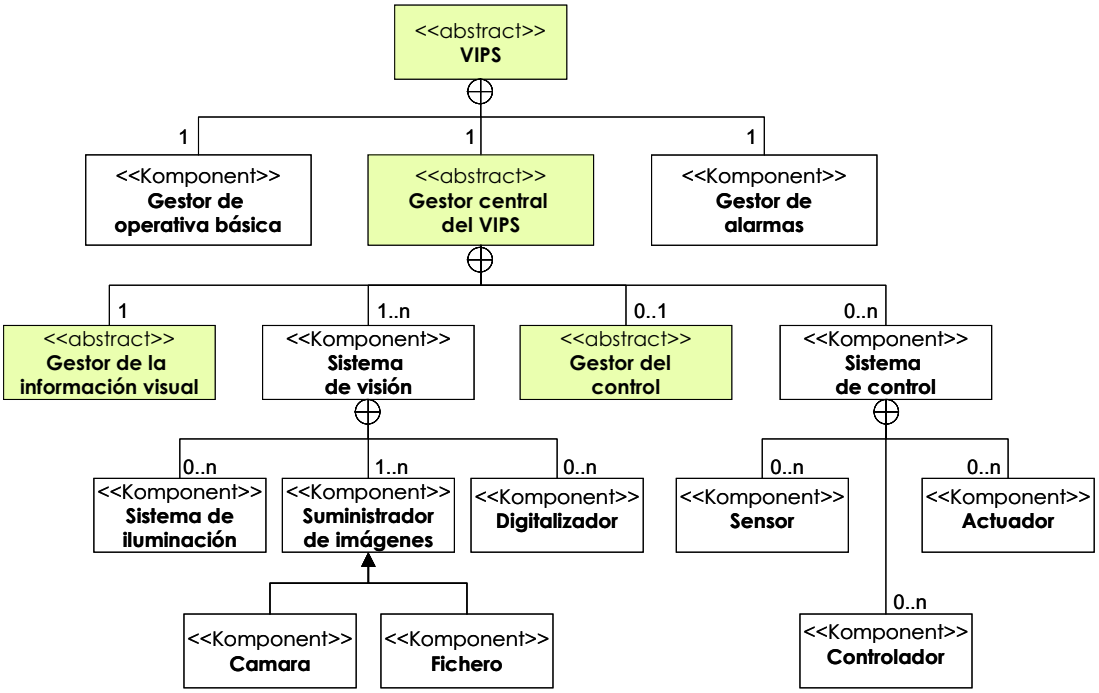


Figura 35. Diagrama del árbol de Komponentes.

Como se observa en el diagrama de la figura anterior, los distintos Komponentes se relacionan entre sí de manera jerárquica mediante relaciones de composición o de herencia. En las primeras (marcadas con un círculo con una cruz en su interior), varios Komponentes se agrupan para formar un Komponente compuesto, mientras que en las segundas (marcadas con una flecha) ciertos Komponentes especializan a otros definiéndolos con un mayor nivel de detalle.

Como también se observa en la Figura 35 algunos de los Komponentes del árbol se han estereotipado como <<Komponent>> mientras que otros aparecen como <<abstract>> (coloreados en verde). Los primeros se corresponden con aquellos elementos que son directamente instanciables a partir de los requisitos específicos del VIPS que se desee construir. Por ejemplo, en el momento que el cliente especifique qué funcionalidad externa debe proporcionar el VIPS, se podrán instanciar los modelos de decisión asociados a la gestión de la operativa básica pudiendo implementarse, de manera automática, el menú de la aplicación así como la invocación a los distintos procesos que lanzará el usuario al seleccionar cada una de sus opciones.

Por el contrario, los komponentes mostrado sobre fondo verde y etiquetados como <<abstract>> se corresponden con elementos que no pueden instanciarse (total o parcialmente) de manera automática a partir de los requisitos específicos de un determinado VIPS. Por ejemplo, el Komponente *Gestor de la Información Visual* no se puede instanciar automáticamente, ni siquiera de forma parcial, ya que su implementación requiere seleccionar una combinación de algoritmos distinta para cada VIPS (selección a medida). Del mismo modo, los Komponentes *VIPS* y *Gestor Central del VIPS* tampoco pueden instanciarse ya que al menos uno de sus Komponentes internos (nodos hijo en el árbol) son a su vez abstractos.

Con la especificación de la arquitectura genérica concluye la definición de la línea LPS-VIPS. En el apartado siguiente se describe brevemente cómo instanciar esta definición para crear nuevos VIPS.

4.5. ¿Cómo crear nuevos VIPS a partir de la definición de la línea LPS-VIPS?

El proceso de construcción de nuevas aplicaciones a partir de una línea de productos requiere instanciar todos y cada uno de los modelos generados durante su definición. Para ello, partiendo de los requisitos del producto que se desea construir, se irán reduciendo los puntos de variabilidad tanto del modelo de análisis como de la arquitectura software utilizando las guías que proporcionan los distintos modelos de decisión.

Como resultado de este proceso de instanciación se obtendrá un nuevo VIPS aún abstracto (ver Figura 35), esto es, un esquema o plantilla de la aplicación final que estará parcialmente rellena con aquella funcionalidad que se haya podido derivar a partir de los requisitos del producto (por ejemplo, la relativa la gestión de la operativa básica). El resto de la plantilla, esencialmente la relativa al procesamiento de la información visual y a la gestión del control, deberán rellenarse de manera específica para cada VIPS concreto.

Para facilitar el relleno de la parte de la plantilla correspondiente al procesamiento de la información visual se ha desarrollado un conjunto de componentes software que permiten integrar la funcionalidad ofrecida por varias de las herramientas y librerías de procesamiento de imágenes existentes en la actualidad. La creación de esta librería ha sido posible gracias a la herramienta de soporte IP-CoDER que se presenta en el capítulo siguiente junto con uno de los VIPS desarrollados para validar tanto la nueva metodología de desarrollo de VIPS propuesta como parte de este trabajo como la propia herramienta IP-CoDER.

Capítulo 5

IP-CoDER: herramienta para desarrollo de prototipos ejecutables de VIPS a partir de componentes software

En este capítulo se presenta la herramienta IP-CoDER implementada para dar soporte a la nueva metodología de desarrollo de VIPS propuesta en el capítulo 3. El diseño de esta herramienta está basado en la definición de un nuevo modelo de componentes software que permite integrar y compatibilizar, de manera eficiente, la funcionalidad ofrecida por varias de las librerías de procesamiento de imágenes existentes en el mercado. IP-CoDER implementa este modelo añadiendo a los componentes una dimensión gráfica que hace de ésta una herramienta de programación visual. Gracias a ella, el usuario puede diseñar nuevos VIPS simplemente seleccionando y conectando entre sí distintos componentes previamente creados y almacenados en una librería. A partir de estos diseños IP-CoDER es capaz de generar, de manera totalmente automática, un prototipo ejecutable que permite al usuario comprobar el funcionamiento del sistema utilizando distintos datos de entrada. Con el fin de validar esta herramienta y de demostrar su utilidad se han desarrollado varios casos de estudio, uno de los cuales se presenta al final de este capítulo.

5.1. Introducción

En la actualidad es posible encontrar en el mercado una amplia variedad de herramientas que resultan de utilidad para el desarrollo de Sistemas de Procesamiento de Información Visual, tal y como ha quedado recogido previamente en el Capítulo 2. Sin embargo, como ya se apuntó entonces, ninguna de estas herramientas proporciona un soporte integral a todo el ciclo de vida de estos productos que, por otra parte, suelen concebirse en torno a una determinada plataforma hardware seleccionada en primera instancia y a la que se supedita el desarrollo posterior del componente software.

Dado que como aportación fundamental de esta Tesis se ha propuesto una nueva metodología para el desarrollo de VIPS (ver Capítulo 3), más centrada en el software y que se fundamenta en la definición de la línea de productos LPS-VIPS descrita en el capítulo anterior, resulta necesario encontrar (o en su defecto construir) una herramienta que dé soporte al proceso de creación de nuevos VIPS siguiendo este nuevo enfoque. Con este objetivo surge la idea de implementar IP-CoDER como una herramienta que permita al usuario llevar a cabo las dos tareas siguientes:

- Crear nuevas instancias de la línea LPS-VIPS mediante la configuración de los distintos modelos de decisión a partir de los requisitos del sistema que se desee construir. Para ello, IP-CoDER proporcionará al usuario las opciones y alternativas recogidas en las tablas de decisión y construirá el esqueleto del nuevo VIPS (aplicación implementada sólo parcialmente) de acuerdo a las decisiones que éste adopte.
- Completar la implementación del esqueleto obtenido como resultado del proceso anterior y, en particular, la correspondiente al procesamiento de la información visual. Para ello el usuario deberá: (1) seleccionar los algoritmos de procesamiento de imágenes, extracción de características y clasificación de patrones que considere más adecuados, (2) implementar dichos algoritmos seleccionando la secuencia en la que deben aplicarse, (3) validar la corrección del diseño implementado y ajustarlo si fuera necesario, y (4) añadir el código final obtenido al esqueleto del VIPS para completarlo.

La implementación de la primera de estas dos tareas (instanciación de la línea LPS-VIPS) justifica ya de por sí la necesidad de crear una nueva herramienta como IP-CoDER por una razón fundamental: en la actualidad, aún son pocas las herramientas que permiten definir e instanciar líneas de productos software y, las que existen, suelen generar esqueletos de aplicaciones poco eficientes y difíciles de completar a posteriori, más aún teniendo en cuenta la naturaleza mixta (Hw/Sw) de los productos que se pretenden derivar de la línea LPS-VIPS (las herramienta existentes están pensadas para líneas de productos exclusivamente software).

En el caso de la segunda tarea (relleno de la parte del esqueleto correspondiente al procesamiento de la información visual) debe decidirse en qué medida resulta conveniente que la herramienta IP-CoDER de soporte a cada una de las cuatro etapas antes señaladas. En este sentido es posible optar por alguna de las dos alternativas siguientes:

- El usuario utiliza alguna de las numerosas herramientas disponibles en el mercado para implementar el código correspondiente al procesamiento de la información visual. Una vez obtenido este código IP-CoDER únicamente se encarga de incorporarlo al esqueleto del VIPS.
- IP-CoDER ofrece al usuario, como parte de su funcionalidad, la posibilidad de implementar el código correspondiente al procesamiento de la información visual. Para ello deberá optarse por una de los dos alternativas siguientes:
  - Implementar esta funcionalidad como parte de IP-CoDER, esto es, crear una nueva biblioteca de funciones de procesamiento de imágenes, extracción de características, etc.
  - Incorporar la funcionalidad ofrecida por alguna de las librerías ya existentes [OPENCV] [IPL] [MIL].

Obviamente, construir una herramienta que proporcione un soporte integral al proceso de implementación de los nuevos VIPS permitiendo, además de la obtención del esqueleto (instanciación de la línea LPS-VIPS), la implementación del código que permite rellenarlo, resulta mucho más costoso que si se delega parte del trabajo en otras herramientas. Sin embargo, esta solución no sólo beneficia al usuario (le ahorra tener que manejar distintas herramientas) sino que además evita posibles problemas de integración del código correspondiente al procesamiento de la información visual en el esqueleto del VIPS.

En cuanto a si IP-CoDER debería implementar internamente o importar de alguna de las numerosas librerías existentes en el mercado la funcionalidad relativa al procesamiento de la información visual, sin duda la segunda opción es la más económica en tiempo y la más eficiente y fiable (las librerías existentes están optimizadas y probadas y suelen tener un buen soporte documental). Ahora bien, ¿qué librería es la más adecuada? ¿Resulta posible/necesario/ conveniente incorporar la funcionalidad de varias de ellas simultáneamente?

Aunque muchas de las librerías disponibles ofrecen un núcleo funcional común, existen ciertas diferencias entre ellas que cabría explotar de ser posible utilizarlas de manera conjunta para el desarrollo de una misma aplicación. Sin embargo, la integración de estas herramientas no resulta en absoluto sencilla por varios motivos:

- En primer lugar, la mayoría de ellas definen sus propios tipos de datos y utilizan distintas convenciones de paso de parámetros para sus funciones.
- En segundo lugar, aunque muchas de ellas están implementadas utilizando C/C++ como lenguaje programación (por ejemplo, Intel© OpenCV [OPENCV] o Matrox© Imaging Library [MIL]), otras como las librerías de adquisición [MATLAB IAT] y procesamiento de imágenes [MATLAB IPT] de The Mathworks© utilizan lenguajes de programación propios [MATLAB].
- Por último, aunque algunas de estas herramientas están disponibles como productos de código abierto (caso de IPL [IPL] y OpenCV [OPENCV] de Intel©), otras muchas se venden a modo de COTS (caso de las [MIL] y de los toolboxes de The Mathworks© [MATLAB IAT] [MATLAB IPT]) por lo que puede resultar bastante complejo averiguar ciertos detalles internos de su implementación.

Para solventar estos inconvenientes y dado que se ha decidido implementar IP-CoDER como una herramienta de desarrollo integral de VIPS que incorpore la funcionalidad ofrecida por varias de las librerías de procesamiento de información visual actualmente disponibles, se ha optado por diseñar un nuevo modelo de componentes software capaces de albergar en su interior funciones heterogéneas (procedentes de varias de estas librerías) proporcionándoles un interfaz externo homogéneo que las hagas compatibles entre sí. En el siguiente apartado se comentan las características de este nuevo modelo de componentes.

5.2. Definición de un nuevo modelo de componentes software para procesamiento de información visual

Antes de comentar los detalles sobre cómo se ha diseñado e implementado el nuevo modelo de componentes software para procesamiento de información visual, en el siguiente apartado se justifica la necesidad del mismo debido a la inadecuación de los modelos de componentes comerciales actualmente disponibles en el mercado.

5.2.1. ¿Por qué un modelo de componentes propio?

El uso de plataformas y modelos de componentes comerciales ampliamente difundidos, como los comentados en el Capítulo 3 (CORBA, EJB. .NET, etc.), reporta una serie de ventajas obvias y nada desdeñables entre las que cabe destacar las siguientes:

- Desarrollados por organizaciones líderes en el desarrollo de software (Sun Microsystems, OMG, Microsoft, etc.).
- Existencia de herramientas que dan soporte a las distintas plataformas y modelos de componentes comerciales.
- Existencia de un mercado emergente de componentes de todo tipo, válidos para un amplio abanico de aplicaciones.
- Múltiples fuentes de documentación (libros, artículos, tutoriales, etc.).
- Experiencias descritas por otros desarrolladores que pueden servir de guía a la hora de abordar un nuevo proyecto.
- Suelen exhibir excelentes prestaciones en cuanto interoperabilidad, independencia de la plataforma, facilidad de uso etc.

Por otra parte, como ya se comentó en el Capítulo 2, el procesamiento de información visual resulta una tarea computacionalmente muy costosa dado el ingente volumen de información contenido en cada una de las imágenes que se deben procesar y la complejidad de muchos de los algoritmos empleados para ello. Así, el modelo de componentes que se seleccione para implementar aplicaciones de tipo VIPS debe ser, ante todo, eficiente de modo que añada la menor carga computacional posible al sistema.

Los modelos de componentes comerciales actualmente disponibles suelen ser considerablemente complejos ya que dan soporte a un sinfín de características

(distribución, acceso a bases de datos, etc.) que, no siendo necesarias para la mayoría de los VIPS, no hacen sino reducir su eficiencia y complicar su utilización. Por ello, se ha considerado más ventajoso desarrollar un nuevo modelo de componentes software que se ajuste a la medida a los requisitos propios de los VIPS y cuyas características se detallan a continuación en el siguiente apartado.

5.2.2. Características que debe ofrecer el modelo de componentes

A la hora de diseñar un nuevo modelo de componentes software se debe especificar tanto la estructura de los distintos tipos de componentes que se desea implementar como los mecanismos necesarios para su creación, conexionado, agrupación (en componentes más complejos), documentación, almacenamiento y recuperación de fichero, etc. A partir de este diseño, la implementación del modelo puede abordarse desde la óptica de varios de los paradigmas actuales de programación si bien, el orientado a objetos, es probablemente el que, de una forma más natural, soporta los conceptos relacionados con el desarrollo de software basado en componentes (DSBC). Por ello, el modelo de componentes para procesamiento de información visual que se propone como parte de este trabajo se ha estructurado en torno a un conjunto de clases, entre las que cabe desatacar las siguientes:

- Un conjunto de clases que modelan las estructuras de datos típicamente utilizadas en las aplicaciones de procesamiento de información visual (imágenes, máscaras, regiones de interés, etc.) y que aparecen, aunque con distinto formato, en las diversas librerías que se pretende integrar. Entre la funcionalidad que ofrecen estas clases cabe destacar la existencia de métodos de conversión de formato que permiten transformar los tipos manejados por cada una de las librerías en tipos propios del modelo de componentes y viceversa. Más adelante, en el apartado 5.2.3.1, quedan recogidos algunos de los detalles relativos al diseño e implementación de estas clases.
- Una clase *Component* que modela la estructura y la funcionalidad de los componentes atómicos. Cada componente atómico alberga en su interior una función procedente de alguna de las librerías preexistentes que se pretenden incorporar como parte de la herramienta IP-CoDER. Los únicos atributos externamente visibles de estos componentes son el número y tipo de sus conectores de entrada y salida, mientras que entre sus atributos privados cada componente almacena información relativa a la función encapsulada, la librería de la que procede, los ficheros que deben importarse para su ejecución, etc. En cuanto a la funcionalidad que ofrecen los componentes atómicos cabe destacar la existencia de métodos para conectar sus entradas y/o salidas a las de otros componentes, así como para generar automáticamente el código de envoltura (*wrapper*) asociado a la función que éste encapsula (código encargado de realizar las conversiones de tipos oportunas antes y después de invocar a la función interna utilizando la convención de paso de parámetros adecuada). Algunos detalles relativos al diseño y la implementación de la clase *Component* pueden encontrarse más adelante en el apartado 5.2.3.2.

- Una clase *CComponent* que modela la estructura y la funcionalidad de los componentes compuestos. Éstos albergan en su interior dos o más componentes (atómicos o a su vez compuestos) que deberán estar convenientemente conectados entre sí. Como se detalla más adelante, en el apartado 0, la clase *CComponent* se ha diseñado como una extensión y una agregación (ambas en el sentido de la Programación Orientada a Objetos) de la clase *Component* previamente definida. Así, un componente compuesto ofrece, además de la misma funcionalidad que uno atómico (métodos para conectarse a otros componentes, para generar su código asociado, para almacenarse en fichero, etc.), algunos métodos adicionales como, por ejemplo, el encargado de conectar entre sí dos de los componentes internos.

El hecho de haber diseñado los componentes del modelo como envolventes o *wrappers*, ya sea de funciones procedentes de las distintas librerías (componentes atómicos), o bien de un conjunto de componentes previamente definidos e interconectados entre sí (componentes compuestos), permite al usuario trabajar con ellos como si de cajas negras se tratara, de modo que puede abstraerse de si los componentes que está utilizando son atómicos o compuestos, o de si encapsulan la funcionalidad procedente de una u otra librería.

5.2.3. Elementos del modelo de componentes

En este apartado se recogen algunos de los detalles relativos al diseño y la implementación de los elementos que se acaban de definir como parte del modelo, esto es, el nuevo conjunto de tipos, los componentes atómicos y los componentes compuestos.

5.2.3.1. Definición del conjunto de tipos

Como ya se ha comentado anteriormente, cada librería de procesamiento de información visual suele utilizar su propio conjunto de tipos para representar los datos que manejan sus funciones (imágenes, máscaras, etc.). Así, para poder integrar en una misma aplicación funciones procedentes de distintas librerías, resulta necesario definir un nuevo conjunto de tipos (homogéneos o normalizados) así como un conjunto de funciones de transformación entre éstos y los empleados por las distintas librerías (ver Figura 36).

Para definir estos tipos homogéneos resulta necesario realizar un estudio exhaustivo de los empleados por las distintas librerías que se desea integrar. Sólo así es posible seleccionar el formato más adecuado para poder representarlos atendiendo al coste que conllevan las distintas conversiones, la portabilidad y extensibilidad de los datos y el rendimiento final del sistema.

A la vista de los tipos manejados por las librerías de Intel© [IPL] [OPENCV] y de Matrox© [MIL], seleccionadas inicialmente para su integración en IP-CoDER, se ha optado por implementar un conjunto de tipos equivalente al empleado por las primeras, esto es, se ha añadido al modelo de componentes una clase por cada tipo de datos no primitivo definido en las librerías IPL/OpenCV.

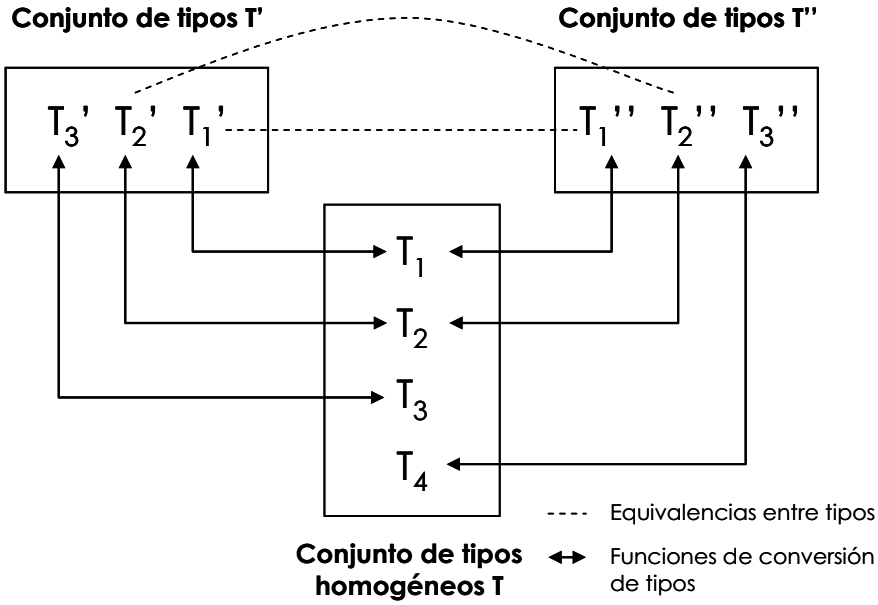


Figura 36. Definición de un conjunto de tipos homogéneo.
Definición del conjunto de tipos T a partir de los conjunto T' y T''. T_1' y T_2' representan tipos de datos equivalentes a T_1'' y T_2''. T_3' y T_3'' representan tipos que no tienen equivalente en T'' y T' respectivamente.

La elección de este conjunto de tipos en lugar del proporcionado por las librerías de Matrox© obedece fundamentalmente a que tanto IPL como OpenCV son gratuitas y de código abierto (al contrario de lo que ocurre con las MIL), lo que permite, por un lado, no tener que pagar derechos cada vez que se utilicen para desarrollar nuevas aplicaciones y, por el otro, tener acceso al código fuente donde se definen sus tipos de datos. Otra de las ventajas que se derivan del hecho de haber seleccionado como conjunto de tipos del modelo el proporcionado por una de las librerías que se van a integrar es que aquellos componentes que encapsulen funciones procedentes de dicha librería no necesitarán realizar ninguna transformación de tipos.

A continuación, a modo de ejemplo, se muestra la estructura de la clase diseñada para modelar el tipo de datos empleado para almacenar una imagen (ver Figura 37). Las variables de instancia de esta clase, denominada *gImage*, se corresponden con los campos del tipo *IplImage* definido por las librerías de Intel©. En color azul se han resaltado los métodos encargados de implementar las respectivas conversiones entre los distintos tipos de imágenes.

Las funciones de conversión entre los tipos *gImage* e *IplImage* resultan triviales. Sin embargo, el formato utilizado por las librerías MIL para representar las imágenes es muy distinto de los anteriores (identificadores de bajo nivel de tipo *MIL_ID* que almacenan la dirección de memoria donde comienza la imagen), por lo que la conversión no resulta tan evidente. Así, en este caso, resulta necesario utilizar varias de las funciones MIL para acceder a los datos contenidos en las imágenes a partir de su identificador para poder almacenarlos en un objeto de la clase *gImage* (y viceversa).

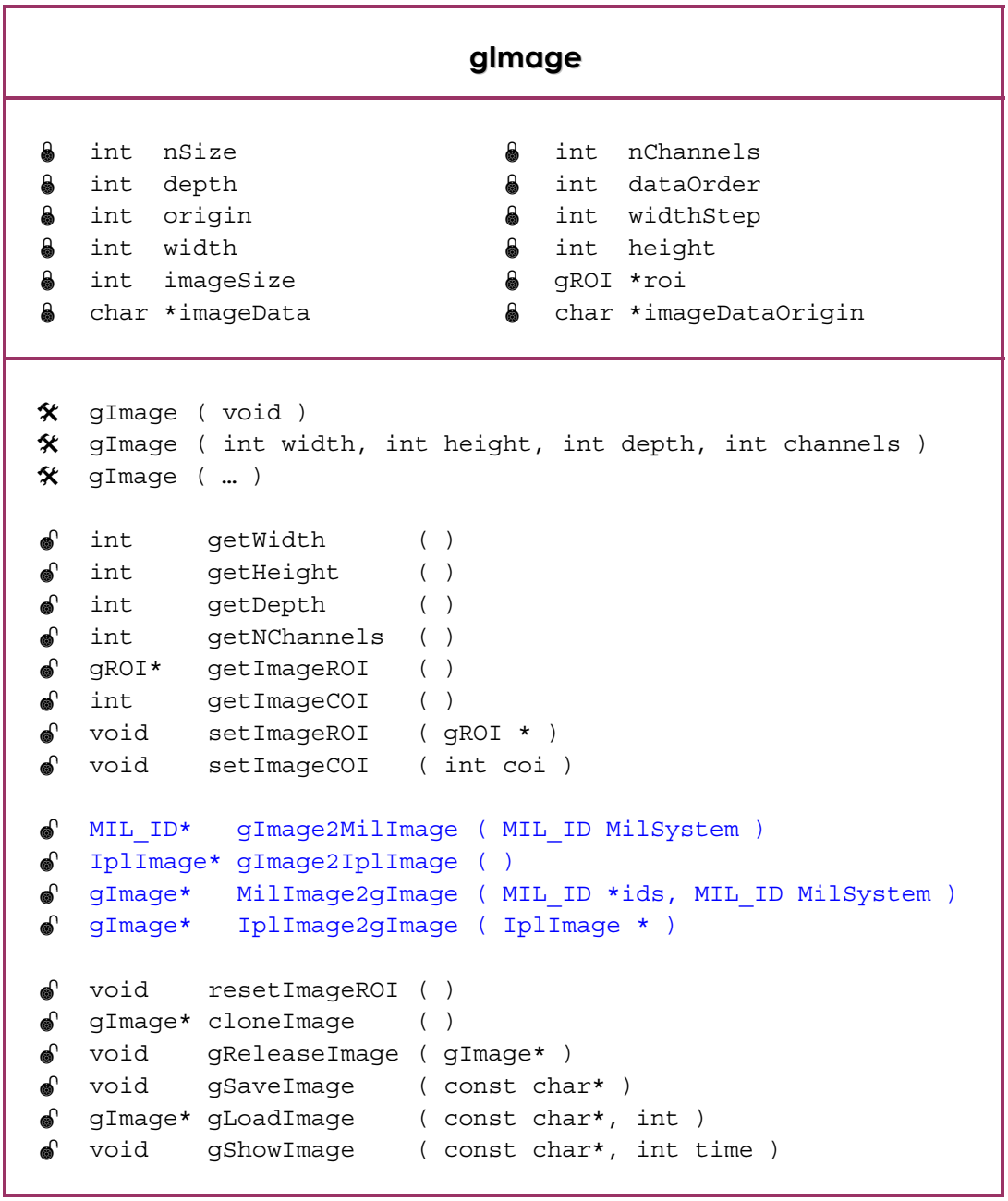


Figura 37. Estructura de la clase gImage.

Además del tipo *gImage* utilizado para almacenar imágenes, el modelo de componentes incorpora, entre otros, los siguientes tipos:

- ***gROI*:** este tipo almacena información sobre las regiones de interés (*Regions Of Interest*, ROI) definidas en cada imagen. El uso de ROI permite reducir considerablemente el coste computacional de muchas aplicaciones puesto que los algoritmos seleccionados sólo se aplicarán a estas regiones en lugar de a toda la imagen. El equivalente IPL/OpenCV es la estructura *CvRoi*.
- ***gMat***: Representa una matriz multi-canal. Su equivalente IPL/OpenCV es la estructura *CvMat*.

- ***gPoint***: Representa un punto de dos dimensiones con coordenadas enteras, cuyo equivalente IPL/OpenCV es la estructura *CvPoint*.
- ***gPoint2D32f***: Representa un punto de dos dimensiones con coordenadas decimales, cuyo equivalente IPL/OpenCV es la estructura *CvPoint2D32f*.
- ***gPoint3D32f***: Representa un punto de tres dimensiones con coordenadas decimales, cuyo equivalente IPL/OpenCV es la estructura *CvPoint3D32f*.
- ***gRect***: Representa un rectángulo caracterizado por su tamaño (altura y anchura) y por el desplazamiento de su posición respecto al origen de coordenadas de la imagen. Utilizado para establecer la ROI de las imágenes, su equivalente IPL/OpenCV es la estructura *CvRect*.
- ***gSize***: Representa un rectángulo definido exclusivamente por su tamaño (altura y anchura en píxeles). Se utiliza para crear imágenes vacías de un tamaño específico Y su equivalente IPL/OpenCV es la estructura *CvSize*.
- ***gSize2d32f***: Representa un rectángulo como el definido por un *gSize*, pero en el que se permiten valores decimales.

Una vez presentado el conjunto de tipos homogéneos definidos como parte del modelo de componentes, a continuación se detalla el diseño y la implementación de los componentes atómicos que serán la base sobre la que posteriormente se definirán los componentes compuestos.

5.2.3.2. Componentes atómicos

Como ya se ha comentado anteriormente, los componentes atómicos encapsulan funciones procedentes de distintas librerías de procesamiento de información visual, consiguiendo homogenizar sus interfaces y en consecuencia facilitando su uso conjunto dentro de una misma aplicación.

Los componentes atómicos ofrecen, como parte de su interfaz, un conjunto de conectores de entrada y salida cuyo tipo deberá ser alguno de los definidos como parte del modelo. Internamente y de manera totalmente transparente para el usuario, estos conectores estarán ligados a los correspondientes argumentos de entrada y salida de la función que encapsula el componente. El tipo de estos argumentos será alguno de los propios de la librería a la que pertenece la función encapsulada, por lo que cuando los conectores del componente se conecten con los argumentos de la función será necesario llevar a cabo las correspondientes conversiones de tipos.

Para modelar la estructura y el comportamiento de estos componentes atómicos se ha diseñado la clase *gComponent,* cuya estructura se muestra a continuación en la Figura 38.

Figura 38. Estructura de la clase gComponent.

A continuación se comenta brevemente cada una de las variables de instancia de esta clase; sus métodos más relevantes se detallarán más adelante en el apartado 5.3.

- **string name:** Nombre del componente. Este será igual al nombre de la función encapsulada eliminando cualquier identificador asociado a su origen. Esto es, si se desea crear un componente que encapsule la función cvThreshold de OpenCV, el nombre del componente deberá ser Threshold (sin los caracteres iniciales "cv"). Esta política de asignación de nombres es necesaria puesto que en las librerías MIL no se añade ninguna secuencia adicional a los nombres de las funciones, mientras que en OpenCV se añade al la marca "cv" al principio del nombre de sus función. Así, cuando el usuario desee crear un componente, no deberá preocuparse de añadir esta marca al nombre del componente, ya que esta se añadirá automáticamente cuando se genere el código, verificando el origen de la función encapsulada.
- **int ninput:** Número de entradas del componente. Debe coincidir con el número de parámetros de entrada de la función que se desee encapsular.
- **int noutput:** Número de salidas del componente. También debe coincidir con el número de parámetros de salida de la función a encapsular (sólo en el caso de componentes atómicos, ya que los componentes compuestos pueden enmascarar las salidas no deseadas).
- **Data* input[]:** Especificación de las entradas del componente, contiendo los tipos de datos y los nombres de los parámetros de entrada a la función encapsulada. La estructura interna de la clase Data se detalla posteriormente.
- **Data* output[]:** Especificación de las salidas del componente, contiendo los tipos de datos y los nombres de los parámetros de de salida de la función encapsulada.
- **string include:** Ficheros de cabecera requeridos por la función encapsulada en el componente. Generalmente se suelen incluir las librerías principales de OpenCV o MIL, según sea uno u otro el origen de la función. También deben añadirse las clases asociadas a los tipos de datos homogéneos definidos en el marco (gImage, gMat, etc.), en caso de que sean usadas como parámetros de entrada o salida.
- **string description:** Descripción textual del componente, que permite obtener una pequeña reseña relativa a su funcionalidad. Éste campo resulta especialmente importante cuando el componente encapsula funciones definidas por el usuario, ya que en estos casos no se puede recurrir a ninguna otra fuente de documentación adicional.
- **string code:** Código fuente asociado al componente. Se genera de forma automática tras la definición de los demás campos del componente y la llamada a la función wrap.
- **bool returnOut:** Indica si la función encapsulada devuelve su salida mediante un return o modificando un argumento pasado por referencia. En caso de ser true, la primera posición del array de salidas (output[0]) almacenará el resultado devuelto por la función, una vez generado el código del componente. En caso de ser false, todas las salidas serán devueltas por referencia.

- **string origin:** Librería de la que procede la función encapsulada por el componente. Este campo podrá tomar uno de los siguientes valore: "OpenCV", "MIL", "OwnCV" (función definida por el usuario que utilice las librerías OpenCV), y "OwnMIL" (función definida por el usuario que hace uso de la librería MIL).
- **bool dataCheck[]:** Este campo sirve para comprobar que todas las entradas y salidas del componente han sido asignadas a parámetros de entrada y salida de la función. El procedimiento es sencillo, ya que cada vez que se añade una entrada o una salida al componente, se coloca un valor true en el elemento correspondiente del array. Una vez finalizada la descripción del componente, se comprueba que todos los elementos tanto del array de entradas como del de salidas, están a true. De no ser así se advertirá el usuario del error.
- **string ID:** Identificador del componente. Este campo toma valores distintos en cada nuevo objeto creado, de modo que es posible distinguir entre sí los componentes creados a partir de la misma definición. Este identificador se asigna de forma automática, de manera que el usuario no tiene que preocuparse de asignar un valor único a este identificador.

Una vez descritos los componentes atómicos, en el siguiente apartado se comentan los detalles más relevantes relativos a la estructura de los componentes compuestos.

5.2.3.3. Componentes compuestos

Los componentes compuestos se construyen agrupando dos o más componentes previamente definidos (ya sean atómicos o compuestos), conectándolos entre sí y seleccionando cuáles de las entradas y salidas disponibles deben formar parte de su interfaz pública. La clase diseñada para modelar estos componentes es la denominada *gCComponent* que es una agregación y a la vez hereda de la clase *gComponent* (ver Figura 39) por lo que su estructura contendrá, al menos, las variables de instancia definidas en la superclase: nombre, número de entradas y salidas, origen, etc. Nótese que el campo origen de los componentes compuestos siempre se rellena con la palabra reservada "Compose" indicando que en su interior encapsula otros componentes y no una función de librería.

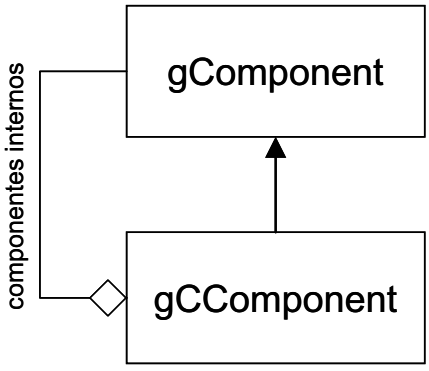


Figura 39. Estructura de la clase gComponent.

La clase *gCComponent* añade a las variables de instancia heredadas las siguientes:

- **gComponent *components[]:** Componentes que forman el componente compuesto. El orden de los componentes en el array es crítico, ya que en el procesamiento realizado para la autogeneración del código fuente se sigue una secuencia de ejecución dada por el orden de los componentes. Por lo tanto, al rellenar el array de componentes se debe verificar que un componente determinado nunca necesite el procesamiento previo de un componente posterior a él en dicho array.
- **int numComponents:** Número de componentes internos que contiene el componente compuesto. Este campo es necesario puesto que, por lo general, el número de componentes no coincidirá el tamaño del array (fijado en tiempo de compilación).
- **struct internalLink internalLinks[]:** Conexiones internas definidas entre los componentes incluidos en el componente compuesto. Una conexión interna viene definida por una estructura en la que se identifican los dos extremos del enlace.
- **struct constantLink constantLinks[]:** Conexiones a constantes dentro del componente compuesto, lo que permite conectar una entrada constante (número, carácter, string, etc.) a uno de los componentes internos Esta conexión difiere de la anterior (definida entre dos componentes), ya que en ese caso la constante se transfiere al componente en tiempo de compilación, mientras que en el caso de la transferencia de información entre dos componentes, ésta se realiza en tiempo de ejecución (una vez que el componente origen calcula la salida que desea transmitir).
- **int numInternalLinks:** Número de conexiones internas entre los componentes que forman parte del componente compuesto.
- **int numConstantLinks:** Número de conexiones a constantes definidas en el componente compuesto.

5.3. IP-CoDER: descripción de la funcionalidad implementada

IP-CoDER es una herramienta de programación visual y generativa diseñada para dar soporte al modelo de componentes presentado en el apartado anterior. Esta herramienta permite al usuario crear, documentar, modificar, ensamblar, y guardar los nuevos componentes en una librería, de donde podrá recuperarlos posteriormente para reutilizarlos tantas veces como sea necesario. Asimismo, esta herramienta permite generar, de manera totalmente automática, prototipos ejecutables de VIPS mediante la instanciación de la línea LPS-VIPS (definida en el Capítulo 4) y el posterior relleno del esqueleto obtenido mediante la selección y el conexionado de varios de los componentes disponibles en la librería. Como se puede ver en la Figura 40, el manejo y el aspecto de IP-CoDER resultan muy sencillos e intuitivos para el usuario.

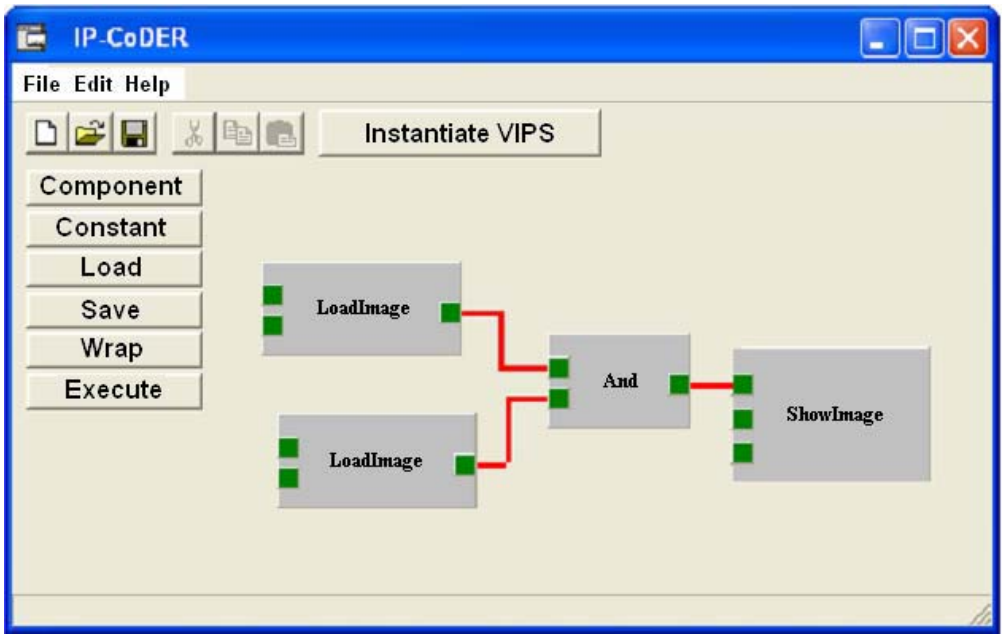


Figura 40. Aspecto de la herramienta IP-CoDER.

En los siguientes apartados se realiza un breve resumen de la funcionalidad que ofrece la herramienta IP-CoDER relacionada con la definición, el conexionado, el almacenamiento y la recuperación de componentes, así como con la generación automática de prototipos ejecutables de VIPS.

5.3.1. Creación de componentes atómicos

IP-CoDER permite la creación de componentes atómicos a partir de las funciones incluidas en las librerías inicialmente incorporadas a la herramienta (IPL, OpenC y MIL), así como de funciones previamente implementadas por el propio usuario en C/C++.

Para crear un nuevo componente atómico el usuario deberá seleccionar el botón "Component" de la aplicación, de modo que ésta le ofrecerá un formulario (ver Figura 41) para que rellene, entre otros, los siguientes datos: el nombre de la función que se va a encapsular, el número y el tipo de sus argumentos, la librería de la que procede, el nombre del nuevo componente atómico así como el número y el tipo de sus conectores y cómo se relacionan éstos con los argumentos de la función, etc.

Una vez completado el formulario anterior, se creará una versión gráfica del componente atómico invocando al constructor de la clase *vgComponent*. Éste inicializará las variables de instancia del objeto gráfico entre las que se encuentran: un panel en el que se muestra el nombre del componente, varios botones que representan los distintos conectores de entrada y salida, y un objeto de la clase *gComponent* que se inicializará invocando al constructor de esta clase, pasándole como argumentos los datos introducidos por el usuario en el formulario. La relación entre las clases *gComponent* y *vgComponent* es la que se muestra en la Figura 42.

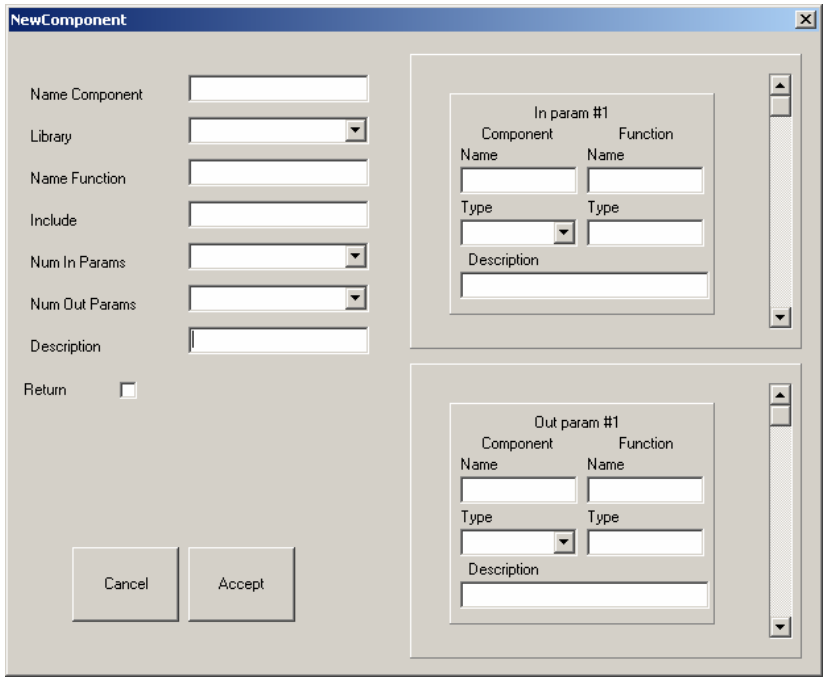


Figura 41. Formulario de definición de componentes atómicos.

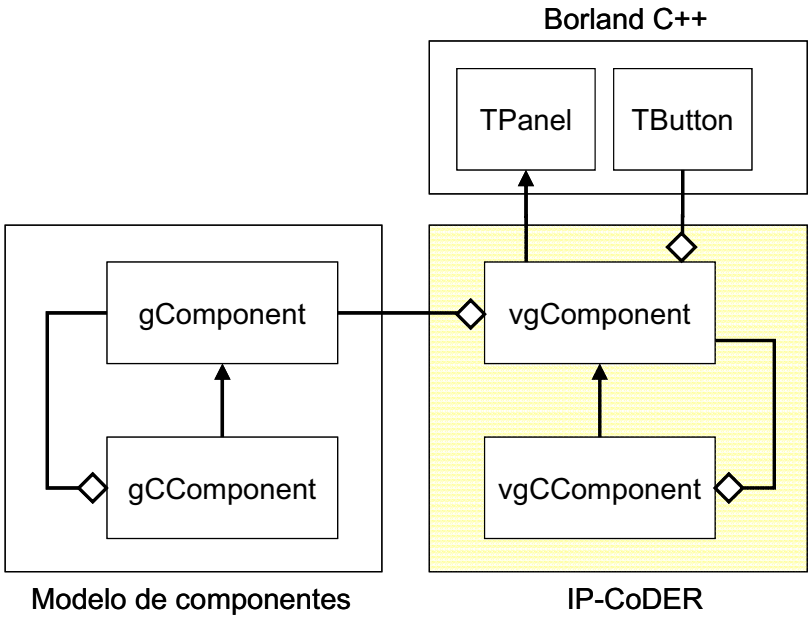


Figura 42. Relación entre las clases utilizadas por IP-CoDER

Como ya se comentó anteriormente al describir la estructura de los componentes atómicos (apartado 5.2.3.2) éstos cuentan, entre sus variables, con una denominada *code*, en la que se almacena el código de envoltura (*wrapper*) de la función que alberga el componente en su interior. Ésta, como el resto de las variables de instancia, se inicializa al invocar al constructor de la clase (método *wrap*) una vez que el usuario ha rellenado la plantilla descrita previamente. A continuación se muestra, a modo de ejemplo, el código generado para un componente atómico configurado por el usuario con los datos que se muestran en la Tabla 12.

Descripción del componente				
Name = LoadImage			Origin = OpenCV	
Nº input = 2			Nº output = 1	
Return = trae			Include = "gImage.h"	
Description: Carga una imagen desde un fichero especificado por el usuario.				
Especificación de los conectores de entrada				
gType	gName	oType	oName	Description
char*	fileName	char*	fileName	Fichero de imagen.
int	isColor	Int	isColor	Indica si imagen es en color.
Especificación de los conectores de salida				
gType	gName	oType	oName	Description
gImage	Out	IplImage	dst	Imagen cargada del fichero.

Tabla 12. Datos especificados por el usuario para crear un componente atómico.

```
gImage* gLoadImage(char* fileName,int isColor)
{
//Function Call
IplImage* dst = cvLoadImage(fileName,isColor);

//Output Type Conversion
gImage* out = gImage::IplImage2gImage(dst);

//Free memory
cvReleaseImage(&dst);

return out;
}
```

Figura 43. Código del *wrapper* generado para el componente atómico anterior.

Como se puede observar en el código mostrado en la Figura 43, el componente *gLoadImage* encapsula la función *cvLoadImage* a la que invoca pasándole como argumentos los datos que le llegan a través de sus conectores de entrada, una vez convertidos al formato requerido por dicha función. Del mismo modo, una vez completada la ejecución de la función, transforma su resultado al tipo homogéneo correspondiente y lo coloca en el conector de salida correspondiente.

Una vez descrito el procedimiento implementado por IP-CoDER para la creación de componentes atómicos en el apartado siguiente se describe el correspondiente a la creación de los componentes compuestos.

5.3.2. Creación de Componentes Compuestos

La creación de componentes compuestos se realiza siguiendo la siguiente secuencia de cuatro pasos:

- **Paso 1:** En primer lugar se deben seleccionar los componentes que se van a encapsular dentro del componente compuesto. Para ello, el usuario podrá cargar cualquiera de los previamente creados y almacenados en la librería, o definir nuevos componentes atómicos si lo estima oportuno. Estos componentes se almacenarán en una tabla de manera ordenada, atendiendo a la secuencia en la que deben ejecutarse. Para ello debe cumplirse la siguiente condición: cualquier componente X de cuyas salidas dependa el componente Y, deberá almacenarse en una posición posX anterior a la posición del componente Y (posX < posY). La ordenación es necesaria dado que, a la hora de generar el código del componente compuesto, las llamadas a los componentes internos se realizarán en el orden en que éstos han sido almacenados en la tabla. De hecho, IP-CoDER impide que se utilicen las salidas de un componente cuyas entradas no hayan sido previamente asignadas. Nótese que esta condición de ordenación puede dar lugar a distintas configuraciones lógicamente equivalentes y por lo tanto correctas.
- **Paso 2:** Una vez seleccionados los componentes internos y tras pulsar el botón "Wrap" de la aplicación, IP-CoDER ofrecerá al usuario un formulario muy similar al utilizado para especificar los componentes atómicos. Una vez completado este formulario se utilizarán los datos introducidos por el usuario para invocar al constructor de la clase v*gCComponent*.
- **Paso 3:** En este punto debe especificarse la interfaz externa del componente compuesto conectando sus conectores de entrada/salida con alguno de los conectores de sus componentes internos.
- **Paso 4:** Por último y para completar la definición del componente compuesto se debe especificar su arquitectura interna, esto es, cómo están conectados entre sí sus componentes internos.

Como en el caso de los componentes atómicos, la creación de compoenntes compuestos también conlleva la generación del *wrapper* correspondiente. Sin embargo, el método *wrap* implementado en la clase *gCComponent* difiere considerablemente del implementado en su superclase. Por una parte, al encapsularse componentes homogéneos (en lugar de funciones heterogéneas) no resulta necesario llevar a cabo ninguna conversión de tipos al definir su interfaz externa. Por otra parte, en el caso de que internamente se conecten dos componentes que encapsulen funciones pertenecientes a una misma librería sería posible evitar las conversiones de tipos que llevan a cabo sus *wrappers* correspondientes. Cuando esto ocurre, IP-CoDER es capaz de detectarlo y de eliminar las conversiones de tipos innecesarios, consiguiendo una mejora considerable en el rendimiento de muchos de los componentes compuestos.

5.3.3. Definición de Constantes

Además de componentes atómicos y compuestos, IP-CoDER ofrece la posibilidad de añadir constantes a los diseños. Para ello, el usuario deberá seleccionar la opción "Constant" de la aplicación y rellenar el formulario que se muestra a continuación en la Figura 44. Tras seleccionar el tipo y el valor de la constante se creará un componente gráfico con un único conector de salida que podrá conectarse a la entrada de cualquiera de los componentes atómicos o compuestos del diseño (siempre que tengan tipos compatibles).

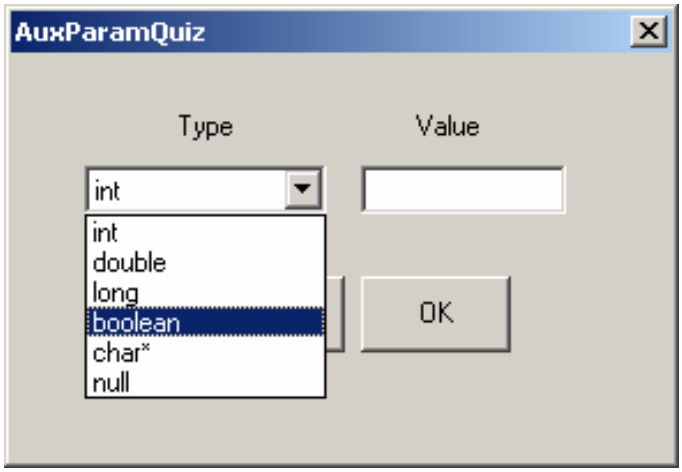


Figura 44. Formulario de creación de nueva constante.

5.3.4. Almacenamiento y recuperación de componentes

El almacenamiento de los componentes en una librería o repositorio resulta indispensable para poder recuperarlos y reutilizarlos con posterioridad en otras aplicaciones. Para ello, IP-CoDER proporciona en su interfaz las opciones "*Load*" y "*Save*". Al seleccionar la opción "*Load*" el usuario podrá recuperar cualquiera de los componentes previamente almacenados en la librería de modo que éste se añadirá al diseño (componente) que esté activo en ese momento.

Mediante la opción "Save" el usuario podrá almacenar el componente activo en pantalla en un fichero de texto con extensión .g y cuyo nombre será el mismo que el del componente. Al seleccionar esta opción se invocará el método *save* de la clase *gComponent* o *gCComponent*, dependiendo de si el componente activo es atómico o compuesto. Este método se encargará de comprobar si ya existe un fichero con el nombre del componente activo y, de ser así, ofrecerá al usuario la posibilidad de sobrescribirlo o bien de guardarlo con otro nombre. A continuación se almacenarán, por orden y en líneas separadas, los valores de las variables de instancia del componente activo. En la Figura 45 se muestra, a modo de ejemplo, el formato con el que se ha almacenado el componente atómico *Threshold* en el fichero Threshold.g.

```
name=Threshold
origin=OpenCV
ninput=4
noutput=1
include=#include "gImage.h"
description=Umbraliza una imagen
Input=gImage(.,.,.,.) | in | IplImage | src | descripcion
Input=double | threshold | double | threshold | descripcion
Input=double | maxValue | double | maxValue | descripcion
Input=int | thresholdType | int | thresholdType | descripcion
output=gImage(.,.,.,.) | out | IplImage | dst | descripcion
returnOut=false
code = void gThreshold(gImage *in,gImage *out,double threshold,double maxValue,int threshType) {
//Input Type conversion
IplImage *src=in->gImage2IplImage();
//Output declaration
IplImage *dst;
if(out==NULL) dst=NULL;
else
    dst=cvCreateImage(cvSize(src->width,src->height),src->depth,src->nChannels);
//Function Call
cvThreshold(src,dst,threshold,maxValue,threshType);
//Output Type Conversion
if(out!=NULL)
    *out= *gImage::IplImage2gImage(dst);
//Free memory
cvReleaseImage(&src);
cvReleaseImage(&dst);
}
```

Figura 45. Formato con el que se almacena un componente atómico.

El almacenamiento de componentes compuestos requiere que, además de los valores de las variables de instancia, se almacenen los datos relativos a los componentes internos que éste encapsula. Dado que estos componentes pueden ser a su vez compuestos, se ha optado por implementar el método *save* de la clase *gCComponent* de manera recursiva. En la Figura 46 se muestra el formato con el que se almacenaría un componente compuesto por 3 componentes internos, estando el segundo de ellos compuesto a su vez de dos componentes atómicos.

Atributos del componente compuesto
-COMPONENT-
Atributos del componente interno 1 (atómico)
-COMPONENT-
Atributos del componente interno 2 (compuesto)
-COMPONENT-
Atributos del componente interno 2.1 (atómico)
-COMPONENT-
Atributos del componente interno 2.2 (atómico)
-COMPONENT-
Atributos del componente interno 3 (atómico)
…

Figura 46. Formato con el que se almacenan los componente compuestos.

5.3.5. Instanciación de la línea LPS-VIPS

IP-CoDER permite al usuario crear nuevas instancias de la línea LPS-VIPS mediante la opción "Instantiate VIPS". Al pulsar este botón, IP-CoDER cerrará todos los componentes activos en ese momento solicitando al usuario confirmación para guardar los posibles cambios realizados en ellos. Una vez hecho esto, IP-CoDER mostrará al usuario una serie de formularios para que pueda seleccionar las opciones y alternativas que desea que tenga el nuevo VIPS. Estos formularios reproducen las tablas de decisión del modelo de análisis y de la arquitectura software de la línea LPS-VIPS descritos en el Capítulo anterior.

Una vez que el usuario haya configurado todas las características variables del VIPS, IP-CoDER procederá a generar el esqueleto de la aplicación siguiendo las indicaciones incluidas en las tablas de decisión. Este esqueleto estará formado por un conjunto de clases abstractas que modelarán el comportamiento de los subsistemas incluidos en el VIPS según las opciones seleccionadas por el usuario (procesamiento de la información visual, control, alarmas, etc.) y una clase ejecutable que contendrá un menú con las opciones seleccionadas, todas ellas operativas aunque asociadas a métodos abstractos de las clases anteriores, por lo que su ejecución no tendrá un efecto visible.

Una vez creado el esqueleto del VIPS, IP-CoDER ofrecerá al usuario la posibilidad de rellenar el método *procesarInformaciónVisual* incluido en la clase *SistemaProcesamientoVisual* con el código asociado a cualquiera de los componentes previamente creados y almacenados en la biblioteca. La ejecución de estos componentes podrá haber sido previamente simulada y validada mediante el procedimiento que se expone a continuación.

5.3.6. Generación de prototipos ejecutables para su validación

IP-CoDER ofrece la posibilidad de ejecutar el código asociado al componente activo en cada momento a través del botón "Execute". Al seleccionar esta opción se

pedirá al usuario que proporcione un valor para todas aquellas entradas del componente activo que no estén conectadas y que por lo tanto estén señaladas en azul (ver Figura 47). Para ello, IP-CoDER mostrará al usuario un formulario con tantos campos como entradas deba rellenar (ver Figura 48), indicándole el tipo de cada una de ellas. Una vez completas estas entradas, la herramienta generará una aplicación ejecutable cuyo método *main* contendrá: (1) la declaración de una constante por cada una de las entradas fijadas por el usuario y (2) el código del componente activo que se desea validar.

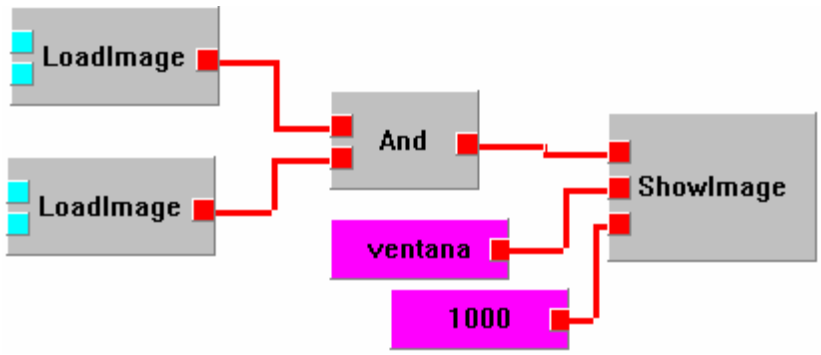


Figura 47. Diseño del componente que se desea ejecutar.

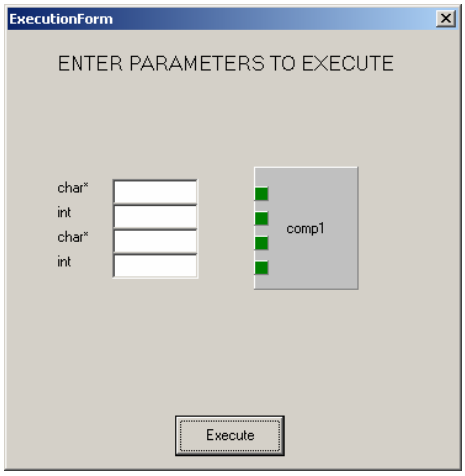


Figura 48. Formulario para ejecutar el diseño mostrado en la Figura 47.

Una vez descrita la principal funcionalidad proporcionada por IP-CoDER, en el siguiente apartado se muestra uno de los casos de estudio realizados con esta herramienta a fin de validar su correcto funcionamiento y demostrar su utilidad.

5.4. Validación de la herramienta mediante un caso de estudio sencillo

Con el fin de validar el correcto funcionamiento de la herramienta IP-CoDER se han desarrollado varios casos de estudio uno de los cuales, consistente en la implementación de un VIPS para segmentación de caras en imágenes en color, se presenta a continuación.

En primer lugar se creo una instancia de la línea LPS-VIPS seleccionando únicamente las opciones básicas (sólo funcionalidad relativa al procesamiento de la información visual). El esqueleto de esta aplicación se rellenó diseñando un componente compuesto cuyo diseño se describe a continuación.

La secuencia de algoritmos empleada para realizar este diseño ya había sido probada anteriormente como parte de un trabajo previo en el que se construyó un sistema de detección y segmentación de caras para reconocimiento de expresiones faciales [García 00]. Gracias a ello el proceso la selección de los algoritmos resultó sencillo (ver Figura 49). Sin embargo, dado que la librería cuenta aún con un número reducido de componentes, no todos ellos estaban disponibles. Así, hubo que implementar, entre otros, un componente para detección y caracterización de *blobs*. Esto supuso una buena ocasión para comprobar la integración de componentes generados a partir de código escrito por el usuario con otros generados a partir de funciones de las distintas librerías.

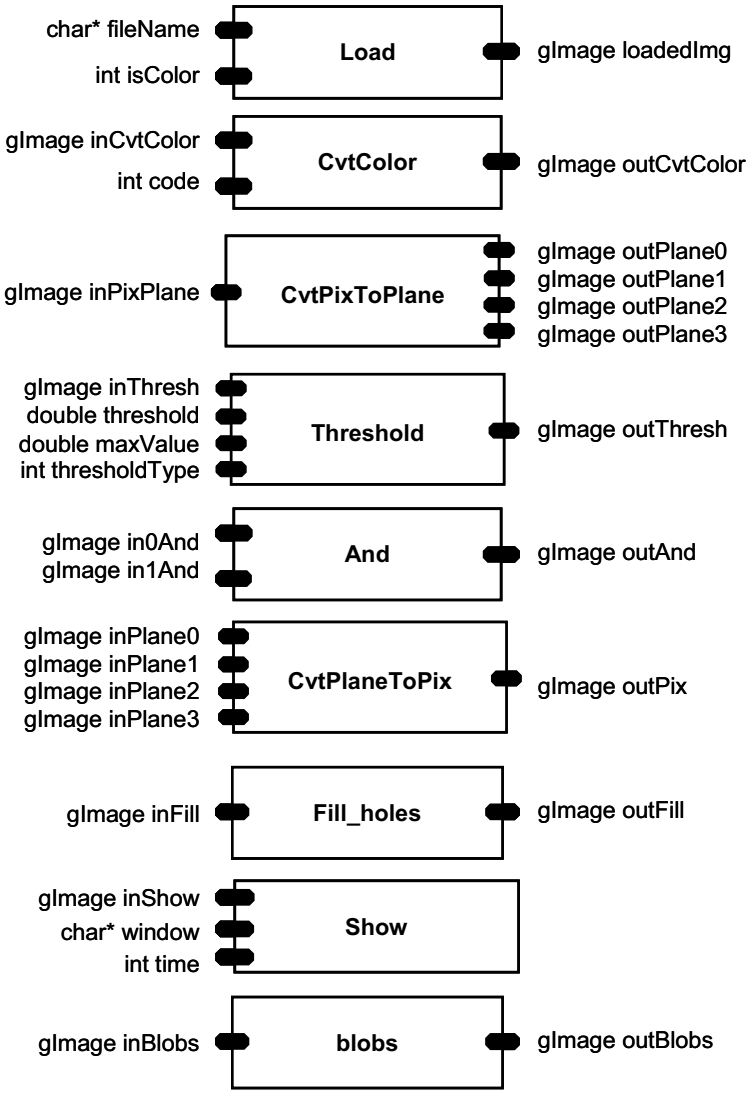


Figura 49. Componentes que se emplearán durante el caso de estudio

Una vez disponibles todos los componentes necesarios se procedió a especificar el diseño utilizando el entorno gráfico proporcionado por IP-CoDER. En la Figura 50 se muestra el diagrama obtenido como resultado.

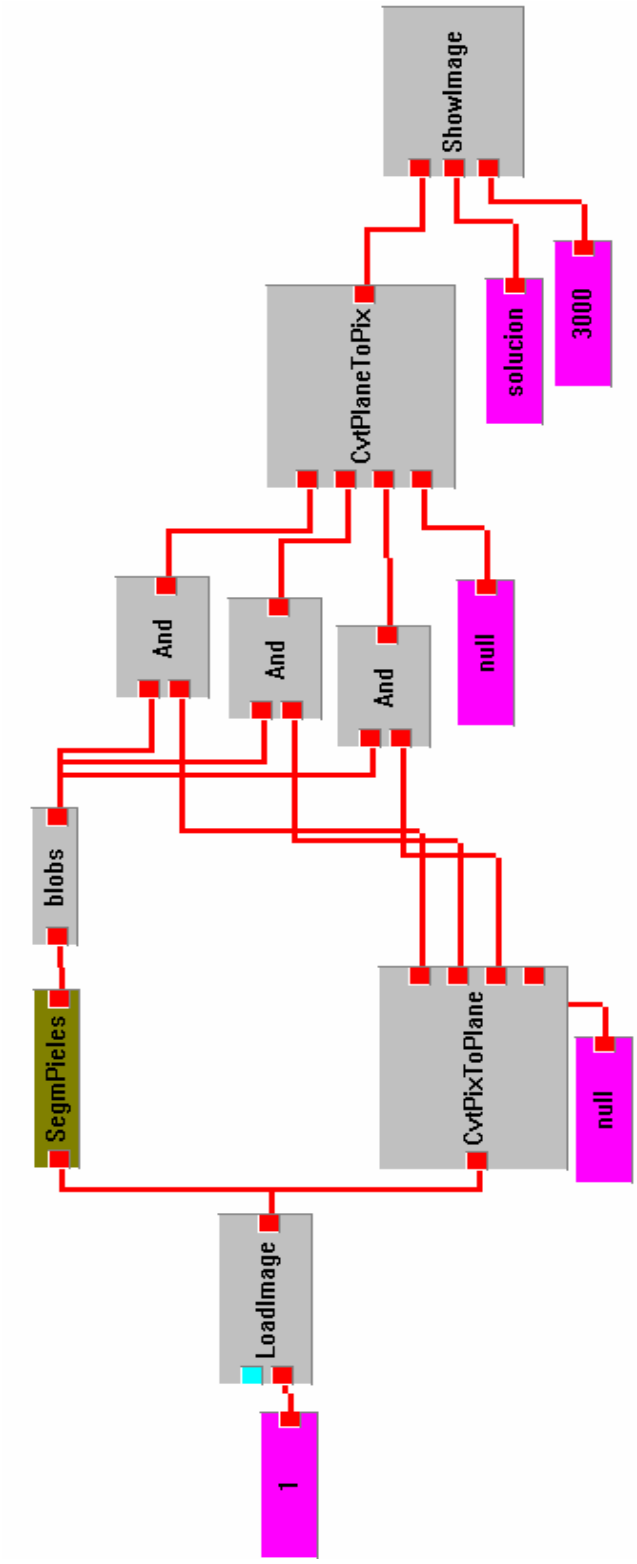


Figura 50. Especificación del prototipo del caso de estudio utilizando IP-CoDER.

Como se puede apreciar en el diagrama de componentes de la Figura 50, uno de los componentes del prototipo es compuesto (color verde). Esto no se debe a que este componente existiera previamente en la librería sino a que, para mejorar la legibilidad del diagrama se decidió implementar los algoritmos correspondientes a la segmentación de pieles en un componente compuesto separado. El detalle de los componentes internos de este componente compuesto se muestra en la Figura 51.

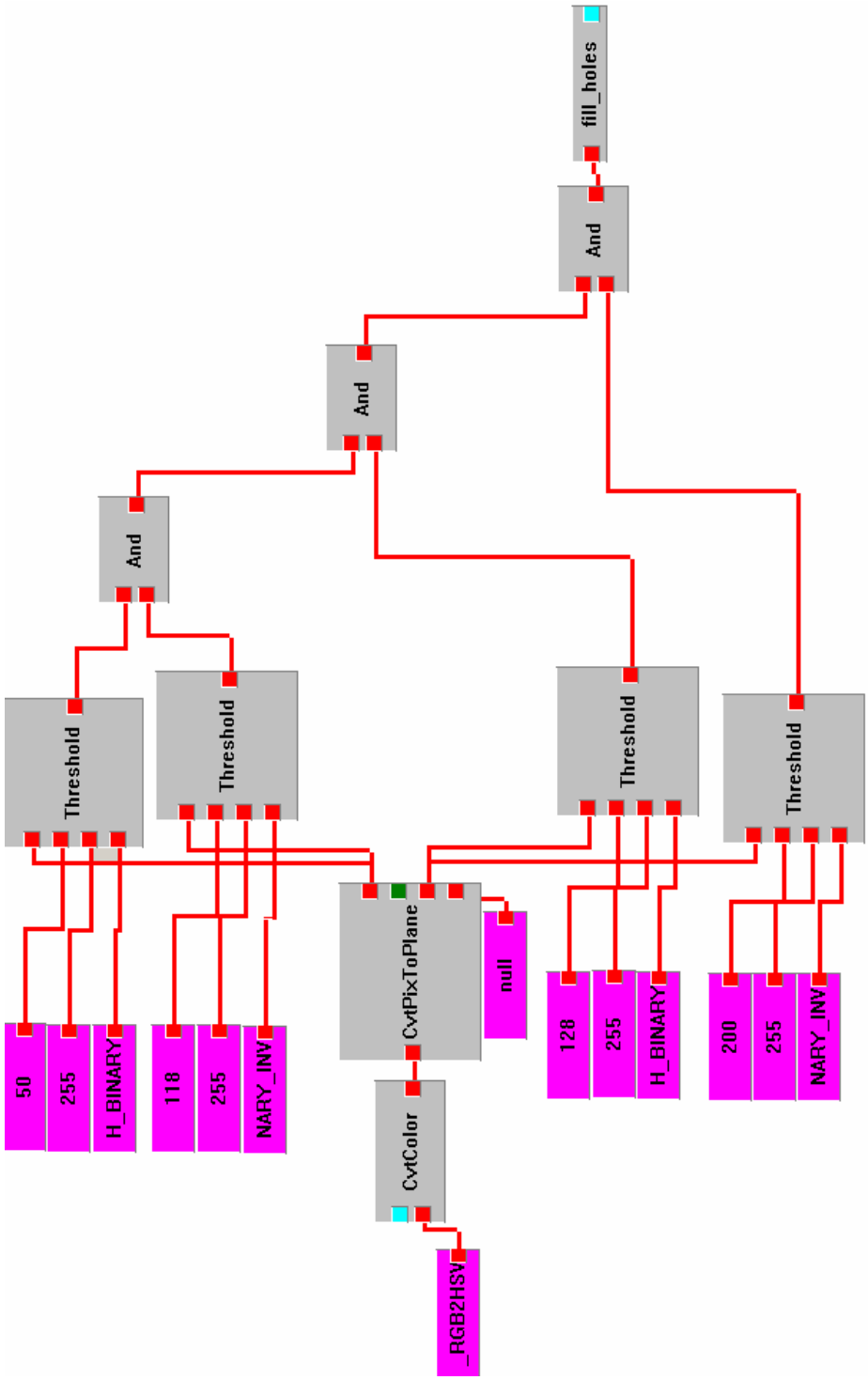


Figura 51. Componentes internos del componente compuesto SegmentPieles

Finalmente, una vez completado el diseño se procedió a crear el correpondiente ejecutable y a validarlo con distintas imágenes de entrada. Como se observa en el diagrama de la Figura 51, es precisamente el nombre de la imagen que se quiere utilizar durante la ejecución él único parámetro configurable del prototipo (conector de color azul en el componente LoadImage).

Algunos de los resultados obtenidos de la ejecución del prototipo utilizando distintas imágenes de entrada se muestran a continuación en la Figura 52.

Figura 52. Resultados de la validación del prototipo con distintas imágenes.

Con este caso de estudio se ha conseguido demostrar la utilidad de la herramienta para desarrollar aplicaciones de procesamiento de imágenes de forma rápida, sencilla e intuitiva.

Esta página se deja en blanco intencionadamente

Capítulo 6

Conclusiones, Aportaciones y Líneas de Trabajo Futuras

En este capítulo se presentan las conclusiones de la Tesis, las principales aportaciones que ofrece y los resultados concretos obtenidos durante su realización. Por último, se proponen algunas líneas de investigación en las que resultaría interesante profundizar en el futuro.

6.1. Conclusiones

A continuación se enumeran algunas de las conclusiones más relevantes a las que se ha llegado durante el desarrollo de esta Tesis Doctoral.

- El estudio de los distintos tipos de VIPS que se desarrollan en la actualidad permite, por una parte, identificar una serie de rasgos comunes a todos ellos y, por otra, clasificarlos atendiendo a aquellos aspectos que les diferencian entre sí. Gracias a este análisis es posible definir un patrón genérico para los VIPS en el que queden recogidos todas aquellas características identificadas como comunes y en el que se representen, a modo de puntos de variabilidad, aquellas opciones o alternativas ofrecidas sólo por algunos de ellos. A partir de este patrón de diseño es posible derivar cualquier nuevo VIPS sin tener que partir desde cero cada vez, tal y como ocurre en la actualidad.
- Abordar el diseño y la implementación de sistemas mixtos Hw/Sw y en particular la construcción de Sistemas de Procesamiento de Información Visual desde una perspectiva de Ingeniería del Software (en lugar de seguir el actual enfoque muy centrado en el Hw) reporta, entre otras ventajas, una mejora considerable en la flexibilidad, el grado de reutilización y el tiempo de desarrollo de estos productos.

- La integración de varios de los paradigmas actuales de desarrollo de software en una única metodología para la construcción de VIPS no sólo se demuestra viable sino que produce un efecto sinérgico ya que, además de aprovechar las ventajas que ofrece cada uno de estos paradigmas de manera individual, se consigue reducir considerablemente sus limitaciones gracias a su complementariedad.
- El desarrollo de un marco de componentes software para procesamiento de imágenes permite integrar varias de las herramientas y librerías de procesamiento de imágenes actualmente disponibles en el mercado, lo que facilita su uso conjunto para el desarrollo de VIPS.
- La implementación de una herramienta de programación visual y generativa para gestionar los componentes de procesamiento de imágenes antes mencionados permite, al desarrollador de VIPS, abstraerse de muchos de los detalles de la implementación y consigue no sólo reducir el tiempo de desarrollo de estas aplicaciones sino también mejorar su grado de reutilización, modularidad, extensibilidad y corrección.

6.2. Principales aportaciones

A continuación se enumeran de manera resumida las principales aportaciones realizas en esta Tesis.

- Se ha realizado un estudio exhaustivo y una clasificación de los Sistemas de Procesamiento de Información Visual que ha permitido identificar aquellas características que tienen todos ellos en común así como las principales diferencias entres unos y otros.
- Se ha diseñado una nueva metodología de desarrollo de VIPS basada en la integración de líneas de productos, desarrollo de software basado en componentes, programación visual y generación automática de código. Esta metodología cubre todas las fases del ciclo de vida de estos productos, desde la captura de requisitos y el análisis del dominio, pasando por la especificación de su arquitectura software genérica, hasta su implementación y posterior mantenimiento utilizando la herramienta IP-CoDER que se comenta más adelante.
- Se ha diseñado un modelo de componentes para albergar la funcionalidad ofrecida por varias de las herramientas y librerías de procesamiento de imágenes existentes en el mercado, lo que permite su uso conjunto dentro de un mismo entorno de desarrollo. Aunque en la actualidad existen varios modelos de componentes comerciales como CORBA, .NET, o JavaBeans que proporcionan unas excelentes prestaciones en cuanto a interoperabilidad, ejecución distribuida, etc., éstos no resultan especialmente eficientes. Esta ha sido la principal razón por la que se ha optado por diseñar un nuevo modelo de componentes más simple y eficiente que los anteriores ya que se limita a ofrecer las prestaciones necesarias para la construcción de VIPS.

- Se ha desarrollado la herramienta de programación visual IP-CoDER que implementa el modelo de componentes previamente definido y con la que es posible llevar a cabo las siguientes operaciones:
  - Crear nuevos componentes simples a partir de la funcionalidad ofrecida por varias de las herramientas y librerías de procesamiento de imágenes existentes en el mercado.
  - Crear nuevos componentes compuestos seleccionando y conectando adecuadamente varios de los componentes previamente definidos (ya sean simples o también compuestos).
  - Almacenar ambos tipos de componentes en una librería de la que posteriormente podrán recuperarse para ser reutilizados.
  - Generar automáticamente prototipos ejecutables de VIPS a partir componentes compuestos enlazando el código asociado a sus componentes internos en función de cómo éstos estén conectados entre sí.
- Se ha desarrollado un caso de estudio sencillo para validar tanto la nueva metodología propuesta como la herramienta IP-CoDER. El VIPS seleccionado para este caso de estudio es una aplicación de segmentación de caras utilizando imágenes en color.

6.3. Publicación de resultados y financiación de la investigación

Los resultados obtenidos como parte de este trabajo, realizado en el seno del grupo de investigación DSIE (División de Sistemas e Ingeniería Electrónica) de la Universidad Politécnica de Cartagena, han sido publicados en las siguientes revistas y conferencias de ámbito nacional e internacional:

➢ **Congresos internacionales con actas listadas en el *Journal Citation Reports***

- G. García, **C. Vicente**, A new model and process architecture for facial expression recognition, Proc. Joint IAPR International Workshops on Advances in Pattern Recognition, Alicante, Septiembre 2000. Lecture Notes in Computer Science, Vol. 1876, pp. 716-726 (ISI 0,390).
- G. García, **C. Vicente**, Face Detection on Still Images Using HIT Maps, Proc. 3rd International Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA'01), Halmstad (Suecia), 6-8 Junio, 2001. Lecture Notes in Computer Science, Vol. 2091, pp. 102-107 (ISI 0,415).
- **C. Vicente**, A. Toledo, P. Sánchez, Image Processing Application Development: From Rapid Prototyping to SW/HW Co-Simulation and Automated Code Generation, Proc. 2nd Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'05), Estoril (Portugal), 7-9 Junio, 2005. Lecture Notes in Computer Science, Vol. 3522, pp. 659-666 (ISI 0,51).

- A. Toledo, **C. Vicente**, J. Suardíaz, S. Cuenca, Xilinx System Generator Based HW Components for Rapid Prototyping of Computer Vision SW/HW Systems, Proc. 2nd Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'05), Estoril (Portugal), 7-9 Junio, 2005. Lecture Notes in Computer Science, Vol. 3522, pp. 667-674 (ISI 0,51).
- **C. Vicente**, A. Toledo, C. Fernandez, Heterogeneous COTS Product Integration to Allow the Comprehensive Development of Image Processing Systems, Proc. IV International Conference on COTS-Based Software Systems (ICCBSS'05), Bilbao, 7-9 February, 2005. Lecture Notes in Computer Science, Vol. 3412, pp. 8 (ISI 0,51).
- G. García, A. García, **C. Vicente**, A. Ruiz, P. E. López de Teruel, Time and Date OCR in CCTV Video, Proc. 13th International Conference on Image Analysis and Processing (ICIAP'05), Cagliari (Italia), 6-8 Septiembre, 2005. Lecture Notes in Computer Science, Vol. 3617, pp. 703-710 (ISI 0,51).

➢ Otros congresos internacionales

- C. Fernández, J. Suardíaz, C. Jiménez, **C. Vicente**, Automated Visual Inspection Application within the Industry of Preserved Vegatables, Proc. 5th International Conference on Quality Control by Artificial Vision (QCAV'01), Le Creusot (Francia), 21-23 Mayo, 2001.
- G. García, **C. Vicente**, A. García, Localización y Seguimiento de Caras Usando Búsqueda en Rejilla, Proc. Conf. Iberoamericana en Sistemas, Cibernética e Informática (CISCI'02), Orlando, Florida (USA), 19-21 Julio, 2002.

➢ Revistas nacionales

- **C. Vicente**, C. Fernandez, P. Sánchez, Desarrollo de Sistemas de Inspección Visual Automatizada a partir de la descripción de un Patrón Arquitectural Genérico, NOVATICA, No. 171, pp. 63-65, 2004.

➢ Congresos nacionales

- G. García, **C. Vicente**, A Unified Approach to Face Detection, Segmentation and Location Using HIT Maps, Proc. IX Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes (SNRFAI'01), Castellón, 14-16 Mayo, 2001.
- **C. Vicente**, G. García, A. García, Seguimiento de Caras Humanas Mediante Búsqueda Logarítmica en Rejilla, Proc. III Jornadas Regionales de Informática Gráfica (JRIG'02), Jaén, 14-15 Febrero, 2002.
- **C. Vicente**, C. Fernandez, P. Sánchez, Sistemas de Inspección Visual Automatizada: de la Arquitectura Software a la Generación de Prototipos Ejecutables, Proc. IX Jornadas de Ingeniería del Software y Bases de Datos (JISBD'04), Málaga, 11-13 Noviembre, 2004.

- **C. Vicente**, A. Toledo, C. Fernández, P. Sánchez, Generación Automática de Aplicaciones Mixtas Sw/Hw mediante la Integración de Componentes COTS, X Jornadas de Ingeniería del Software y Bases de Datos (JISBD'04), Granada, 14-16 Septiembre, 2005. Este artículo ha sido preseleccionado para su publicación en un número especial de la revista IEEE América Latina.

El trabajo realizado en esta Tesis Doctoral ha sido parcialmente financiado a través de los siguientes proyectos de investigación y contratos con empresas:

- **Proyectos de investigación financiados con cargo a fondos públicos:**

  - Título: Técnicas de Inspección Visual Automatizada para el Control de Calidad de Gajos de Mandarina (1FD 97-1606-C02-01).

    Entidad financiadora: CICYT
    Entidades participantes: UPCT, INEMUR,Centro Tecnológico del Metal
    Desde: 30/12/1999 hasta: 31/12/2001

  - Título: *Técnicas de Co-diseño para SIVAs* (TIC 2000-1765-C03-02).

    Entidad financiadora: CICYT
    Entidades participantes: UPCT, UPM, INEMUR,Centro Tecnológico del Metal
    Desde: 28/12/2000 hasta: 27/12/2003

  - Título: EFTCoR: Environmental Friendly and Cost-Effective Technology for Coating Removal (G3RD-CT-2002-00794).

    Entidad financiadora: Comisión Europea
    Entidades participantes: UPCT, UPM, IZAR, EDER, IAPETOS, LISNAVE, BYGSYS,
    Desde: 01/10/2002 hasta: 30/09/2005

- **Contratos con empresas**

  - Título: Sistema de Inspección Visual de Alta Velocidad Aplicado a la Industria de la Conserva Vegetal.

    Empresa/Administración financiadora: M.E.C. y HRS-SPIRATUBE, S.L.
    Entidades participantes: UMU , HRS-SPIRATUBE, S.L.
    Desde: 01/03/1998 hasta: 15/10/1999

  - Título: Reconocimiento de fecha y hora en imágenes de CCTV.

    Empresa/Administración financiadora: Vision Base Ltd.
    Entidades participantes: UMU, UPCT, Vision Base Ltd.
    Desde: 01/11/2003 hasta: 01/03/2003

6.4. Líneas de trabajo futuras

Con relación a los resultados obtenidos en esta Tesis, a continuación se recogen algunas de las líneas en las que se sigue trabajando en la actualidad así como otras en las que resultaría de interés seguir profundizando en el futuro.

- **En cuanto a la metodología de desarrollo de VIPS que se ha presentado:**
  - En la actualidad se trabaja en la extensión del alcance de la línea de productos LPS-VIPS a fin de incorporar algunos de los dominios que quedaron excluidos en la versión inicial de la misma como, por ejemplo, el de gestión de de bases de datos o el de virtualización.
  - Entre las cuestiones que han quedado abiertas y que resultaría interesante abordar en el futuro cabría mencionar, entre otras, la incorporación a la metodología de alguna herramienta que permita dar un soporte más riguroso a la especificación y el seguimiento de los requisitos no funcionales, así como considerar la incorporación de otros paradigmas de desarrollo de software como el orientado a aspectos o el dirigido por modelos.
- **En cuanto a la herramienta IP-CoDER:**
  - En la actualidad se trabaja en una extensión de la misma que permita particionar la funcionalidad de los prototipos que se desarrollan en módulos software y hardware, a fin de poder validarlos conjuntamente empleando para ello plataformas mixtas Hw/Sw simuladas o reales.
  - También se está trabajando en la inclusión de nuevas librerías de procesamiento de imágenes en el modelo de componentes actual.
  - A medio plazo se plantea incorporar a la herramienta un sistema de ayuda a la decisión que facilite la selección de los algoritmos que mejor se comporten para resolver el problema específico planteado por cada VIPS.
- **En cuanto a la validación de la propuesta presentada en esta Tesis:**
  - En la actualidad se trabaja en el desarrollo de nuevos casos de estudio más complejos que los abordados hasta el momento y con los que se pretende realizar una validación exhaustiva tanto de la nueva metodología de desarrollo de VIPS como de la herramienta IP-CoDER.
  - Una vez completado este proceso de validación interna se pretende revalidar la propuesta ofreciéndola a distintos usuarios externos a fin de que éstos puedan detectar posibles debilidades o mejoras previamente no identificadas.

Apéndice I

Descripción de las metodologías PuLSE™ y KobrA

En este apéndice se presentan las metodologías PuLSE™ [PULSE] y KobrA [KOBRA] ambas concebidas para el desarrollo de líneas de productos software y desarrolladas por el Fraunhofer Institute of Experimental Software Engineering (IESE). El uso combinado de estas metodologías, tal y como se propone en esta Tesis, ha demostrado excelentes resultados en muchos casos de estudio realizados en diversos dominios de aplicación [Bayer 01][Muthig 04][de Souza 05].

La elección de estas metodologías se ha basado en cuatro factores claves: (1) en ambos casos se apuesta por el uso de notaciones estándar como UML (Unified Modeling Language) [UML] o FODA (Feature-Oriented Domain Analisys) [Kang 90], (2) la metodología KobrA, empleada para describir la arquitectura de la línea LPS-VIPS, sigue un enfoque orientado a componentes que resulta compatible y fácil de integrar con el resto del trabajo desarrollado en esta Tesis, (3) al diferencia de otras, tanto PuLSE™ como KobrA tienen un carácter marcadamente práctico, y (4) ambas metodologías están clara y extensamente documentadas.

A continuación se realiza una descripción de los aspectos fundamentales de estas dos metodologías tratando de recoger, de forma esquemática, tanto los procesos que se llevan a cabo en cada una de ellas como los resultados que se obtienen al final de cada una de sus etapas.

I.1. PuLSE™ (Product Line Software Engineering™)

PuLSE™ es una metodología que permite definir, construir y mantener líneas de productos software (LPS) en una amplia variedad de contextos empresariales. Esta metodología gira en torno a tres conjuntos de elementos (ver Figura 53): el primero de ellos recoge el conjunto de etapas que comprenden el ciclo de vida PuLSE™, el segundo, un conjunto de componentes técnicos que dan soporte a cada una de estas etapas y, por último, el tercero incluye un conjunto de componentes de soporte que proporcionan información adicional al proceso de desarrollo de LPS de una cierta envergadura.

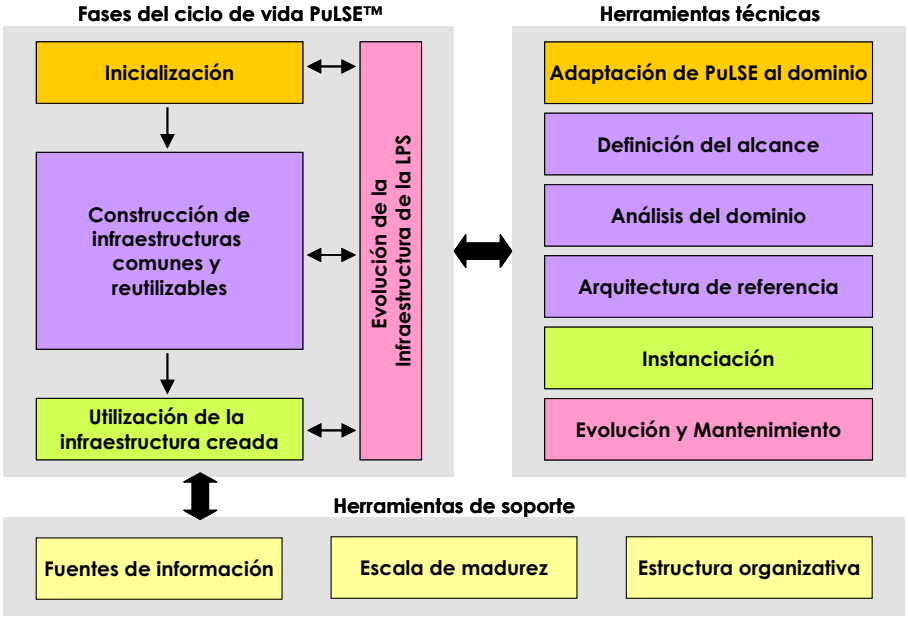


Figura 53. Vista general de PuLSE

En los siguientes apartados se describe brevemente cada uno de estos tres conjuntos de elementos.

I.1.1. Fases del ciclo de vida PuLSE™

Las fases del ciclo de vida PuLSE™ describen las distintas actividades que deben llevarse a cabo para poder inicializar, construir, utilizar y hacer evolucionar una línea de productos software. Las cuatro etapas que comprende este ciclo de vida son las siguientes:

1. **Inicialización.** Durante esta etapa se adaptan los componentes PuLSE™ al dominio específico de las aplicaciones que se van a desarrollar a partir de la LPS. Para ello se establece la estructura del proyecto, el contexto organizativo y los objetivos de reutilización que se persiguen al optar por un enfoque de LPS en lugar de por el desarrollo de uno o más sistemas individuales. La fase de inicialización se realiza utilizando el componente técnico de adaptación PuLSE™-BC (ver apartado I.1.2.1).

2. **Construcción de un núcleo de infraestructuras comunes y reutilizables:** Esta fase se corresponde con el proceso de *ingeniería de dominio* (ver Capítulo 3, Figura 21) en el que se analizan los aspectos comunes y variables de los productos que se van a derivar de la LPS y se desarrolla un núcleo de infraestructuras para su reutilización posterior (software ***para*** reutilización). Esta fase se divide a su vez en las tres etapas siguientes:

   2.1. Definición del alcance de la LPS: Durante esta etapa se acota la LPS estableciéndose qué productos podrán derivarse a partir de ella. La definición del alcance se realiza utilizando el componente técnico PuLSE™-Eco (ver apartado I.1.2.2).

   2.2. Modelado de análisis: Durante esta etapa se modelan los elementos comunes y variantes de los productos incluidos en el alcance de la LPS y se genera un mapa de las distintas opciones y alternativas de configuración por las que se puede optar a la hora de construir los distintos productos. Esta etapa se lleva a cabo utilizando el componente técnico PuLSE-CDA (ver apartado I.1.2.3).

   2.3. Definición de la arquitectura software de la LPS. En esta etapa se describe la arquitectura genérica de la LPS compartida por todos los productos que se pueden derivar a partir de ella. La definición de esta arquitectura se realiza utilizando el componente técnico PuLSE-DSSA (ver apartado I.1.2.4).

   Estos tres elementos, y en particular la arquitectura software, desempeñan un papel central en toda LPS por lo que se debe prestar una especial atención durante su definición mediante los correspondientes componentes técnicos de PuLSE™. Además, dependiendo del dominio concreto al que pertenezca una determinada LPS, puede resultar conveniente añadir al núcleo de infraestructuras reutilizables ciertos recursos adicionales como, por ejemplo, librerías de funciones o clases propias del dominio, herramientas preexistentes, etc.

3. **Construcción de nuevos productos a partir de las infraestructuras creadas.** Esta fase se corresponde con el proceso de *ingeniería de las aplicaciones* (ver Capítulo 3, Figura 21) gracias al que es posible instanciar nuevos productos a partir de las infraestructuras previamente creadas (desarrollo de software ***con*** reutilización). Este proceso de instanciación de la LPS se lleva a cabo utilizando el componente técnico PuLSE™-I (ver apartado I.1.2.5).

4. **Mantenimiento y evolución de la LPS.** Los requisitos de una LPS pueden evolucionar a lo largo del tiempo (inclusión de nuevos productos, exclusión de otros que ya no se venden, incorporación de nuevas características a los productos considerados dentro del alcance de la LPS, etc.) Durante esta fase se vigilan y capturan estas posibles variaciones en los requisitos para hacer evolucionar la LPS de modo que se adecue a ellos. Esta fase se lleva a cabo utilizando el componente PuLSE™-EM (ver apartado I.1.2.6).

A continuación se describen los componentes técnicos que dan soporte a cada una de las fases anteriores del ciclo de vida PuLSE™.

I.1.2. Componentes técnicos

La metodología PuLSE™ proporciona un conjunto de seis componentes o herramientas técnicas que facilitan el desarrollo de cada una de las seis etapas incluidas en su ciclo de vida y descritas en el apartado anterior. A continuación se enumeran y describen brevemente cada una de ellas.

I.1.2.1. PuLSE™ - BC (Baselining and Customization)

Esta herramienta permite establecer el contexto empresarial en el que se desarrolla la LPS y permite adaptar convenientemente la metodología PuLSE™ (tanto los procesos como los componentes técnicos y de soporte que éstos utilizan) para que se adecue a dicho contexto.

I.1.2.2. PuLSE™-Eco (Economic Scoping)

Este componente técnico está formado por un conjunto de herramientas que permiten identificar, describir y acotar la LPS determinando las características de los productos que se construirán a partir de ella. La definición del alcance en PuLSE™ se realiza en términos económicos, esto es, en función de los objetivos del negocio y de la planificación a corto, medio y largo plazo de la producción, datos éstos obtenidos durante la fase de inicialización utilizando el componente PuLSE™-BC. El componente PuLSE™-Eco está formado por las siguientes tres herramientas:

1. **PuLSE™-Eco-Plantilla de descripción de productos.** En esta plantilla se recoge información relativa al estado actual de desarrollo del producto, la funcionalidad que debe proporcionar, el segmento del mercado al que va dirigido, las características que lo diferencian de otros productos, etc. Se rellenará una de estas plantillas para cada producto que se desee poder derivar de la LPS.
2. **Herramientas para la identificación y descripción de las áreas funcionales (dominios) más relevantes identificadas en los productos:** la información obtenida de cada uno de los productos a través de las plantillas anteriores se utilizará como entrada para las siguientes herramientas:
   - 2.1. **PuLSE™-Eco-Tabla de características:** Se elabora una tabla en la que cada columna representa uno de los productos y cada fila una de las características (funcionalidades) identificadas a partir de su descripción (las características aparecerán agrupadas por dominios o áreas funcionales). La casilla (*c, p*) de la tabla aparecerá marcada sólo si el producto *p* ofrece la característica *c*.
   - 2.2. **PuLSE™-Eco-Plantilla de descripción de dominios.** En esta plantilla se recoge información relativa a cada uno de los dominios identificados en la

tabla de características previa: funcionalidad que ofrece y demanda el dominio, datos que maneja y/o almacena, recursos preexistentes relacionados con el domino ya sean de desarrollo propio o adquiridos a terceros, etc. Durante la descripción de estos dominios pueden surgir nuevos subdominos y/o dominios internos que también se deben documentar utilizando la misma plantilla.

2.3. PuLSE™-Eco-Mapa de la estructura de dominios: utilizando las descripciones anteriores se elabora un diagrama en el que se recogen las relaciones de dependencia y/o jerarquía existentes entre los distintos dominios.

2.4. PuLSE™-Eco-Mapa inicial de productos: Este mapa recoge la funcionalidad tanto externa como interna ofrecida por los distintos productos pudiendo considerarse una extensión reordenada de la tabla de características 0.

3. Herramientas de valoración y selección de dominios: la información obtenida como resultado de la descripción previa de los dominios, junto con la información proporcionada por expertos en cada uno de ellos, se empleará como entrada para las siguientes herramientas:

3.1. PuLSE™-Eco-Estudio de la relevancia: Tabla en la se recoge una valoración de la relevancia y aportación funcional de cada uno de los dominios al desarrollo de los distintos productos.

3.2. PuLSE™-Eco-Estudio del potencial de reutilización: Tabla que recoge las valoraciones relativas al potencial de reutilización de cada uno de los dominios en el contexto de la LPS.

3.3. PuLSE™-Eco-Selección de dominios: Herramienta para evaluar el coste/beneficio de incorporar la funcionalidad ofrecida por cada uno de los dominios al núcleo de recursos comunes de la LPS. Los criterios de evaluación y selección de dominios se establecen de acuerdo a los resultados obtenidos durante la fase de inicialización utilizando la herramienta PuLSE™-BC.

Como resultado final del uso de todas las herramientas contenidas en el componente técnico PuLSE™-Eco se obtiene la descripción del alcance de la LPS consistente en una lista de los dominios seleccionados por su relevancia y potencial de reutilización para formar parte del núcleo de recursos comunes.

I.1.2.3. PuLSE™-CDA (Customizable Domain Analysis)

Este componente técnico está formado por un conjunto de herramientas que permiten modelar los dominios previamente seleccionados durante la fase de definición del alcance (PuLSE™-Eco). La descripción del modelo de análisis de una LPS requiere utilizar herramientas que permitan describir la variabilidad inherente a las mismas por lo que, en muchos casos, las herramientas tradicionales de análisis deben adaptarse o extenderse para incluir este aspecto.

A continuación se describen las tres herramientas que constituyen el componente técnico PuLSE™-CDA:

1. **Modelos de Casos de Uso (CU).** Cada Caso de Uso representa una de las características o funcionalidades que ofrece un sistema a quien lo usa. En el caso de las LPS deben hacerse dos consideraciones especiales: (1) no todos los productos de la línea ofrecen las mismas características (algunas sólo están disponibles en ciertos productos), y (2) una misma funcionalidad puede implementarse de distinto modo en cada uno de los productos. Esta variabilidad deberá haber quedado recogida en las descripciones previas de los dominios por lo que éstas servirán para identificar y modelar los CU de la LPS. Para ello se utilizarán las dos herramientas siguientes:
   - 1.1. **PuLSE™-CDA-Diagrama de CU:** esta herramienta permite representar gráficamente los distintos CU identificados en las descripciones de los dominios, así como las relaciones que éstos establecen entre sí y con los actores externos del sistema. Para elaborar este diagrama PuLSE™ propone utilizar la notación UML, extendida de modo que pueda representarse la variabilidad en los distintos elementos del diagrama (CU y actores externos e internos).
   - 1.2. **PuLSE™-CDA-Plantilla para la descripción de CU:** Cada CU identificado dentro de los distintos dominios se documentará utilizando una plantilla en la que se recogerá, entre otra, la siguiente información: secuencia de eventos y acciones que se llevan a cabo durante una ejecución normal del CU (incluyendo las opciones y alternativas de los distintos productos), las posibles alteraciones de esta secuencia como consecuencia de algún error, las pre- y post condiciones asociadas al CU, etc.
2. **PuLSE™-CDA-Diagrama de variabilidad:** Esta herramienta permite representar gráficamente el conjunto de elementos variantes identificados en cada CU así como las relaciones existentes ente ellos. Para la elaboración de estos diagramas existen diversas notaciones cuyo uso está más o menos extendido: FODA [Kang 90], FORM [Kang 98], FeatuRSEB [Griss 98], etc. Una comparativa de varias de estas notaciones puede encontrarse en el artículo de J. C. Trigaux [Trigaux 03].
3. **PuLSE™-CDA-Tabla de decisión:** utilizando los resultados obtenidos con las dos herramientas anteriores se elaborará una tabla de decisión en la que se recogerán: (1) las distintas alternativas posibles para cada uno de los elementos modelados como variantes y (2) para cada una estas alternativas, qué consecuencias tiene su elección en el modelo de análisis global, esto es, qué otras alternativas deben incluirse o desaparecer como consecuencia de dicha elección.

Como resultado final del uso de todas las herramientas contenidas en el componente técnico PuLSE™-CDA se obtiene un modelo de variabilidad en el que cada elemento identificado como variante se conecta con una entrada en la tabla de decisión. De este modo, cuando se desee instanciar un nuevo producto a partir de

la LPS, bastará con seleccionar las alternativas que éste debe incorporar para obtener así una instancia del modelo de análisis para dicho producto.

I.1.2.4. PuLSE™-DSSA (Domain Specific Software Architecture)

Esta herramienta permite especificar la arquitectura de referencia[27] de la LPS mediante la elaboración de varios modelos en los que se recogen distintas vistas arquitecturales de la LPS. Cada uno de estos modelos recogerá una serie de componentes (los relevantes para la vista que ofrece el modelo) así como de conectores a través de los que éstos se comunican entre sí.

Como los modelos generados durante la fase de análisis (PuLSE™-CDA), los modelos arquitecturales deben permitir también expresar la variabilidad inherente a las LPS. De hecho, además de incorporar la variabilidad identificada en las fases previas de la metodología, estos modelos pueden representar ciertos aspectos de variabilidad independientes del dominio y que por lo tanto no hayan sido identificadas previamente.

Para la especificación de los distintos modelos o vistas de la arquitectura pueden emplearse diversas herramientas gráficas y/o textuales, aunque quizá las más sencillas de integrar con PuLSE™ sean las proporcionadas por la metodología KobrA dado su origen común[28]. La metodología KobrA se describe en detalle en siguiente apartado de este apéndice.

I.1.2.5. PuLSE™-I (Instantiation)

Esta herramienta permite especificar, construir y validad distintos productos a partir de los resultados obtenidos en las fases previas de la metodología. En particular, la construcción de cada nuevo producto requiere instanciar el modelo de análisis y la arquitectura de referencia previamente definidos. En ambos casos deberá seleccionarse, para cada uno elementos modelado como variantes (CU/actores en el modelo de análisis, componentes/conectores en el modelo arquitectural), aquella opción o alternativa que mejor se ajuste a los requisitos del producto.

Adicionalmente, puede resultar necesario desarrollar ciertas infraestructuras no incluidas hasta este momento en el núcleo de recursos comunes. En este caso, sobre todo cuando se trata de desarrollos de una cierta envergadura y que por lo tanto suponen una inversión considerable, se debe tratar de desarrollar soluciones que sean lo mas reutilizables posible. De este modo, cada vez que estas soluciones se incorporen a un nuevo producto se recuperará parte de la inversión realizada.

I.1.2.6. PuLSE™-EM (Evolution and Management)

Esta herramienta guía y da soporte al resto de componentes técnicos de PuLSE™ centrándose en tres aspectos fundamentales:

---

[27] Una arquitectura de referencia es una arquitectura genérica diseñada para una familia de sistemas o un dominio específicos [Hofmeister 00].

[28] Tanto PuLSE™ como KobrA han sido desarrolladas en el Fraunhofer Institute of Experimental Software Engineering (IESE) por el equipo de investigación liderado por los doctores C. Atkinson y D. Muthig.

1. **Aspectos de gestión:** este componente es el encargado de programar y coordinar el uso del resto de componentes técnicos, así como de recoger información del entorno (tanto interna como externa a la organización) a fin de anticipar, en la medida de lo posible, cualquier fallo o situación de cambio que pudiera afectar a la LPS.
2. **Aspectos de evolución:** el componente PuLSE™-EM también se encarga de identificar los posibles cambios en los requisitos de la LPS y analizar su impacto en el resto de los componentes técnicos.
3. **Aspectos de aprendizaje:** este componente es el responsable de analizar los cambios que sufre la LPS en el tiempo. Este análisis permite identificar patrones de evolución que permiten anticipar los posibles problemas y necesidades futuras de la LPS.

Adicionalmente, el componente PuLSE™-EM es el encargado de mantener y dar soporte a toda la infraestructura de la LPS, así como de gestionar la configuración e instanciación de los distintos productos.

I.1.3. Componentes de soporte

Los componentes de soporte proporcionan directrices de ayuda a los distintos componentes técnicos por lo que pueden considerarse herramientas de apoyo transversales. Existen tres tipos de componentes de soporte:

1. **Puntos de entrada de información:** las fuentes de información de las que se nutre la LPS pueden ser externas (estudios de mercado, estudios de la competencia, etc.) o internas (análisis de productividad, información relativa a productos desarrollados previamente, etc.). Para cada fuente de información se creará un punto de entrada que controlará la vigencia y la relevancia de dicha información y la hará llegar a aquellos componentes que la soliciten.
2. **Escala de madurez**: herramienta utilizada para evaluar la calidad tanto del proceso como de los resultados obtenidos durante la aplicación de la metodología PuLSE™ a fin de identificar y mejorar los puntos débiles detectados. Los niveles que se definen en esta escala son: iniciado, definido, controlado y optimizado.
3. **Estructura organizativa:** esta herramienta es la encargada estudiar si la estructura organizativa en la que se sustenta la LPS (personas asignadas a cada unidad de trabajo, unidades de trabajo que se encargan de cada una de las tarea, distribución geográfica del personal en distintas oficinas, etc.) es la más adecuada y eficiente y, en el caso de no ser así, proponer los cambios necesarios.

Hasta aquí la descripción de la metodología PuLSE™. A continuación se presenta la metodología KobrA que, en el marco de esta tesis doctoral, se utilizará como una de las herramienta del componente técnico PuLSE™-DSSA para describir la arquitectura software de la línea LPS-VIPS.

I.2. La metodología KobrA

La metodología KobrA aglutina varias tecnologías avanzadas de Ingeniería del software entre las que se incluyen el desarrollo de Líneas de Productos Software (LPS), el Desarrollo de Software Basado en Componentes (DSBC), el diseño de *frameworks* y de arquitecturas software, el uso de métricas de calidad, o el modelado de procesos. Frente a otras metodologías de desarrollo de LPS KobrA pretende, ante todo, resultar práctica y precisa.

El proceso que sigue KobrA para definir una LPS se divide en dos etapas:

- El proceso de **ingeniería del dominio** permite definir un marco genérico de desarrollo en el que se recogen todas las posibles variantes dentro de una familia de productos (incluyendo información sobre sus características comunes y diferenciales).
- El proceso de **ingeniería de las aplicaciones** permite instanciar este marco de desarrollo para crear distintos productos (variantes particulares dentro de la familia de productos), cada uno de ellos diseñado para satisfacer las necesidades específicas de los distintos clientes. Así, un dominio puede instanciarse muchas veces dando lugar a distintas aplicaciones.

Es importante destacar que lo que distingue estos dos procesos es el nivel de generalidad/especificidad, no el nivel de detalle con el que se describen los distintos modelos. De hecho, los distintos elementos del sistema se describen, en ambos casos, utilizando una mezcla de modelos textuales y gráficos (basados en UML). Así pues, la diferencia entre ambos radica en que los modelos que se elaboran durante el proceso de ingeniería del dominio contienen un cierto grado de variabilidad mientras que los generados durante el proceso de ingeniería de aplicación no. La ventaja del uso de UML como notación gráfica para especificar los componentes tanto del dominio como de las aplicaciones es, por una parte, lo extendido de su uso, y por otra, su independencia de cualquier lenguaje de programación o tecnología de componentes. De hecho, la traducción de las especificaciones UML a código ejecutable se realiza de forma independiente (ortogonal) a las actividades relacionadas con la ingeniería de dominio y la ingeniería de las aplicaciones.

Uno de los principios fundamentales de KobrA es la estricta distinción que se establece entre productos y procesos. Los productos de un proyecto (modelos, casos de prueba, módulos de código, etc.) se definen de forma independiente y con anterioridad a los procesos que los utilizan. Más aún, todos los productos se organizan y se orientan en torno a la descripción de componentes individuales. Esto significa que, en lo posible, todos los productos (y sus correspondientes procesos) se definen para contener información sólo relacionada con el componente en el que se ubican. De este modo se consigue maximizar la cohesión entre los productos y los procesos definidos dentro de cada componente a la vez que se minimiza el acoplamiento entre los distintos componentes. Gracias a esta cualidad los componentes desarrollados resultan fácilmente intercambiables y pueden reutilizarse de manera individual.

Desde la perspectiva de las LPS, el método KobrA representa una adaptación orientada a objetos de la metodología PuLSE™. De hecho, existe una correspondencia directa entre los procesos que llevan a cabo ambas metodologías: (1) la fase de desarrollo del núcleo de infraestructuras comunes y reutilizables de PuLSE™ se corresponde con el proceso de ingeniería del dominio en KobrA, (2) la fase de utilización de esta infraestructura para crear nuevos productos se corresponde con el proceso de ingeniería de aplicaciones, y (3) la fase de evolución PuLSE™ se corresponde con el proceso de mantenimiento del marco de desarrollo y de las aplicaciones. En el siguiente apartado se describen más detalladamente cada uno de los procesos definidos en KobrA, así como las herramientas que utilizan y los resultados que producen.

I.2.1. Proceso de ingeniería del dominio

Para KobrA un dominio es la representación estática de una serie de Komponentes[29] organizados en forma de árbol. Cada Komponente se describe a dos niveles de abstracción: (1) a nivel de especificación se define la funcionalidad que ofrece el Komponente visto desde fuera (contrato que debe cumplir), y (2) a nivel de definición se describe cómo el Komponente cumple dicho contrato gracias a sus Komponentes internos. Cada uno de estos Komponentes internos debe a su vez especificarse y definirse lo que dará lugar a una estructura de árbol.

El proceso de ingeniería del dominio en KobrA comienza a partir de la definición del contexto de la aplicación que se quiere desarrollar y que constituye la raíz del árbol de Komponentes. A partir de esta definición comienza un proceso recursivo de especificación/definición de los componentes internos que concluye cuando se alcanza el máximo nivel de descomposición posible, esto es, cuando todos los Komponentes internos están completamente definidos (el proceso de definición no genera nuevos Komponentes internos).

Como ya se comentó previamente, tanto al especificar como al definir los distintos Komponentes pueden identificarse diversas fuentes de variabilidad que deben quedar convenientemente documentadas. Para ello, y siguiendo la misma filosofía que la empleada durante el análisis del dominio con PuLSE™ (PuLSE™-CDA), KobrA recurre al uso de tablas de decisión que asocia a las especificaciones y/o a las definiciones de aquellos Komponentes que presentan un cierto grado de variabilidad. Estas tablas recogen las distintas alternativas posibles así como las consecuencias que la elección de una u otra tienen en el diseño del Komponente en cuestión[30].

El árbol que se genera durante las sucesivas especificaciones y definiciones de los distintos Komponentes, incluyendo las tablas de decisión que éstas puedan tener asociadas, constituye el marco de desarrollo genérico que se empleará para generar los distintos productos durante el proceso de ingeniería de las aplicaciones (*framework* reutilizable).

---

[29] Komponent se utiliza como abreviatura de "KobrA component"

[30] El diseño del Komponente se verá afectado a nivel externo y/o interno en función de si la tabla de decisión está asociada, por este orden, a la especificación y/o a la definición del mismo.

En los apartados siguientes se detallan los tres subprocesos que comprenden el proceso de ingeniería del dominio en KobrA: definición del contexto y especificación y definición de los Komponentes.

I.2.1.1. Definición del contexto

La ingeniería del dominio comienza con la extracción de las propiedades de la familia de sistemas que se desea desarrollar incluyendo la definición del alcance de la LPS. Esta etapa tiene como equivalente en PuLSE™ la definición del alcance implementada por el componente técnico PuLSE™-Eco.

I.2.1.2. Especificación de los Komponentes

El objetivo del proceso de especificación de los Komponentes es crear una serie de modelos que, de manera conjunta, permitan describir las propiedades externamente visibles de los mismos. Como tal, la especificación puede considerarse como la definición de la interfaz de un Komponente y la descripción de los servicios que éste ofrece.

El proceso de especificación de un Komponente en KobrA consiste en modelar tres de sus aspectos fundamentales: su estructura, su funcionalidad y su comportamiento. Además, como ya se comentó con anterioridad, si el Komponente presenta algún rasgo variable también deberá elaborarse un modelo de decisión.

1. **El modelo estructural** describe las clases y las relaciones mediante las que un Komponente interactúa con su entorno, así como cualquier estructura interna del Komponente que sea visible en su interfaz. El modelo estructural se compone de diagramas UML de clases y de objetos. Los diagramas de clases definen las clases, sus atributos y los servicios que proporcionan (funcionalidad externamente visible) ilustrando la interacción de los Komponentes con su entorno. Los diagramas de objetos se necesitan únicamente si el Komponente a especificar contiene componentes de caja blanca. Si este es el caso, el propósito de estos diagramas es describir las partes de la estructura interna que son visibles desde el exterior.
2. **El modelo de comportamiento** describe cómo responde un Komponente a un estímulo externo. Consiste en un número arbitrario de diagramas de estado UML y un mapa de eventos ocasionales. El diagrama de estados en una especificación KobrA describe los estados del componente que son visibles para el usuario, así como los cambios que se producen en ellos como consecuencia de ciertos eventos (también visibles para el usuario). Cada evento representa los requisitos que deben cumplirse para ejecutar una determinada operación de las definidas en el diagrama de clases del Komponente.
3. **El modelo funcional** de un Komponente KobrA consiste en una serie de esquemas de operación[31] que describen los efectos externamente visibles de las operaciones que éste ofrece. Cada una de las operaciones listadas en el

[31] Los esquemas de operación no forman parte de la notación UML sino que tienen su origen en el método de desarrollo de software orientado a objetos denominado Fusion [FUSION].

diagrama de clases debe tener su correspondiente esquema de operación en el que se definirán: (1) sus parámetros de entrada, (2) las variables a las que accede y aquellas que modifica, (3) los valores que se producen como resultado de su ejecución, y (4) sus pre-condiciones y post-condiciones.

4. **El modelo de decisión** asociado a un Komponente describe todas sus distintas variantes y constituye una extensión del modelo de decisión de su Komponente padre.

I.2.1.3. Definición de los Komponentes

El objetivo del proceso de definición de un Komponente es crear una serie de modelos que de manera conjunta describan su diseño interno, esto es, qué sub-Komponentes lo forman y como colaboran éstos para conseguir implementar la funcionalidad externa recogida en la especificación previa del mismo. Como en el caso de la especificación, la definición de un Komponente en KobrA requiere elaborar tres modelos: el de interacción, el de actividad, y el estructural. También como en el caso anterior, si el diseño del Komponente presenta algún rasgo variante deberá añadirse un cuarto modelo de decisión.

1. **El modelo de interacción** de un Komponente consiste en un conjunto de diagramas de interacción UML (ya sean de colaboración o de secuencia), uno para cada una de las operaciones identificadas en el modelo funcional de la especificación (esquemas de operación).
2. **El modelo de actividad** de un Komponente consiste en un conjunto de diagramas de actividad UML, uno para cada operación especificada. Este modelo puede considerarse un modelo intermedio entre el modelo funcional de la especificación y el modelo de interacción anterior.
3. **El modelo estructural** describe las clases y las relaciones que existen entre ellas conformando la arquitectura interna del Komponente. Como en el caso del modelo estructural elaborado durante la especificación, en este caso también se definen diagramas de clases y de objetos UML. Los diagramas de clases son extensiones de los elaborados durante la especificación, esto es, cada una de las clases especificadas se define con un mayor nivel de detalle siendo además posible incluir nuevos elementos (a menudo sub-Komponentes) identificados durante la creación del modelo de interacción. Los diagramas de objeto, a nivel de definición, suministran una instantánea de una configuración típica de los objetos internos del Komponente.

Otra posible forma de definir la especificación de un Komponente es mediante la reutilización de alguno de los componentes previamente desarrollados o adquiridos a terceros (por ejemplo, componentes de tipo COTS). Para ello, suele resultar necesario adaptar (al menos parcialmente) la interfaz especificada para el Komponente de modo que se ajuste a la del componente que se desea incorporar. Lo contrario, esto es, adaptar el componente preexistente a la especificación, puede resultar imposible si éste es de tipo caja negra como ocurre en la mayoría de los COTS.

I.2.2. Proceso de ingeniería de las aplicaciones

El proceso de ingeniería de las aplicaciones toma como punto de partida el marco de desarrollo (*framework*) construido durante el proceso de ingeniería del dominio (árbol de komponentes con sus correspondientes definiciones, especificaciones y modelos de decisión) y deriva a partir de él distintas aplicaciones específicas. Este proceso de instanciación del *framework* parte de la especificación de los requisitos de la aplicación concreta que se desea desarrollar y, a partir de ellos, resuelve de manera recursiva y descendente (desde la raíz del árbol hasta los komponentes de más bajo nivel) los correspondientes modelos de decisión. Como resultado, los modelos genéricos y variantes del *framework* se transforman en modelos concretos y sin variabilidad.

El proceso de ingeniería de las aplicaciones se divide en las dos etapas: instanciación del contexto e instanciación del *framework*, cada una de las cuales se comenta brevemente a continuación.

I.2.2.1. Instanciación del contexto de la LPS

El contexto de la LPS se define (de manera genérica) durante el proceso de ingeniería del dominio (ver 0) y constituye la raíz del árbol de Komponentes que se definen y especifican a continuación a partir de él. La instanciación de este contexto genérico comienza cuando un cliente potencial manifiesta su interés por adquirir un producto perteneciente al dominio de la LPS. De manera ideal, un consultor experto tanto en el dominio de la aplicación como en el *framework* de desarrollo será el encargado de gestionar la interacción con el cliente, identificando sus necesidades y capturando los requisitos del producto que éste demanda.

El proceso de extracción de requisitos estará guiado en todo momento por el modelo de decisión asociado al Komponente raíz (contexto genérico), de modo que será el consultor el encargado de ofrecer al cliente las distintas alternativas soportadas por la LPS para que éste seleccione aquella que mejor se ajuste a sus necesidades. De este modo, se facilita enormemente el proceso de captura de los requisitos tanto para el cliente – a quien se le ofrece un conjunto de soluciones previamente elaboradas entre las que poder elegir – como para el consultor – que adquiere un rol de guía para el cliente durante todo el proceso.

Cuando ninguna de las alternativas soportadas por el *framework* satisfaga suficientemente las necesidades del cliente se seleccionará aquella que mejor se aproxime a sus requisitos y se especificarán aquellos aspectos que el usuario desea que se añadan, eliminen o modifiquen de la misma. Esto permitirá seguir reutilizando el *framework* de desarrollo, adaptando sólo aquellos Komponentes afectados por la introducción de la nueva variante en el modelo del contexto.

En el peor de los casos podría ocurrir que los requisitos demandados por el cliente resultasen incompatibles con los establecidos durante el análisis del domino. En este caso resultaría imposible reutilizar el *framework* de desarrollo predefinido y debería construirse una solución totalmente a medida y considerablemente más costosa. El consultor deberá entonces tratar de buscar con el cliente una manera alternativa de

formular estos requisitos de manera que resulten compatibles con los del *framework* a fin de encontrar una solución más sencilla y económica. En cualquier caso, la aparición de nuevos requisitos – sobre todo cuando éstos tienen un impacto considerable en el proceso de ingeniería de las aplicaciones y/o son varios los clientes que los demandan – sugiere la necesidad de repetir el proceso de ingeniería del dominio para modificar el *framework* de desarrollo de modo que incluya los nuevos requisitos.

Una vez terminado el proceso de toma de decisiones la captura de requisitos se da por concluida. El resultado es una instancia del modelo del contexto (Komponente raíz del árbol) en el que quedan reflejados los requisitos particulares de la aplicación que se va a desarrollar. Este resultado puede utilizarse a modo de contrato con el cliente y, una vez aprobado por éste, será el punto de partida para instanciar el resto del *framework* de la LPS, tal y como se detalla a continuación en el siguiente apartado.

I.2.2.2. Instanciación del framework de la LPS

Una vez instanciado el contexto de la aplicación (raíz del árbol de Komponentes) debe hacerse lo propio con el resto del *framework* definido durante el proceso de ingeniería del dominio, esto es, deben instanciarse tanto las definiciones como las especificaciones de cada uno de los Komponentes internos (resto de nodos del árbol). Para ello, las alternativas seleccionadas por el cliente durante la instanciación del contexto se propagan a lo largo de los modelos de decisión asociados a los distintos Komponentes, debiendo asegurarse que se mantiene la consistencia tanto horizontal (consistencia entre la definición y la especificación de un mismo Komponente), como en vertical (consistencia entre cada Komponente y sus Komponentes internos).

Como resultado de este proceso se obtiene una jerarquía parcialmente instanciada en la que por lo general quedan ciertos puntos de variabilidad por resolver. Estas variantes no resueltas suelen estar relacionadas con ciertos detalles relativos al diseño de la aplicación (no planteados al cliente durante la captura de los requisitos por tratarse de cuestiones de muy bajo nivel), o bien con los nuevos requisitos impuestos por el cliente y que inicialmente no estaban contemplados en el *framework*. En cualquier caso, el conjunto de variantes sin resolver se envía al consultor responsable del proyecto quien será el encargado de tomar las decisiones que considere más oportunas ya sea individualmente, junto a su equipo de desarrolladores y/o en colaboración con el cliente.

El proceso de instanciación del *framework* concluye cuando todos los puntos de variabilidad quedan resueltos, los requisitos del cliente han sido integrados en la solución, y la aplicación resultante pasa con éxito todos los controles de calidad tanto de los desarrolladores como del cliente.

Apéndice II

Definición de la Línea LPS-VIPS: Resultados Adicionales

Este apéndice recoge parte de las resultados obtenidos al aplicar la metodología PuLSE™ para definir la línea LPS-VIPS, en particular, las descripciones de los dominios identificados durante la fase PuLSE™-Eco (ver Capítulo 4, apartado 4.2.2) y las descripciones de los Casos de Uso (CU) del modelo de análisis correspondiente a la fase PuLSE™-CDA (ver Capítulo 4, apartado 4.3.1).

II.1. Descripción de los dominios relevantes (PuLSE™-Eco)

En este apartado se recogen las descripciones detalladas de los dominios identificados durante la fase Eco de la metodología PuLSE™ (ver Capítulo 4, apartado 4.2.2). La plantilla que se empleará para describir estos dominios es la que se muestra a continuación en el cuadro de la Figura 54.

P1 - Nombre: Nombre del dominio
P2 - Descripción: Breve descripción.
P3 - Funcionalidad ofrecida y demandada por el dominio:
- **Entradas** (funcionalidad demandada)
- **Salidas** (funcionalidad ofrecida)

P4 - Dominios de orden superior: dominios de los que depende.
P5 - Dominios de orden inferior: dominios que dependen de éste.
P6 - Subdominios:
P7 - Funcionalidad proporcionadas por el dominio:
F1 Funcionalidad_1
… …
Fn Funcionalidad_N
P8 - Datos manejados/almacenados en el dominio:
- **Datos de entrada:** datos que recibe el dominio.
- **Datos de salida:** datos que envía el dominio.
- **Datos almacenados:** datos almacenados en el dominio.

P9 - Recursos preexistentes.
P10 - Productos en los que se implementará la funcionalidad de este dominio.

Figura 54. Plantilla para la descripción de dominios.

A continuación se recogen las especificaciones de los dominios identificados en las distintas familias de VIPS así como las de los nuevos dominios y subdominios que surjan de dichas especificaciones.

► ***Descripción del dominio de gestión de la operativa básica del VIPS***

P1 - Nombre: Gestión de operativa básica.
P2 - Descripción: Proporciona la funcionalidad básica del VIPS (arranque y parada del sistema, calibración, inicio y detención del proceso, etc.).
P3 - Funcionalidad ofrecida y demandada por el dominio:
- **Entradas:**
  - Desde la interfaz de usuario se reciben los distintos comandos asociados al menú de operativa básica del VIPS.
  - Desde el *gestor central del VIPS* (dominio interno) se reciben los resultados de los procesos internos: procesamiento de la información visual, control, comunicaciones y alarmas.
- **Salidas:**
  - Al *gestor central del VIPS* se le envían las órdenes recibidas del usuario para que éste las atienda convenientemente (por ejemplo, cuando el estado del sistema lo permita).
  - A la interfaz de usuario se le envían los resultados de los procesos internos y las alarmas. En función de la configuración de la aplicación y/o de los permisos que tenga el usuario activo algunos de estos resultados y/o alarmas pueden no enviarse (filtrado).

P4 - Dominios de orden superior (de los que éste depende): Interfaz de usuario.

P5 - Dominios de orden inferior (que dependen de éste): Gestor central del VIPS.

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

F1 Identificar usuario: verificar la identidad de un usuario que trata de acceder al sistema; si la verificación es correcta establecer la configuración y los permisos correspondientes a dicho usuario.

F2 Arrancar VIPS: se inicializa el sistema cargando los parámetros de configuración actuales y se envía orden de arranque al gestor central del VIPS para que inicialice los dispositivos externos (sensores, actuadores, controladores, dispositivos de comunicaciones, etc.).

F3 Parar VIPS: se envía orden de parada al gestor central del VIPS para que detenga los dispositivos externos; una vez recibida la confirmación de que se ha completado esta operación se finaliza la aplicación.

F4 Iniciar proceso: se envía orden al gestor central del VIPS de que comience el procesamiento de la información visual; mientras no se detenga este proceso se visualizarán los resultados seleccionados por el usuario a medida que vayan llegando desde el gestor central del VIPS.

F5 Detener proceso: se envía orden al gestor central del VIPS de que detenga el procesamiento de la información visual.

F6 Calibrar sistema: se envía orden al gestor central del VIPS para que se realice la calibración de los sensores y/o actuadores del sistema.

F7 Iniciar alarma de usuario: Si el usuario detecta cualquier situación anómala podrá disparar en cualquier momento una señal de alarma.

F8 Detener alarma: El usuario detiene la alarma activa (notificada por el sistema o por otro usuario) de lo que se informa al gestor central del VIPS.

F9 Configurar proceso: en función de sus permisos el usuario puede modificar ciertos parámetros del VIPS tanto externos (por ejemplo, resultados que desea visualizar) como internos (por ejemplo, parámetros de control, comunicaciones, etc.).

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  - Desde la interfaz de usuario: parámetros de configuración del VIPS (internos y externos).
  - Desde el *gestor central del VIPS*: resultados de las solicitudes enviadas (información solicitada a los gestores de: informes, bases de datos, procesamiento visual, control y comunicaciones).
  - Desde el *gestor de alarmas*: notificaciones de alarmas activadas.
- **Datos de salida:**
  - A la interfaz de usuario: resultados solicitados (correctamente formateados) y notificaciones de alarmas activadas.
  - Al *gestor central del VIPS*: parámetros de configuración del VIPS (internos y externos) y notificación de las alarmas de usuario.
- **Datos almacenados:** usuario activo y nivel de permisos.

P9 - Recursos preexistentes: ---

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1, F_8 y F_9 con frecuencia en todos los VIPS.
- F_2, F_3 y F_7 en los SIVA y los SNAV; ocasionalmente también en los SIMED.
- F_4 en todos los VIPS.
- F_5 con frecuencia en los SIVA, SNAV e IPERC; ocasionalmente también en el resto de los VIPS.
- F_6 en todos los VIPS salvo en los SIG donde aparece sólo ocasionalmente.

► *Descripción del dominio de gestión de informes*

P1 - Nombre: Gestión de informes.

P2 - Descripción: Proporciona la funcionalidad necesaria para crear, mostrar, imprimir y configurar distintos tipos de informes.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde la interfaz de usuario se reciben los distintos comandos asociados al menú de gestión de informes.
  - Desde el *gestor central del VIPS* se reciben los datos necesarios para rellenar el informe solicitado.
- **Salidas:**
  - A la interfaz de usuario se le envía el informe solicitado correctamente formateado para su visualización.
  - Cuando se solicite la impresión de un informe éste se enviará, correctamente formateado, a la impresora seleccionada por el usuario o a la configurada por defecto.
  - Cuando se solicite guardar un informe éste se almacenará, correctamente formateado, en el dispositivo y directorio seleccionados por el usuario o configurados por defecto.

P4 - Dominios de orden superior: Interfaz de usuario.

P5 - Dominios de orden inferior: *gestor central del VIPS*.

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

Importante: los informes pueden ser puntuales (obtenidos en el momento de la solicitud) o continuos (datos almacenados a modo de histórico).

F1 Generar informe puntual: se genera un informe puntual con los datos seleccionados por el usuario (ver configurar informes) solicitándoselos al *gestor central del VIPS*. Una vez recibidos se formatean según la plantilla de formato seleccionada (ver configurar informes) y se devuelve el informe para su visualización, impresión o almacenamiento.

F2 Generar informe continuo: se generará un informe en el que se irán visualizando y/o almacenando en fichero los datos seleccionados por el usuario, solicitándoselos al *gestor central del VIPS* periódicamente. El proceso terminará por expresa petición del usuario o cuando expire un plazo de tiempo previamente establecido (ver configurar informes).

F3 **Imprimir informe**: el informe activo en la interfaz de usuario, una vez formateado correctamente, se envía a la impresora seleccionada por el usuario o a la configurada por defecto.

F4 **Guardar informe**: una copia del informe activo en la interfaz de usuario, una vez formateado adecuadamente, se almacena en el dispositivo y ruta seleccionados por el usuario o configurados por defecto.

F5 **Cargar informe**: se carga, formatea y activa el informe almacenado en el dispositivo y ruta seleccionados por el usuario o configurados por defecto.

F6 **Configurar informes**: esta opción permitirá al usuario (1) seleccionar los campos que desea que contengan sus informes, (2) crear y seleccionar distintas plantillas de formato para los mismos, (3) seleccionar una impresora donde imprimirlos, (4) seleccionar el dispositivo de almacenamiento y la ruta donde guardar/cargar los informes, y (5) establecer la duración temporal de un informe continuo.

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  - Desde la interfaz de usuario: parámetros de configuración de informes.
  - Desde el *gestor central del VIPS*: datos solicitados para el informe.
  - Desde un dispositivo de almacenamiento: informes previamente creados y almacenados.
- **Datos de salida:**
  - A la interfaz de usuario, impresora o dispositivo de almacenamiento: informe correctamente formateado.
  - Al *gestor central del VIPS*: lista de los datos que se le solicitan.
- **Datos almacenados:** parámetros de configuración de los informes.

P9 - Recursos preexistentes: existen diversas herramientas de gestión documental y de generación de informes; actualmente se carece de experiencia en su manejo.

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1, F_3, F_4 y F_5 con frecuencia en todos los VIPS.
- F_2 con frecuencia en los SIVA, los SNAV y los IPERC; ocasionalmente también en el resto de VIPS.
- F_5 ocasionalmente en los SIVA, los SBIO, los SIMED y los SIG.

► ***Descripción del dominio de gestión de bases de datos***

P1 - Nombre: Gestión de bases de datos.

P2 - Descripción: Proporciona toda la funcionalidad relacionada con el acceso, modificación y configuración de bases de datos.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde la interfaz de usuario se reciben los distintos comandos asociados al menú de gestión de bases de datos.
  - Desde el *gestor central del VIPS* se reciben:
    - Consultas para localizar un determinado registro en la base de datos.
    - Solicitudes para añadir, eliminar o modificar alguno de los registros.
- **Salidas:**
  - Al la interfaz de usuario y al *gestor central del VIPS* se envían los registros solicitados o una notificación de error si fue imposible localizarlos o acceder a la base de datos.

P4 - Dominios de orden superior: Interfaz de usuario.

P5 - Dominios de orden inferior: *gestor central del VIPS*

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

Importante: las bases de datos pueden ser textuales y/o de imágenes.

F1 **Buscar registro**: se localiza y devuelve un registro en la base de datos utilizando los criterios de búsqueda establecidos.

F2 **Crear nuevo registro**: se crea un registro a partir de los datos proporcionados, bien por el usuario a través de la interfaz de la aplicación, o bien por el *gestor central del VIPS*.

F3 **Modificar registro**: tras una operación de búsqueda de un registro (F_1) el usuario puede, en función de sus permisos, modificar algunos de los campos; una vez terminada la modificación se almacena el registro en la base de datos reemplazando al anterior.

F4 **Eliminar registro**: tras realizar la búsqueda de un registro (F_1) el usuario puede, en función de sus permisos, eliminarlo de la base de datos.

F5 **Iniciar alarma de acceso a la base de datos:** Cualquier fallo en alguna de las operaciones de acceso a las bases de datos generará una alarma que se hará llegar a quien solicitó dicho acceso, esto es, al usuario o al *gestor central del VIPS*.

F6 **Configurar acceso a base de datos**: A petición del usuario y en función de sus permisos, se le permite modificar algunos de los parámetros de acceso a la base de datos (por ejemplo, la ruta local o remota donde se encuentra almacenada).

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  - Desde la interfaz de usuario: parámetros de configuración de la base de datos, claves de búsqueda para localizar registros, registros que se desea añadir o modificar en la base de datos.
  - Desde el *gestor central del VIPS*: claves de búsqueda para localizar registros, nuevos registros para almacenarlos en la base de datos.
  - Desde la base de datos: registros solicitados.
- **Datos de salida:**
  - A la interfaz de usuario: registros obtenidos como resultado de una búsqueda, alarmas de acceso a la base de datos.

- Al *gestor central del VIPS*: registros obtenidos como resultado de una búsqueda, alarmas de acceso a la base de datos.
- A la base de datos: registros que se desea añadir/modificar/eliminar.

- **Datos almacenados:** parámetros de configuración de la base de datos.

P9 - Recursos preexistentes: Existen multitud de sistemas gestores de bases de datos (MySQL, Oracle©, Microsoft© Access, Paradox, etc.). Actualmente se cuenta con alguna experiencia en su manejo.

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1 y F_5 con frecuencia en los SBIO, los SIMED y los SIG.
- F_2 en los SBIO y con frecuencia y con frecuencia también en los SIMED.
- F_3 ocasionalmente en los SIMED.
- F_4 esta funcionalidad raramente se ofrece en los VIPS.

► *Descripción del dominio de virtualización*

P1 - Nombre: Virtualización.

P2 - Descripción: Proporciona la funcionalidad necesaria para reconstruir virtualmente la información visual procesada por el VIPS.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde la interfaz de usuario se reciben los comandos asociados al menú de virtualización.
  - Desde el *gestor central del VIPS* se reciben los resultados del procesamiento de la información visual.
- **Salidas:**
  - A la interfaz de usuario se envían las imágenes resultado de la reconstrucción virtual.

P4 - Dominios de orden superior: Interfaz de usuario.

P5 - Dominios de orden inferior: *gestor central del VIPS*

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

F1 **Iniciar reconstrucción virtual**: se solicita al gestor central del VIPS los resultados del procesamiento de la información visual y a partir de ellos se elabora una reconstrucción virtual. Este proceso puede ser puntual o continuo (bucle que termina cuando el usuario selecciona F_2).

F2 **Detener reconstrucción virtual** (sólo disponible cuando la reconstrucción es continua): se detiene el bucle encargado de la reconstrucción virtual.

F3 **Configurar reconstrucción virtual**: el usuario, en función de sus permisos, configura ciertos parámetros de la reconstrucción virtual (velocidad de refresco de las imágenes, tamaño y resolución de las mismas, significado asociado a ciertos colores, etc.).

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  - Desde la interfaz de usuario: parámetros de configuración del entorno virtual.

  - Desde el *gestor central del VIPS*: resultados del procesamiento de la información visual.
- **Datos de salida:**
  - A la interfaz de usuario: imágenes resultado de la reconstrucción virtual.
- **Datos almacenados:** parámetros de configuración del entorno virtual.

P9 - Recursos preexistentes: Existen varias librerías para desarrollo de entornos virtuales (por ejemplo, OpenGL); actualmente se cuenta con poca experiencia en su manejo.

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1 con frecuencia en SIMED, IPERC y SIG; ocasionalmente en SNAV.
- F_2 y F_3 ocasionalmente en SIMED, IPERC, SNAV y SIG.

► *Descripción del dominio interno de gestión de alarmas*

P1 - Nombre: Gestión de alarmas.

P2 - Descripción: Proporciona la funcionalidad necesaria para configurar y gestionar las alarmas del sistema.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde el *gestor central del VIPS* se reciben las solicitudes de Iniciar alarma de usuario y detener alarma (vía gestor de operativa básica), así como las notificaciones de alarma relacionadas con el fallo en el acceso a una base de datos (vía el gestor de bases de datos).
  - Desde los dominios internos (gestor de procesamiento de la información visual, del control o de las comunicaciones) se reciben las alarmas relativas a fallos en los dispositivos o durante el proceso que éstos llevan a cabo.

P4 - Dominios de orden superior: *gestor central del VIPS*.

P5 - Dominios de orden inferior: ---

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

Importante: las alarmas pueden ser originadas por el sistema o por el usuario; en ambos casos éstas pueden ser de tipo informativo (leves), grave o crítico.

F1 **Iniciar alarma**: creación y disparo de una alarma; si ésta es de tipo leve se informa al usuario mostrando un mensaje en la interfaz de la aplicación; si es de tipo grave o crítico además se informa al *gestor central del VIPS* para que tome las medidas oportunas (por ejemplo, detener el sistema).

F2 **Detener alarma**: a petición del usuario (vía el *gestor central del VIPS*) se detiene la alarma activa.

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:** ---
- **Datos de salida:** notificación de alarma indicando origen, causa y severidad.
- **Datos almacenados:** lista de alarmas activadas.

P9 - Recursos preexistentes: ---

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1 y F_2 en todos los VIPS.

► *Descripción del dominio interno de gestión del procesamiento visual*

P1 - Nombre: Gestión del procesamiento visual.

P2 - Descripción: Proporciona la funcionalidad necesaria para obtener, procesar, caracterizar y clasificar la información visual que llega al VIPS.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde el *gestor central del VIPS* se reciben los siguientes mensajes:
    - Solicitud de arranque/parada/calibración del sistema de visión.
    - Solicitud para iniciar el procesamiento de la información visual. Típicamente el gestor sigue la siguiente secuencia de operaciones: obtener la imagen (de una cámara, de un fichero, o de una base de datos), procesarla convenientemente, extraer de ella la información relevante (características discriminantes) y por último clasificar la información extraída para inferir ciertas propiedades del entorno.

      La operación de procesamiento de la información visual puede contener o no un bucle dependiendo de si el VIPS procesa las imágenes de manera individual o continua. En este último caso cada paso del bucle puede iniciarse de manera síncrona (al terminar el paso anterior), asíncrona (cuando se reciba una señal externa, por ejemplo, desde un sensor), o temporizada (cada cierto tiempo).

      El resultado devuelto de por esta operación al *gestor central del VIP* se utilizará para (1) mostrarlo al usuario a través de la interfaz de la aplicación, (2) almacenarlo en un fichero, en un informe y/o en una base de datos, (3) utilizarlo para construir una simulación virtual del entorno, (4) utilizarlo para realimentar el sistema de control, o (5) enviarlo al exterior a través del sistema de comunicaciones.
    - Solicitud para modificar los parámetros de configuración interna del dominio (vía el gestor de operativa básica) como, por ejemplo, los umbrales de segmentación, la selección de características discriminantes, el tipo de clasificador empleado, etc.). Por lo general los cambios de configuración suelen realizarse mientras el VIPS está detenido aunque algunos sistemas permiten modificaciones on-line.
- **Salidas:**
  - Al *gestor central del VIPS* se envían:
    - Solicitudes para obtener imágenes almacenadas en una base de datos determinada (vía el gestor de bases de datos) para procesarlas o utilizarlas durante el procesamiento de otras.
    - Solicitudes para almacenar los resultados del procesamiento en un fichero o una base de datos (vía el gesto de bases de datos).

- Al *gestor de alarmas* se le envía notificación de todas las situaciones anormales detectadas para que dispare la alarma correspondiente.

P4 - Dominios de orden superior: *gestor central del VIPS*.

P5 - Dominios de orden inferior: ---

P6 - Subdominios: Dada la complejidad de las operaciones de procesamiento, caracterización y clasificación de la información visual resulta adecuado crear un subdominio para gestionar cada una de estas operaciones.

P7 - Funciones / características proporcionadas por el dominio:

Importante: la información visual (imágenes) que se procesa en este dominio puede no corresponderse con la que se percibe a simple vista; este es el caso, por ejemplo, de las imágenes de rayos X, las imágenes térmicas, los mapas de presión atmosférica, etc. Independientemente de lo que signifique la información representada en las imágenes, éstas pueden clasificarse atendiendo a múltiples factores: número de planos, profundidad del píxel, resolución, etc.

F1 **Arrancar sistema de visión:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se procede a arrancar los dispositivos del sistema de visión (cámaras, digitalizadores, etc.) siguiendo el protocolo adecuado.

F2 **Detener sistema de visión:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se procede a detener todos los dispositivos del sistema de visión siguiendo el protocolo adecuado.

F3 **Calibrar cámaras:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se procede a calibrar las cámaras del sistema de visión siguiendo un procedimiento manual o (semi-)automático.

F4 **Procesar información visual:** a petición del gestor central del VIPS se inicia el procesamiento de una o más imágenes que se obtendrán solicitándolas al sistema de visión o bien al gestor de bases de datos (vía el gestor central del VIPS). Las etapas que se llevan a cabo como parte de esta operación (procesamiento de la imagen, caracterización y clasificación) se han diferido a los correspondientes subdominios.

F5 **Notificar alarma del sistema de visión:** a petición de alguno de los dispositivos del sistema de visión (cámaras, digitalizadores, etc.) se notifica la detección de un fallo al gestor de alarmas.

F6 **Configurar procesamiento visual:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se modifican los parámetros relacionados con la adquisición y el procesamiento de las imágenes.

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  - Desde el *gestor central del VIPS*: parámetros de configuración interna relacionados con la adquisición/procesamiento de las imágenes (obtenidos vía el gestor de operativa básica) e imágenes solicitadas a una base de datos.
- **Datos de salida:**
  - Al *gestor central del VIPS*: resultados de procesar la información visual.

- **Datos almacenados:** parámetros de configuración necesarios para el procesamiento de la información visual.

P9 - Recursos preexistentes: Existen diversas herramientas y librerías de funciones para procesamiento de imágenes y reconocimiento de patrones; se cuenta con bastante experiencia en el manejo de varias de ellas (Intel© IPL y OpenCV, Matlab© Image Processing Toolbox, Matrox© Imaging Library, etc.).

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1, F_2, F_3 y F_5 con frecuencia en todos los VIPS salvo en los SIG.
- F_4 en todos los VIPS.
- F_6 con frecuencia en todos los VIPS.

► ***Descripción del subdominio de procesamiento de imágenes***

P1 - Nombre: Procesamiento de imágenes.

P2 - Descripción: Conjunto de funciones para procesamiento de imágenes seleccionadas de antemano (durante el proceso de instanciación de la aplicación a partir de la línea LPS-VIPS) para resaltar la información de interés en las imágenes de entrada.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde el *gestor de procesamiento visual* se envía:
    - Mensaje para configurar los parámetros de algunos de los algoritmos empleados (por ejemplo, umbrales de segmentación).
  - Solicitud para que se procese la imagen que se le envía.
- **Salidas:** ---

P4 - Dominios de orden superior: *gestor de procesamiento visual*.

P5 - Dominios de orden inferior: ---

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

F1 Procesar imagen: A petición del gestor de procesamiento visual se procesa una imagen y se devuelve la imagen resultado.

F2 Configurar parámetros de procesamiento de imágenes: A petición del gestor de procesamiento visual (vía el gestor de operativa básica) se establecen los parámetros necesarios para el correcto funcionamiento de los algoritmos de procesamiento de imágenes seleccionados.

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:** imagen a procesar
- **Datos de salida:** imagen procesada
- **Datos almacenados:** parámetros de configuración de los algoritmos de procesamiento de imágenes.

P9 - Recursos preexistentes: (los ya comentados en el dominio padre).

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1 en todos los VIPS.
- F_2 con frecuencia en todos los VIPS.

► *Descripción del subdominio de extracción de características*

P1 - Nombre: Extracción de características.

P2 - Descripción: Conjunto de funciones para extracción de características visuales discriminantes seleccionadas de antemano (durante el proceso de instanciación de la aplicación a partir de la línea LPS-VIPS).

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde el *gestor de procesamiento visual* se envía:
    - Mensaje para configurar los parámetros de algunos de los algoritmos empleados (por ejemplo, número de descriptores de Fourier que se emplearán para caracterizar una forma).
    - Solicitud para que se extraigan las características visuales discriminantes de la imagen que se le envía (previamente procesada).
- **Salidas:** ---

P4 - Dominios de orden superior: *gestor de procesamiento visual*.

P5 - Dominios de orden inferior: ---

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

F1 **Extraer características:** A petición del gestor de procesamiento visual se extraen las características visuales discriminantes de una imagen y se devuelve el vector numérico resultante.

F2 **Configurar parámetros de extracción de características:** A petición del gestor de procesamiento visual (vía el gestor de operativa básica) se establecen los parámetros necesarios para el correcto funcionamiento de los algoritmos de extracción de características seleccionados.

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:** imagen procesada.
- **Datos de salida:** vector de características numéricas.
- **Datos almacenados:** parámetros de configuración de los algoritmos de extracción de características.

P9 - Recursos preexistentes: (los ya comentados en el dominio padre).

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1 en todos los VIPS.
- F_2 con frecuencia en todos los VIPS.

► *Descripción del subdominio de clasificación de características*

P1 - Nombre: Clasificación de características.

P2 - Descripción: Una o más funciones de clasificación seleccionadas de antemano (durante el proceso de instanciación de la aplicación a partir de la línea LPS-VIPS) que asocian un significado a los vectores de características previamente extraídos de una imagen.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde el *gestor de procesamiento visual* se envía:
    - Mensaje para configurar los parámetros de algunos de los algoritmos empleados (por ejemplo, parámetros de una red neuronal).
    - Solicitud para que clasifique el vector de características que se le envía en una de las categorías previamente definidas.
- **Salidas:** ---

P4 - Dominios de orden superior: *gestor de procesamiento visual*.

P5 - Dominios de orden inferior: ---

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

F1 **Clasificar características:** A petición del gestor de procesamiento visual se clasifica el vector de características y se devuelve el identificador de la categoría resultante.

F2 **Configurar parámetros de clasificación:** A petición del gestor de procesamiento visual (vía el gestor de operativa básica) se establecen los parámetros necesarios para el correcto funcionamiento de los algoritmos de clasificación seleccionados.

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:** vector de características.
- **Datos de salida:** identificador de la categoría del vector.
- **Datos almacenados:** parámetros de configuración de los algoritmos de extracción de características.

P9 - Recursos preexistentes: (los ya comentados en el dominio padre).

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1 en todos los VIPS.
- F_2 con frecuencia en todos los VIPS.

► ***Descripción del dominio interno de gestión del control***

P1 - Nombre: Gestión del control.

P2 - Descripción: Proporciona la funcionalidad necesaria para que el VIPS pueda interactuar correctamente con su entorno, permitiéndole controlar sus actuadores en función de lo que percibe a través de sus sensores y, en particular, de la información visual que recibe a través del gestor de procesamiento visual (vía el *gestor central del VIPS*).

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde el *gestor central del VIPS* se reciben mensajes para:
    - Arrancar, parar y calibrar los sensores y actuadores del sistema.
    - Iniciar el proceso de regulación del sistema a partir de los datos obtenidos de los sensores y/o de los resultados del procesamiento de la información visual. Como resultado de esta operación se devuelve al gestor central del

VIP el estado del sistema de control tras la regulación (lecturas de los sensores, órdenes enviadas a los actuadores, etc.).
- Modificar los parámetros de configuración interna del dominio (vía el gestor de operativa básica) como, por ejemplo, las constantes de regulación, la periodicidad de lectura de sensores, etc. Por lo general los cambios de configuración suelen realizarse mientras el VIPS está detenido aunque algunos sistemas permiten realizar estas modificaciones on-line.

  ○ Desde cualquiera de los elementos físicos del sistema de control (sensores, actuadores, controladores) puede recibirse una señal de alarma informando de un fallo.

- **Salidas:**
  ○ A los sensores se les envían órdenes de arranque/parada/calibración y mensajes solicitándoles su lectura actual.
  ○ A los actuadores órdenes de arranque/parada y calibración.
  ○ Al *gestor de alarmas* se le envían las notificaciones de los fallos detectados en los dispositivos o de las situaciones de alarma generadas durante el proceso de regulación (por ejemplo, si la lectura de un sensor sobrepasa un umbral máximo establecido).

P4 - Dominios de orden superior: *gestor central del VIPS*.

P5 - Dominios de orden inferior: ---

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

F1 **Arrancar sistema de control:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se procederá a arrancar los sensores y actuadores que forman parte del sistema de control siguiendo el protocolo correspondiente.

F2 **Detener sistema de control:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se procederá a detener los sensores y actuadores siguiendo el protocolo correspondiente.

F3 **Calibrar sistema de control:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se procederá a la calibración de los sensores y actuadores del sistema de control siguiendo un procedimiento manual o (semi-)automático.

F4 **Regular sistema:** a petición del gestor central del VIPS se regula el sistema (a veces activando de uno o más de los actuadores) en función de los resultados del procesamiento visual y/o de las lecturas de los sensores.

F5 **Configurar control:** a petición del gestor de operativa básica (vía el gestor central del VIPS) se configuran ciertos parámetros del control como las constantes de regulación, el rango válido de los sensores, etc.

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  ○ Desde el *gestor central del VIPS*: parámetros de configuración del sistema de control (obtenidos vía el gestor de operativa básica).
  ○ Resultados del procesamiento de la información visual.

- **Datos de salida:**
  - Al *gestor central del VIPS*: estado del sistema de control tras la regulación (lecturas de sensores, órdenes enviadas a actuadores, etc.).
- **Datos almacenados:** parámetros de configuración del sistema de control.

P9 - Recursos preexistentes: Existen diversas herramientas y librerías de funciones para control y automatización de procesos; actualmente se cuenta con alguna experiencia en su manejo.

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1, F_2, F_3 y F_4 en los SIVA y en los SNAV; ocasionalmente en el resto de VIPS.
- F_5 con frecuencia en los SIVA y en los SNAV.

► ***Descripción del dominio interno de gestión de la comunicación***

P1 - Nombre: Gestión de la comunicación.

P2 - Descripción: Proporciona la funcionalidad necesaria para que el resto de los dominios se comuniquen entre sí, con los dispositivos externos y con los sistemas de ficheros ubicados remotamente.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde el gestor central del VIPS se reciben mensajes para:
    - Arrancar y parar el sistema de comunicación.
    - Enviar (recibir) datos de un dispositivo externo, de una base de datos remota o de un sistema de ficheros remoto.
    - Modificar parámetros de configuración del sistema de comunicación (vía el gestor de operativa básica) como, por ejemplo, la configuración de los puertos, la conexión a emplear en cada caso (si hubiera más de una), etc.
- **Salidas:**
  - Al *gestor de alarmas* se le envían las notificaciones de los fallos detectados en la comunicación.

P4 - Dominios de orden superior: *gestor central del VIPS*.

P5 - Dominios de orden inferior: ---

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

Importante: la comunicación puede realizarse utilizando canales seguros o no seguros, mensajes cifrados o no, protocolos TPC, UDP o específicos de un determinado dispositivos (por ejemplo, comunicación vía PCI con una FPGA), etc.

F1 Arrancar sistema de comunicación: a petición del gestor de operativa básica (vía el gestor central del VIPS) se procederá a arrancar los dispositivos de comunicación siguiendo el protocolo correspondiente.

F2 Detener sistema de comunicación: a petición del gestor de operativa básica (vía el gestor central del VIPS) se procederá a detener los dispositivos de comunicación siguiendo el protocolo correspondiente.

F3 Enviar dato: a petición del gestor central del VIPS se envían los datos al destino solicitado.

F4 Recibir dato: a petición del gestor central del VIPS se solicita al destino solicitado que envíe un determinado dato.

F5 Configurar comunicación: a petición del gestor de operativa básica (vía el gestor central del VIPS) se configuran los parámetros de la comunicación (puertos, protocolos a emplear, etc.).

P8 - Datos manejados / almacenados en el dominio:

- **Datos de entrada:**
  - Desde el *gestor central del VIPS*: parámetros de configuración del sistema de comunicaciones (obtenidos vía el gestor de operativa básica) y datos a enviar.
- **Datos de salida:**
  - Al *gestor central del VIPS*: datos recibidos.
- **Datos almacenados:** parámetros de configuración del sistema de control.

P9 - Recursos preexistentes: Existen diversas herramientas y librerías de funciones para gestión de las comunicaciones; actualmente se cuenta con alguna experiencia en su manejo.

P10 - Productos en los que se implementará la funcionalidad de este dominio:

- F_1, F_2, F_3 y F_4 en los SIVA y en los SNAV; ocasionalmente en el resto de VIPS.
- F_5 con frecuencia en los SIVA y en los SNAV.

► ***Descripción del dominio interno de gestión central del VIPS***

P1 - Nombre: Gestión central del VIPS.

P2 - Descripción: Este dominio sincroniza y gestiona de manera centralizada todas las operaciones del VIPS actuando como mediador entre los distintos dominios.

P3 - Funcionalidad ofrecida y demandada por el dominio:

- **Entradas:**
  - Desde los dominios externos se reciben los comandos asociados a los distintos menús.
  - Desde el gestor de bases de datos se reciben las notificaciones de alarma que éste pueda generar (el resto de notificaciones se envían directamente al gestor de alarmas).
  - Desde el *gestor de procesamiento visual* se reciben solicitudes para obtener imágenes desde una base de datos o para almacenar nuevos registros en ella.
- **Salidas:**
  - Al *gestor de procesamiento visual* se le envían mensajes de arranque, parada, calibración y configuración del sistema de visión, así como la orden de iniciar el procesamiento de la información visual.
  - Al *gestor de control* se le envían mensajes de arranque, parada, calibración y configuración del sistema de control, así como la orden de iniciar el proceso de regulación.
  - Al gestor de la comunicación se le envían mensajes de arranque, parada y configuración del sistema de comunicación, así como las órdenes de enviar o recibir datos.

P4 - Dominios de orden superior: ---

P5 - Dominios de orden inferior: *todos.*

P6 - Subdominios: ---

P7 - Funciones / características proporcionadas por el dominio:

Se omite la enumeración exhaustiva de las funciones del dominio debido a su excesiva longitud. En el punto P_3 está recogida de forma resumida toda la información relativa a la funcionalidad del dominio.

P8 - Datos manejados en el dominio:

- **Datos de entrada obtenidos desde:**
  - El gestor de operativa básica: parámetros de configuración (internos y externos).
  - El gestor de informes: listado de datos que solicita se le devuelvan para completar un informe.
  - El gestor de bases de datos: registros recuperados tras una búsqueda.
  - El gestor de alarmas: notificación de alarmas para que le sean comunicadas al usuario.
  - El *gestor de procesamiento visual*: resultados tras procesar una imagen.
  - El *gestor de control*: estado del sistema de control tras regulación.
  - El *gestor de comunicación:* datos solicitados a un sistema remoto.
- **Datos de salida enviados a:**
  - El gestor de operativa básica: resultados de los procesos internos.
  - El gestor de informes: datos solicitados para rellenar informe.
  - El gestor de bases de datos: registros a añadir/modificar.
  - El gestor de alarmas: notificación de alarma de acceso a una BD.
  - El *gestor de procesamiento visual*: imágenes solicitadas a una BD y parámetros de configuración del sistema de visión.
  - El *gestor de control*: resultados del procesamiento visual y parámetros de configuración del sistema de control.
  - El *gestor de comunicación:* datos a enviar a un sistema remoto y parámetros de configuración del sistema de comunicación.

P9 - Recursos preexistentes: ---

P10 - Productos en los que se implementará la funcionalidad de este dominio:

Se omite la enumeración debido a su excesiva longitud.

II.2. Descripción de Casos de Uso (PuLSE™-CDA)

En este apartado se recogen las descripciones textuales de los Casos de Uso (CU) identificados en el diagrama de UML elaborado como parte de la fase de análisis del dominio PuLSE™-CDA (ver Capítulo 4, apartado 4.3.1). La plantilla que se empleará para describir estos CU es la que se muestra a continuación en el cuadro de la Figura 55.

P1 - Nombre: nombre del CU. La utilización del estereotipo <<VARIANT>> antes del nombre del CU indicará que éste puede aparecer o no en los productos de la línea (equivalente a añadir una alternativa nula (∅) a cada uno de los pasos de la secuencia de éxito y a sus alternativas,).
P2 - Actor principal: Actor que desencadena la acción representada por el CU.
P3 - Dominio: Dominio de aplicación en el que se encuadra la funcionalidad recogida en el CU.
P4 - Nivel: Nivel desde el que se describe el CU (global, usuario o subsistema).
P5 - Precondiciones: Estado en el que debe estar el sistema y condiciones que deben cumplirse antes de comenzar el CU.
P6 - Postcondición en un escenario de éxito: Estado en el que quedará el sistema una vez completado el CU si no se produce ningún fallo.
P7 - Principal secuencia de éxito: Se describen las etapas de una ejecución típica del CU en la que no ocurre ningún error. Aquellas etapas que se ejecuten sólo de manera opcional se etiquetarán con el estereotipo <<OPT>>; aquellas en las que existan distintas alternativas de ejecución se etiquetarán con el estereotipo << ALT i >>.
P8 - Extensiones: Se describen las distintas situaciones de fallo que pueden ocurrir durante la ejecución de cada una de las etapas de la secuencia de éxito.
P9 - Requisitos no funcionales: Se enumeran los requisitos no funcionales de aplicación a este caso de uso.

Figura 55. Plantilla para la descripción de CU extendida con variabilidad.

A continuación se presentan las especificaciones de los CU identificados en los dominios de gestión de operativa básica, gestión del procesamiento visual y gestión del control, respectivamente.

► ***Caso de Uso: arrancar VIPS***

P1 - Nombre: <<*VARIANT*>> arrancar VIPS
Sólo los SIVA, los SNAV y ocasionalmente los SIMED ofrecen al usuario (de manera explícita) esta funcionalidad.
P2 - Actor principal: usuario.
P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones:

- El sistema muestra activada la opción "arrancar VIPS" en el menú de gestión de operativa básica.
- Los dispositivos externos del VIPS están correctamente conectados.

P6 - Postcondición en un escenario de éxito: El sistema muestra activada la opción "parar VIPS" en el menú de gestión de operativa básica y todos los subsistemas y dispositivos externos están arrancados y funcionando.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "arrancar VIPS" del menú de gestión de operativa básica.
2. El sistema carga los parámetros de configuración del arranque y los envía al gestor central del VIPS mediante un mensaje solicitándole la inicialización de los distintos subsistemas y dispositivos externos.
3. El sistema espera la notificación del gestor central del VIPS indicándole que las operaciones de arranque e inicialización se han completado con éxito.
4. El sistema desactiva la opción "arrancar VIPS" y activa la opción "parar VIPS" en el menú de gestión de operativa básica.
5. El sistema notifica al usuario que la operación ha concluido con éxito mediante una ventana de mensaje.

P8 - Extensiones:

3a) Se recibe una notificación de alarma indicando que durante el arranque se ha producido un fallo en alguno de los dispositivos externos que se trataban de inicializar. El sistema notifica la alarma al usuario en una ventana de mensajes.

P9 - Requisitos no funcionales:

***** Una vez que el usuario ha activado la opción "arrancar VIPS" se le deberá notificar el éxito o el fallo en la operación en un tiempo limitado (depende de los dispositivos externos que tenga el VIPS).

► ***Caso de Uso: parar VIPS***

P1 - Nombre: <<*VARIANT*>> parar VIPS

Como en el caso anterior, sólo los SIVA, los SNAV y ocasionalmente los SIMED ofrecen esta funcionalidad explícitamente al usuario.

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones:

- El sistema muestra activada la opción "parar VIPS" en el menú de gestión de operativa básica.
- Los dispositivos externos del VIPS están funcionando correctamente.

P6 - Postcondición en un escenario de éxito: El sistema muestra activada la opción "arrancar VIPS" en el menú de gestión de operativa básica y todos los subsistemas y dispositivos externos están parados.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "parar VIPS" del menú de gestión de operativa básica.
2. El sistema envía un mensaje al *gestor central del VIPS* solicitándole la parada de los distintos subsistemas y dispositivos externos.
3. El sistema espera la notificación del gestor central del VIPS indicándole que la operación de parada se ha completado con éxito.
4. El sistema desactiva la opción "parar VIPS" y activa la opción "arrancar VIPS" en el menú de gestión de operativa básica.
5. El sistema notifica al usuario que la operación ha concluido con éxito mediante una ventana de mensaje.

P8 - Extensiones:

3a) Se recibe una notificación de alarma indicando que durante la parada se ha producido un fallo en alguno de los dispositivos externos. El sistema notifica la alarma al usuario en una ventana de mensajes.

P9 - Requisitos no funcionales:

***** Una vez que el usuario ha activado la opción "parar VIPS" se le deberá notificar el éxito o el fallo en la operación en un tiempo limitado (depende de los dispositivos externos que tenga el VIPS).

► ***Caso de Uso: iniciar proceso***

P1 - Nombre: iniciar proceso
Todos los VIPS ofrecen esta funcionalidad.

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones: El sistema está arrancado y funcionando (no hay pendiente ninguna alarma).

P6 - Postcondición en un escenario de éxito: El sistema está arrancado y funcionando (no hay alarmas pendientes). Al menos se ha realizado una operación de procesamiento de información visual.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "iniciar proceso" del menú de gestión de operativa básica. Una vez seleccionada esta opción deja de estar activa en el menú.
2. En las opciones de configuración del VIPS ¿está activada la opción de mostrar los resultados del procesamiento de la información visual?
   <OPT> El gestor de operativa básica crea una ventana de resultados con el formato indicado en las opciones de configuración, esto es, con los elementos gráficos necesarios para visualizar toda la información que se genera durante el proceso.

3. El sistema envía un mensaje al *gestor central del VIPS* solicitándole que inicie el procesamiento de la información visual. Si se creó una ventana para visualizar los resultados, en la solicitud se adjuntará una referencia a la misma (se difiere al *gestor central del VIPS* la tarea de actualizarla con los resultados).

4. ¿Qué tipo de información visual se va a procesar?
   <ALT 1> Una imagen individual (SBIO, SIMED y SIG). En este caso el sistema esperará a que el gestor central del VIPS le notifique que la operación ha concluido con éxito y volverá a activar la opción de "iniciar proceso" en el menú.
   <ALT 2> Una secuencia de vídeo almacenada en un fichero (SIG, SBIO) o un flujo de vídeo en vivo que se procesará durante un intervalo de tiempo prefijado de antemano (SIMED) o de manera indefinida (SIVA, SNAV, IPERC). En este caso el sistema activará la opción "detener proceso" y esperará a que, o bien el usuario seleccione esta opción, o bien el gestor central del VIPS le informe de que la operación ha concluido con éxito. Cuando ocurra alguna de estas dos situaciones, se volverá a activar la opción de "iniciar proceso" en el menú.

P8 - Extensiones:

4a) Se recibe una notificación de alarma indicando que durante el proceso se ha producido algún error ocasionado por alguna de las operaciones realizadas por los subsistemas o dispositivos implicados en el proceso.

P9 - Requisitos no funcionales:

- ***** En muchos VIPS, la velocidad con la que se procesa la información visual es un factor crítico (por ejemplo en los SIVA y en los SNAV); en algunos de estos VIPS existen fuertes restricciones de tiempo real.
- ***** En ciertos casos resulta imprescindible asegurar la robustez y fiabilidad de este proceso. Por ejemplo, en el caso de los SNAV, éstos deben ser capaces de reaccionar de forma segura aún cuando las condiciones de visibilidad sean pésimas.
- ***** En ocasiones la precisión también resulta esencial, por ejemplo en algunos SIMED.

► *Caso de Uso: detener proceso*

P1 - Nombre: <<*VARIANT*>> detener proceso
Sólo ofrecen esta funcionalidad aquellos VIPS que procesan información visual de continua (flujos de vídeo grabados o en vivo).

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones:

- El sistema muestra activada la opción "detener proceso" en el menú de gestión de operativa básica.
- El sistema está procesando información visual de manera continua.

P6 - Postcondición en un escenario de éxito: El sistema muestra activada la opción "iniciar proceso" en el menú de gestión de operativa básica y el procesamiento de información visual se ha detenido.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "detener proceso" del menú de gestión de operativa básica; el sistema desactiva en ese momento la entrada del menú.
2. El sistema envía al *gestor central del VIPS* la orden de que termine de procesar los últimos datos y espera a recibir confirmación de que el proceso ha concluido satisfactoriamente.
3. Se activa en el menú de gestión de operativa básica la opción "iniciar proceso".

P8 - Extensiones:

2a) Se recibe una notificación de alarma indicando un fallo en el sistema.

P9 - Requisitos no funcionales:

***** Una vez que el usuario ha activado la opción "detener VIPS" se le deberá notificar el éxito o el fallo en la operación en un tiempo limitado (depende de los dispositivos externos que tenga el VIPS).

► ***Caso de Uso: calibrar VIPS***

P1 - Nombre: <<*VARIANT*>> Calibrar VIPS
La mayoría de los VIPS ofrecen esta funcionalidad salvo quizá algunos SIG.

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones:

- El sistema está arrancado, funcionando (no existen alarmas pendientes), y se encuentra detenido (la opción "iniciar proceso" está activada).
- La opción "calibrar VIPS" está activada en el menú de gestión de operativa básica.

P6 - Postcondición en un escenario de éxito: El sistema sigue arrancando, funcionando y detenido (mismo estado que antes de iniciar la calibración) y los sensores y actuadores externos están correctamente calibrados.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "calibrar VIPS" del menú de gestión de operativa básica; en ese momento se desactivan todos los menús de la aplicación (no se puede realizar ninguna operación durante la calibración).
2. El sistema envía al gestor central del VIPS una orden de calibración y una lista de los dispositivos que requieren este proceso (este dato debe estar almacenado en los parámetros de configuración, así como las respuestas a las siguientes cuestiones).

3. ¿Qué tipo de calibración se realiza para los sensores?
   <ALT 1> Calibración manual (todos los VIPS). En este caso, el *gestor central del VIPS* mostrará al usuario una ventana de calibración en la que se le mostrarán las lecturas de cada uno de los sensores que hay que calibrar (obtenidos mediante el envío de una solicitud al gestor del control o del procesamiento visual). El usuario deberá observar cómo varían estas lecturas ante los cambios que se producen en la propiedad que se está midiendo. El resultado de la calibración de cada sensor será un conjunto de pares (valor medido por el usuario, lectura del sensor). Será el usuario quien anotará estos pares en el espacio al efecto proporcionado en la ventana de calibración.
   <ALT 2> Calibración (semi-) automática (muchos de los VIPS). En este caso, es el sistema el que se encarga de obtener y emparejar las medidas reales y las lecturas de los sensores. En algunos casos este proceso requiere cierta intervención del usuario, por ejemplo, para ubicar el sensor en un lugar adecuado.
4. ¿Se van a calibrar los actuadores? (no en todos los casos tiene sentido, por ejemplo, en el caso de una electroválvula).
   <OPT> Si existen actuadores susceptibles de ser calibrados (por ejemplo, brazos robóticos), se ofrecerá al usuario una ventana de calibración mediante la que podrán variar su estado siguiendo un patrón de calibración predefinido. Cada vez que se alcance una marca en dicho patrón se guardará el estado del actuador. El resultado de la calibración será un conjunto de pares (marca, estado).
5. Una vez completado el proceso de calibración, el *gestor central del VIPS* enviará notificación al gestor de operativa básica quien volverá a activar el menú en el mismo estado en el que se encontraba antes de la calibración.

P8 - Extensiones:

3-4a) Se recibe una notificación de alarma indicando un fallo durante la calibración. El sistema notifica la alarma al usuario en una ventana de mensajes. A continuación carga los datos de la última calibración completada con éxito y vuelve a activar el menú de operativa básica.

P9 - Requisitos no funcionales:

***** Una vez que el usuario ha activado la opción "calibrar VIPS" se le deberá notificar el éxito o el fallo en la operación en un tiempo limitado (depende de los dispositivos externos que tenga el VIPS).

► *Caso de Uso: configurar parámetros*

P1 - Nombre: <<*VARIANT*>> configurar parámetros
Ofrecido por la mayoría de los VIPS pero no necesariamente por todos.

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones:

- El sistema muestra activada la opción "configurar parámetros" en el menú de gestión de operativa básica. En función del tipo de VIPS del que se trate, esta opción estará activa durante el procesamiento de la información visual (cambio de configuración on-line), o sólo cuando el sistema esté detenido.

P6 - Postcondición en un escenario de éxito: El sistema muestra activada la opción "configurar parámetros" en el menú de gestión de operativa básica y los parámetros de configuración están correctamente establecidos.

P7 - Principal secuencia de éxito:

1. El usuario selecciona la opción "configurar parámetros" del menú de gestión de operativa básica.
2. El sistema ofrece al usuario una ventana modal (impide el acceso a la aplicación principal hasta que no se cierre) en la que aparecen los valores actuales de los parámetros de configuración. Aquellos que puedan ser modificados por el usuario aparecerán como campos editables. Para cerrar dicha ventana y volver a la aplicación principal el usuario deberá seleccionar una de las siguientes opciones: "aceptar cambios" o "cancelar cambios". En el primer caso, se almacenarán los datos contenidos en la ventana (hayan sido modificados o no) como nuevos valores de configuración. Si por el contrario el usuario decide cancelar los cambios, no se realizará ninguna actualización de valores. En ambos casos se cerrará la ventana de configuración.

P8 - Extensiones: ---

P9 - Requisitos no funcionales: ---.

► ***Caso de Uso: detener alarma***

P1 - Nombre: <<*VARIANT*>> detener alarma
Ofrecido por la mayoría de los VIPS pero no necesariamente por todos.

P2 - Actor principal: usuario.

P3 - Dominio: Gestión de operativa básica.

P4 - Nivel: usuario.

P5 - Precondiciones:

- Existe al menos una notificación de alarma activa en el sistema.

P6 - Postcondición en un escenario de éxito: La alarma activada ha cesado.

P7 - Principal secuencia de éxito:

- El sistema muestra las notificaciones de alarmas activas en una ventana.
- ¿De qué tipo de alarma se trata?

  <ALT 1> Alarma informativa (leve). En este caso, basta con que el usuario se de por informado y cierre la ventana de notificación. Por ejemplo, una alarma leve es la que se produce cuando una operación de calibración tarda más del tiempo previsto.

<ALT 2> Alarma grave. Este tipo de alarmas puede requerir la intervención del usuario. Por ejemplo, a este tipo corresponden las alarmas ocasionadas por un fallo en un dispositivo externo (puede ser necesario que el usuario lo reinicie manualmente).

<ALT 3> Alarma crítica. Este tipo de alarmas responden a fallos graves en los sensores y/o en los actuadores del sistema hasta el punto de poner en serio peligro la integridad del sistema e incluso a veces la de las personas. En la mayoría de los casos, este tipo de alarmas se solventan parando totalmente el VIPS; a veces, es el usuario el que debe realizar esta operación manualmente.

P8 - Extensiones: ---

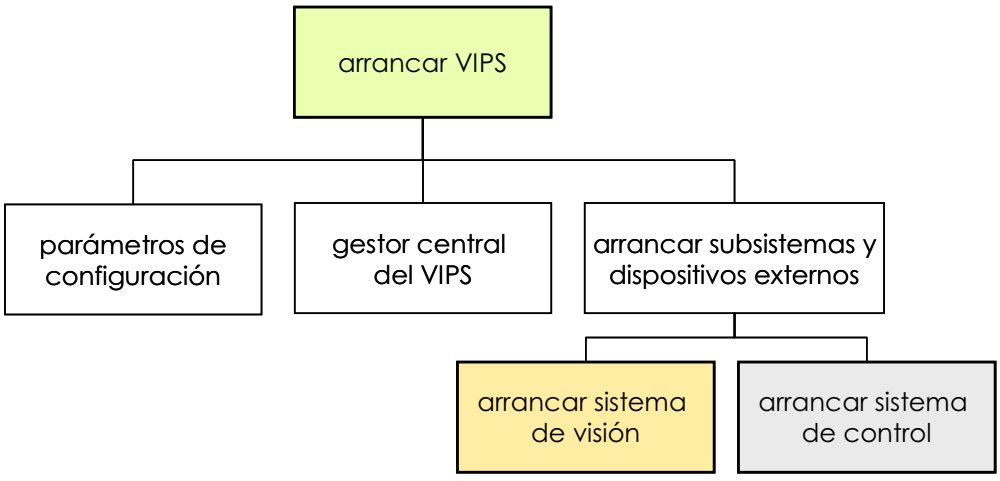
P9 - Requisitos no funcionales:

* Cuanto más grave sea el fallo que ocasiona una alarma, más rápidamente se debe tratar de detectar, informar y solventar. La velocidad en la gestión de alarmas es crucial en muchos VIPS; en particular en los SIVA, SBIO, SIMED, y SNAV.

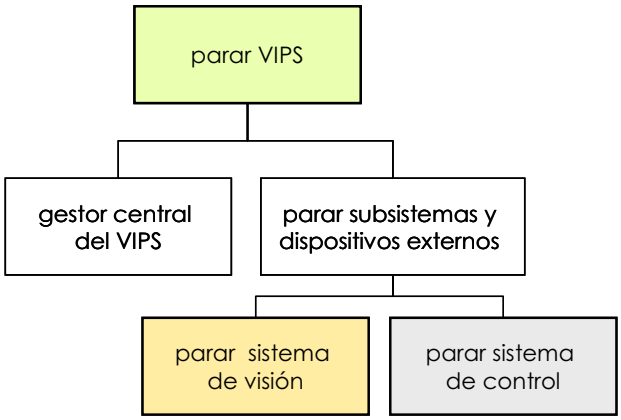
II.3. Diagramas de variabilidad (PuLSE™-CDA)

En este apartado se recogen los diagramas de variabilidad asociados a aquellos Casos de Uso (CU) en cuya descripción se haya identificado algún elemento variante. Estos diagramas se han realizado utilizando la notación FODA (Feature-Oriented Domain Analisys) [Kang 90]. Los distintos CU se han coloreado en función de los distintos subsistemas en los que se han incluido: en verde los correspondientes al subsistema de operativa básica, en naranja los del subsistema de gestión de la información visual y, por último, en violeta los correspondientes al subsistema de gestión del control.

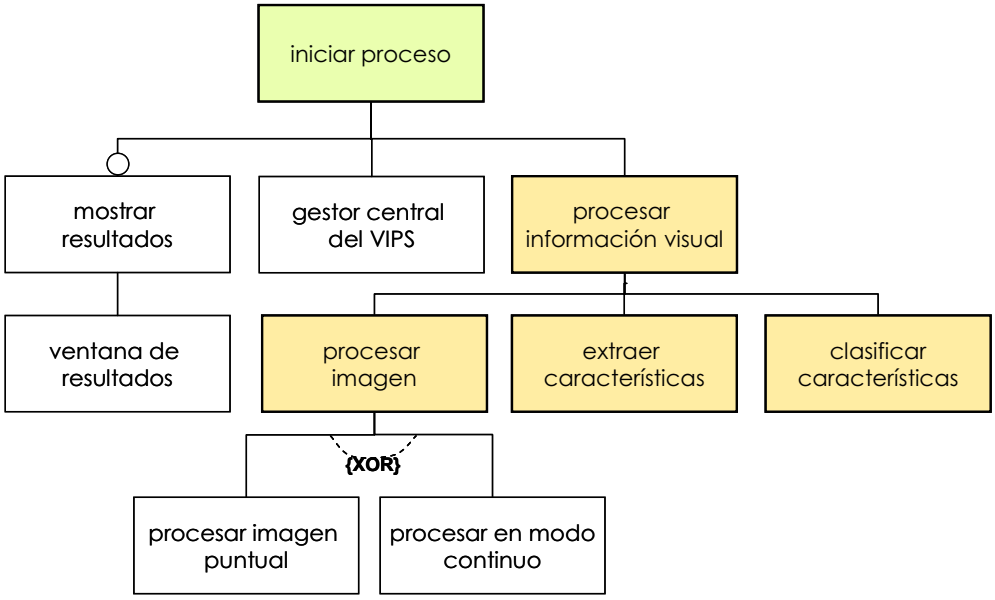
► **Diagrama de variabilidad del CU: *arrancar VIPS***



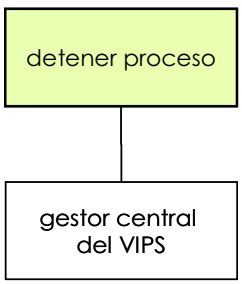
► **Diagrama de variabilidad del CU: *parar VIPS***



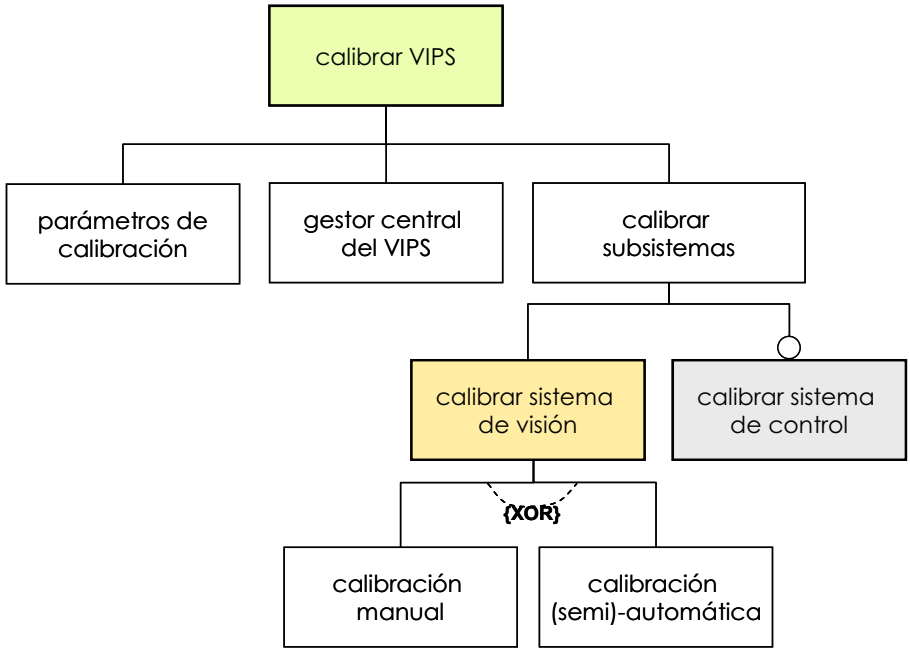
► **Diagrama de variabilidad del CU: *iniciar proceso***



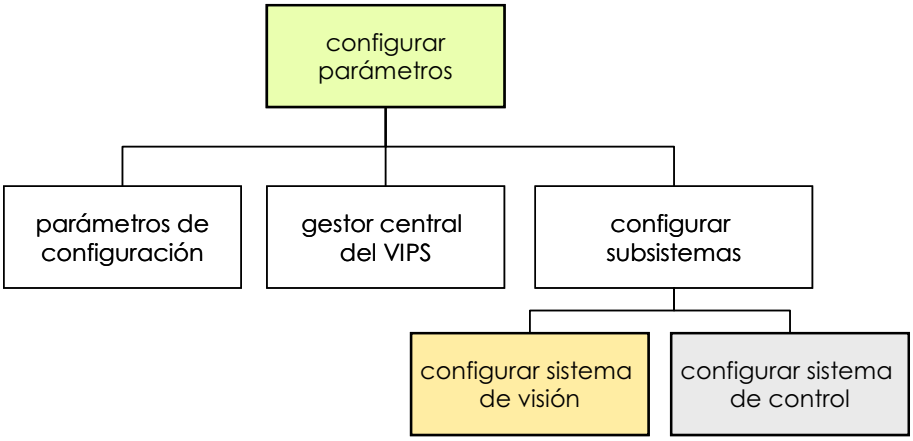
► **Diagrama de variabilidad del CU: *detener proceso***



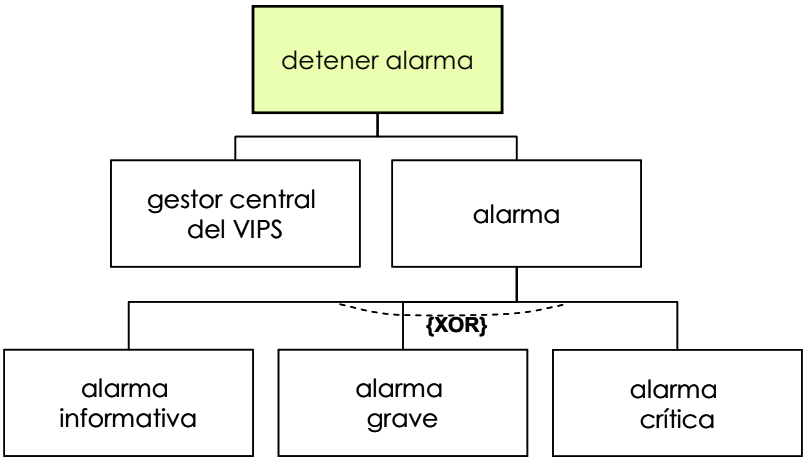
► **Diagrama de variabilidad del CU: *calibrar VIPS***



► **Diagrama de variabilidad del CU: *configurar parámetros***



► **Diagrama de variabilidad del CU: *detener alarma***



II.4. Modelos de decisión (PuLSE™-CDA)

En este apartado se recogen, en forma de tabla, los modelos de decisión asociados a los CU previamente descritos en los que se han identificado características opcionales y/o alternativas.

► **Modelo de decisión asociado al CU: *iniciar proceso***

ID	Pregunta	Elemento variante	Resolución	Efecto
1	¿Se oferta la opción "mostrar resultados"?	Opción de menú "mostrar resultados"	Sí	Paso 2 del CU *iniciar proceso* es obligatorio
			No	Eliminar paso 2 del CU *iniciar proceso*
2	¿Qué tipo de procesamiento se realiza?	iniciar procesamiento visual	Puntual	Eliminar <ALT 2> en el paso 4 del CU *iniciar proceso*
			Continuo	Eliminar <ALT 1> en el paso 4 del CU *iniciar proceso*

► Modelo de decisión asociado al CU: *calibrar VIPS*

ID	Pregunta	Elemento variante	Resolución	Efecto
1	¿Qué tipo de calibración se realiza de los sensores?	Tipo de calibración	Manual	Eliminar <ALT 2> en el paso 3 del CU *calibrar VIPS*
			(Semi-) automática	Eliminar <ALT 1> en el paso 3 del CU *calibrar VIPS*
2	¿Existen actuadores que necesiten calibrarse?	Opción "calibrar actuadores"	Sí	Paso 4 del CU *calibrar VIPS* es obligatorio
			No	Eliminar paso 4 del CU *calibrar VIPS*

► Modelo de decisión asociado al CU: *detener alarma*

ID	Pregunta	Elemento variante	Resolución	Efecto
1	¿De qué tipo de alarma se trata?	Alarma	Informativa	Eliminar <ALT 2> y <ALT 3> en el paso 2 del CU *detener alarma*
			Grave	Eliminar <ALT 1> y <ALT 3> en el paso 2 del CU *detener alarma*
			Crítica	Eliminar <ALT 1> y <ALT 2> en el paso 2 del CU detener alarma

Listado de Acrónimos

AHEAD	Algebraic Hierarchical Equations for Application Design
AOSD	Aspect-Oriented Software Development
AS	Arquitectura Software
ASIC	Application-Specific Integrated Cirtuit
CAD	Computer-Aided Design
CASE	Computer-Aided Software Engineering
CBSE	Component-Based Software Engineering
CCTV	Closed Ciruit TV
COM	Component Object Model (Microsoft®)
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-the-Shelf
CS	Componente Software
CU	Caso de Uso
DCOM	Distributed Component Object Model (Microsoft®)
DSBC	Desarrollo de Software Basado en Componentes
DSIE	División de Sistemas e Ingeniería Electrónica
DSOA	Desarrollo de Software Orientado a Aspectos
DSP	Digital Signal Processor
EJB	Enterprise JavaBeans (Sun MicroSystems®)
EJBC	EJB Container
FPGA	Field Programmable Gate Array
GCC	GNU Compiler Collection
GIS	Geographical Information System

GP	Generative Programming
GPS	Global Positioning System
HCI	Human-Computer Interaction
HDL	Hardware Description Language
Hw/Sw	Hardware/Software
IDL	Interface Description Language
IEEE	Institute of Electrical and Electronics Engineers
IP	Intentional Programming
IPL	Image Processing Library (Intel®)
IPP	Integrated Performance Primitives (Intel®)
JAI	Java Advanced Imaging (Sun Microsystems®)
JPEG	Joint Photographic Experts Group Estándar ISO/IEC para la codificación de imágenes.
LOU	Ley Orgánica de Universidades
LPS	Línea de Productos Software (ver SPL)
LPS-VIPS	Línea de Productos Software de la familia de los VIPS
LPV	Lenguaje de Programación Visual
LRU	Ley de Reforma Universitaria
MDA	Model-Driven Architecture
MIDL	Microsoft® Interface Description Language
MIL	Matrox® Imaging Library (Matrox®)
MPEG	Moving Picture Experts Group. Estándar ISO/IEC para la codificación de audio y vídeo.
OCR	Optical □nálisis Recognition
OMA	Object Management Architecture
OMG	Object Management Group
OOSD	Object-Oriented Software Development
OpenCV	Open Computer Vision (Intel®)
ORB	Object Request Broker
OSCI	Open SystemC™ Initiative
PDA	Personal Digital Assistant
PDL	Programmable Logic Device
PG	Programación Generativa (ver GP)
PUI	Perceptual User Interface
PuLSE™	Product Line Software Engineering

PV	Programación Visual
SDL	System-level Design Language
SEI	Software Engineering Institute
SIVA	Sistema de Inspección Visual Automatizada
SPL	Software Product Line
UI	User Interface
UML	Unified Modelling Language
UPCT	Universidad Politécnica de Cartagena
VHDL	VHSIC HDL (Very High-Speed Integrated Cirtuit Hardware Description Language)
VIPS	Visual Information Processing Systems
VP	Visual Programming

Bibliografía

[Alencar 91] A. Alencar, J. A. Goguen, "OOZE: An Object Oriented Z Environment", Proc. European Conference on Object-Oriented Programming (ECOOP'91), Vol. 512, pp.180-199, 1991.

[Álvarez 04] B. Álvarez, J. A. Pastor, P. Sánchez, F. Ortiz "An architectural framework to manage the variability in a service robots product line", IX Jornadas de Ingeniería del Software y Bases de Datos (JISBD), Malaga, Noviembre 2004.

[AHEAD] AHEAD, Algebraic Hierarchical Equations for Application Design. Información sobre la herramienta disponible en:
http://www.cs.utexas.edu/users/schwartz/ATS.html

[Albert 02] C. Albert, L. Brownsword, "Evolutionary Process for Integrating COTS-Based Systems (EPIC): An Overview", tech. report CMU/SEI-2002-TR-009, Software Engineering Institute, Carnegie Mellon University, 2002.

[Amedi 05] A. Amedi, P. Meijer, "Neural correlates of visual-to-auditory sensory substitution in proficient blind users", International Multisensory Research Forum (IMRF'05), June 5-8, Rovereto, Italy, 2005.

[ANATQUEST] AnatQuest: Anatomic images on-line, The Visible Human Project. Información disponible en: http://anatquest.nlm.nih.gov/

[Angulo 03] J. Angulo, G. Flandrin, "Automated detection of working area of peripheral blood smears using mathematical morphology", Analytical Cellular Pathology, Vol. 25, Nº 1, p. 37-49, 2003.

[Atkinson 01] Atkinson et. al., "Component-Based Product Line Engineering with the UML", Addison-Wesley, 2001.

[Bachmann 00] Bachmann, F. et al, "Technical Concepts on Component-Based Software Engineering, 2nd Edition", CMU/SEI-2000-TR-008, Carnegie Mellon Software Engineering Institute, 2000.

[Balzer 85] R. Balzer, "A 15 Years Perspective on Automatic Programming", IEEE Trans. Software Engineering, Vol. 11, No. 11, pp. 1257-1268, Noviembre, 1985.

[Bass 98] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley, 1998.

[Bass 00] L. Bass, et al., "Fourth Product Line Practice Workshop Report", Tech. Report CMU/SEI-2000-TR-002, Carnegie Mellon Software Engineering Institute, 2000.

[Bayer 01] J. Bayer, D. Muthig, B. Göpfert, "The Library Systems Product Line: A KobrA Case Study", Frauhofer IESE Report No. 024.01/E, November 2001.

[Benkrid 04] A. Benkrid, K. Benkrid, D. Crookes, "Design and Implementation of Novel FIR Filter Architecture for Efficient Signal Boundary Handling on Xilinx VIRTEX FPGAs", IEEE Annual Symposium on VLSI (ISVLSI), 2004.

[Bertozzi 02] M. Bertozzi, et al., "Artificial Vision in Road Vehicles", Proc. Of the IEEE, Vol. 90, No. 7, pp. 1258-1271, Julio 2002.

[Bhandarkar 99] S. M. Bhandarkar, T. D. Faust, M. Tang, "CATALOG: a system for detection and rendering of internal log defects using computer tomography", Machine Vision and Applications, Vol. 11, Nº 4, pp. 171-190, 1999.

[Boehm 88] B. Boehm "A Spiral Model of Software Development and Enhancement", IEEE Computer, vol. 21, No. 5, pp. 61-72, 1988.

[Brown 99] A. Brown, "Constructing Superior Software", Capítulo 6: Building Systems from Pieces: Principles and Practice of Component-based Software Engineering, Macmillan Technical Publishing.

[Brzakovic 97] D. P. Brzakovic, et al., "A Generalized Development Environment for Inspection of Web Materials", Proc. IEEE International Conference on Robotics and Automation, Albuquerque (USA), 1997.

[Burnett 99] M. Burnett, "Visual Programming". In Encyclopedia of Electrical and Electronics Engineering (John G. Webster, ed.), John Wiley & Sons Inc., 1999.

[Chaudhari 03] U.V. Chaudhari, J. Navratil, S.H. Maes, "Multi-Grained □odelling with pattern specific maximum likelihood transformations for text-independent speaker recognition," IEEE Transactions on Speech and Audio Processing, Vol. 11, No. 1, pp.61-69, 2003.

[Chastek 02] G. Chastek, J. D. McGregor, "Guidelines for Developing a Product Line Production Plan", Tech. Report CMU/SEI-2002-TR-006, Carnegie Mellon Software Engineering Institute, 2002.

[Chen 01] T. Q. Chen, et al., "A Smart Machine Vision System for PCB Inspection", Proc.14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, LNCS Vol. 2070, pp. 513-518, 2001.

[Chin 82] R. T. Chin, C. A. Harlow, "Automated Visual Inspection – A Survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 4, Nº. 6, pp. 557-573, 1982.

[Clements 02] P. Clements, L. Northrop, "Software Product Lines: Practices and Patterns", Addison-Wesley, 2002.

[Cockburn 01] A. Cockburn, "Writing effective Use Cases", Addison-Wesley, 2001.

[Coleman 94] D. Coleman, et al., "Object Oriented Development: The Fusion Method", Prentice-Hall, 1994.

[CORBA] CORBA: Common Object Request Broker Architecture, Información disponible en: http://www.corba.org/

[Crnkovic 02] I. Crnkovic, et al., "Building Reliable Component-Based Software Systems", Chapter 1: Basic Concepts in Component-Based Software Engineering, Artech House Publishers, Julio 2002.

[Czarnecki 00] K. Czarnecki, U. W. Eisenecker, "Generative Programming: Methods, Tools, and Applications", Addison-Wesley, 2000.

[Daugman 93] J. G. Daugman, "High confidence visual recognition of persons by a test of statistical inde-pendence", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, Nº 11, pp.1148–1161, 1993.

[DeSouza 02] G. N. DeSouza, A. C. Kak, "Vision for Mobile Robot Navigation: A Survey", IEEE Transactions on Pattern Recognition and Machine Ingelligence, Vol. 24, Nº 2, pp. 237-267, 2002.

[DoD 91] Department of Defense Software Technology Strategy (draft), Washington D.C.:Department of Defense, December 1991.

[Duda 00] R. O. Duda, P. E. Hart, D. G. Stork, "Pattern Classification (2nd Edition)", Wiley-Interscience, 2000.

[Eisenecker 03] U. W. Eisenecker, et al., "Emerging Product Line Implementation Technologies: C++, Frames, and Generating Graphical User Interfaces", PoLITe Project Technical Report, disponible en:
http://www.polite-project.de/publications.html

[Everding 94] B. S. Everding, et al, "Generalization of an Automated Visual Inspection System", SAE Aerospace Atlantic Conf., 1994.

[Fernández 97] C. Fernández, "Inspección de superficies mediante Visión Artificial: Integración en tiempo real en procesos productivos", Tesis Doctoral, Universidad Politécnica de Madrid, 1997.

[Fernández 99] C. Fernández, J. Suardíaz, A. Iborra, J. M. Fernández Meroño, "On-line Automated Visual Inspection for Quality Control within the Automobile Industry", Proc. 5th International Conference on Quality Control by Artificial Vision, (QCAV), 2001.

[Fernández 01] C. Fernández, et al., "Automated Visual Inspection within the Industry of Preserved Vegetables", 5th Quality Control by Artificial Vision International Conference (QCAV'2001), Le Creusot (Francia), 2001.

[Fernández 05] C. Fernández, A. Iborra, B. Álvarez, J. A. Pastor, P. Sánchez, J. M. Fernández, "Cooperative Robots for Hull Blasting in European Ship Repair Industry", IEEE Robotics & Automation Magazine, Vol. 12, No. 3, 2005.

[Forsyth 02] D. Forsyth, J. Ponce, "Computer Vision: A Modern Approach", Prentice Hall, 2002.

[Gamma 95] E. Gamma, et al., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Professional, 1995.

[García 00] G. García-Mateos, C. Vicente-Chicote, "A new model and process architecture for facial expression recognition", Proc. Joint IAPR Int'l Workshops SSPR 2000 and SPR 2000, Lecture Notes in Computer Science (LNCS), Vol. 1876, pp. 716-722, 2000.

[García 01ª] G. García-Mateos, C. Vicente-Chicote, "Face Detection on Still Images Using HIT Maps", 3rd Int'l Conf. Audio- and Video-based Biometric Person Authentication (AVBPA), Lecture Notes in Computer Science (LNCS), Vol. 2091, pp. 102-107, 2001.

[García 01b] G. García-Mateos, C. Vicente-Chicote, "A Unified Approach to Face Detection, Segmentation and Location Using HIT Maps", Proc. IX Simposium Nacional de Reconocimiento de Formas y Análisis de Imágenes (SNRFAI), pp. 61-66, 2001.

[García 02] G. García-Mateos, C. Vicente-Chicote, A. García-Meroño, "Localización y Seguimiento de caras usando búsqueda en rejilla", Proc. Conf. Iberoamericana en Sistema, Cibernética e Informática (CISCI), pp. 109-114, 2002.

[García 05ª] G. García-Mateos, A. García-Meroño, C. Vicente-Chicote, A. Ruiz, P.E. López-de-Teruel, "Time and Date OCR in CCTV Video", Proc. 13th Int'l Conf. Image Analysis and Processing (ICIAP), Lecture Notes in Computer Science (LNCS), Vol. 3617, pp.703-710, 2005.

[García 05b] G. García-Mateos, S. Fructuoso-Muñoz, "Tierra Inhospita: Exploring a Virtual World with your Face", ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2005), Valencia, Spain, 15-17 June, 2005.

[Garlan 95] D. Garlan, D. Perry, Introduction to the special issue on Software Architecture, IEEE Transactions on Software Engineering, Vol. 21, No. 4, April, 1995.

[González 96] J. R. González, H. Pereira, "Classification of defects in cork planks using image analysis techniques", Wood Science & Technology, Vol. 30, pp. 207-215,1996.

[González 02] R. C. González, R. E. Woods, "Digital Image Processing (2nd Edition)", Prentice-Hall, 2002.

[Gorodnichy 04] D. O. Gorodnichy, G. Roth., "Nouse 'Use your nose as a mouse' perceptual vision technology for hands-free games and interfaces", Image and Vision Computing, Vol. 22, Issue 12 , 931-942, 2004.

[Grasby 04] B. Grassby, "A smarter computer controlled model car", Honours Thesis, School of Computer Science and Software Engineering, Monash University, Noviembre 2002.

[GRASS] Geographic Resources □nálisis Support System, información del producto disponible en: http://grass.itc.it/index.php

[Green 95] T. Green, "Noddy's Guide to Visual Programming", Newsletter of the British Computer Society Human-Computer Interaction Group, 1995.

[Greenfield 04] J. Greenfield, et al., "Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools", Addison-Wesley, 2004.

[Griss 98] M. Griss, J. Favro, M. d'Alessandro, "Integrating Feature Modeling with the RSEB", Proc. 5th International Conference on Software Reuse, pp. 76-85, Vancouver, BC, Canada, June 1998.

[Gomma 04] H. Gomma, "Designing Software Product Lines with UML : From Use Cases to Pattern-Based Software Architectures", Addison-Wesley Professional, 2004.

[HANDELC] Celoxica® HandelC, información del producto disponible en:
http://www.celoxica.com/

[Hazan 03] E. Hazan, L. Joskowicz, "Computer-assisted image-guided intramedullary nailing of femoral shaft fractures", Techniques in Orthopaedics , Special Issue on Computer-Aided Orthopaedic Surgery, Vol. 18, No. 2, 2003.

[Ibarra 95] F. Ibarra, D. Asensi, A. Almagro. Segmentation of Defect in Textile Fabric Using Semi-Cover vector and Self-Organization. Proceedings of the International Conference on Quality Control by Artificial Vision. France, 1995.

[IBM-WEBSPHERE] WebSphere – Technical resources for the WebSphere software platform, Información del producto disponible en:
http://www-130.ibm.com/developerworks/websphere/

[IEEE-Std-1471] The IEEE 1471-2000 Standard, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems". The Institute of Electrical and Electronics Engineers (IEEE), Inc. New York, USA, Septiembre 2000.

[I-GLASSES] I-Glasses™ Video 3D Pro, información del producto disponible en: http://www.i-glassesstore.com/ig-hrvpro.html

[IPP] Intel® Integrated Performance Primitives, información disponible en: http://www.intel.com/cd/software/products/asmo-na/eng/perflib/ipp

[Iribarne 02] L. Iribarne, J. M. Troya, A. Vallecillo, "Selecting Software Components with Multiple Interfaces", Proc. 28th Euromicro Conference, pp. 26-32, IEEE CS Press, September, 2002.

[Iribarne 03] L. Iribarne, "Un Modelo de Mediación para el Desarrollo de Software basado en Componentes COTS", Tesis Doctoral, Universidad de Almería, 2003.

[Iribarne 04] L. Iribarne, J. M. Troya, A. Vallecillo. "A Trading Service for COTS Components", The Computer Journal, Vol. 47, No. 3, pp. 342-357, 2004.

[ISO/IEC-9126 91] International Standard ISO/IEC 9126. Information Technology – Software Product Evaluation – Quality Characteristics and Guidelines for their Use. Geneve, Switzerland, 1991.

[Jacobson 92] I. Jacobson, et al., "Object-Oriented Software Engineering: A Use Case Driven Approach", Addison Wesley, 1992.

[JAI] Sun Microsystems® Java Advanced Imaging API, Información del producto disponible en: http://java.sun.com/developer/releases/jai/

[Jain 03] A. K. Jain, " Biometrics definition", World Book Online Reference Center 2003, disponible en: http://www.worldbookonline.com/ar?/na/ar/co/ar750838.htm

[Jain 04] A. K. Jain, et al., "Biometrics: A grand challenge", Proc. International Conference on Pattern Recognition, Cambridge (Reino Unido), 23-26 Agosto, 2004.

[JPEG] Estándar JPEG. Información disponible en: http://www.jpeg.org/

[Kang 90] K. C. Kang, et al., "Feature-oriented Domain Analisys (FODA) Feasibility Study", Tech. Report CMU/SEI-90-TR-21, ESD-90-TR-222, Carnegie Mellon Software Engineering Institute, 1990.

[Kang 98] K. C. Kang, et al., "FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures." Annals of Software Engineering, Vol. 5, No. 5, pp. 143-168, September 1998.

[Kang 02] K.C. Kang, J. Lee, P. Donohoe, "Feature-Oriented Product Line Engineering", IEEE Software, 19 (4), pp. 58-65, July/August 2002.

[Kettemann 03] S. Kettemann, D. Muthig, M. Anastasopoulos, "Product Line Implementation Technologies: Component Technology View", IESE-Report No. 015.03/E, March 2003.

[Kim 99] T. H. Kim, et At., "Visual inspection system for the classification of solder joints", Pattern Recognition, Vol. 32, Nº 4, pp. 565-575, 1999.

[Ko 00] K. W. Ko, H. S. Cho, "Solder joints inspection using a neural network and fuzzy rule-based classification method", IEEE Transactions on Electronics Packaging and Manufacturing, Vol. 23, Nº2, pp. 93-103, 2000.

[KOBRA] Metodología Kobra (Frauhofer IESE), información disponible en: http://www.iese.fhg.de/Kobra_Method/

[Kwak 00] C. Kwak, J. A. Ventura, K. Tofang-Sazi, "A neural network approach for defect identification and classification on leather fabric", Journal of Intelligent Manufacturing, Vol. 11, Nº 5, pp. 485-499, 2000.

[LABVIEW] Nacional Instruments™ LabView®, información del producto disponible en: http://www.ni.com/labview/

[Li 05] S. Z. Li and A. K. Jain (Eds.), "Handbook of Face Recognition", Springer, 2005.

[LOCUST] Proyecto LOCUST: Life-like Object Detection for Collision Avoidance Using Spatiotemporal Image Processing (IST 2001-38097), información del proyecto disponible en: http://www.imse.cnm.es/locust/

[López 03] P. E. López de Teruel Alcolea, "Una Arquitectura Eficiente de Percepción de Alto Nivel: Navegación Visual para Robots Autónomos en Entornos Estructurados", Tesis Doctoral, Universidad de Murcia, 2003.

[Lucas 04] F. J. Lucas, et al., "Maude como Soporte Formal para una Herramienta de Gestión de Modelos", II Jornadas de Trabajo DYNAMICA, junto a las Jornadas de Ingeniería del Software y Bases de Datos (JISBD'04), Málaga, 2004.

[Maltoni 03] D. Maltoni, et al., "Handbook of Fingerprint Recognition", Springer, 2003.

[Mamic 00] G. Mamic, M. Bennamoun, "Automatic flaw detection in textiles using a Neyman-Pearson detector", Proc. 15th Conference on Pattern Recognition, Vol. 4, pp. 767-770, 2000.

[MATLAB IAT] Matlab® Image Acquisition Toolbox, información del producto disponible en: http://www.mathworks.com/products/imaq/

[MATLAB IPT] Matlab® Image Processing Toolbox, información del producto disponible en: http://www.mathworks.com/products/image/

[MATLAB] Matlab® Información del producto disponible en:
http://www.mathworks.com/products/matlab/

[MAUDE] The Maude System, información disponible en:
http://maude.cs.uiuc.edu/

[Mery 03] D. Mery, M. A. Berti, "Automatic detection of welding defects using texture features", International Symposium on Computed Tomography and Image Processing for Industrial Radiology, Berlin, June 23-25, 2003.

[MIL] Matrox® Imaging Library, información disponible en:
http://www.matrox.com/imaging/products/mil/

[MPEG] Estándar MPEG. Información disponible en: http://www.mpeg.org

[Muthig 02] D. Muthig, et al., "Technology Dimensions of Product Line Implementation Approaches: State-of-the-art and State-of-the-practice Survey", IESE-Report No. 051.02/E, September 2002.

[Muthig 04] D. Muthig, et al., "GoPhone - A Software Product Line in the Mobile Phone Domain", IESE-Report No. 025.04/E, March 2004.

[Nalwa 93] [Nalwa93] V. S. Nalwa, "A guided tour of computer vision", Addison & Wesley, 1993.

[Newman 95] T. S. Newman, A. K. Jain, "A survey of automated visual inspection", Computer Vision and Image Understanding, Vol. 6, N° 2, pp. 231-262, 1995.

[Newman 00] W. S. Newman, et al., "Design Lessons for Building Agile Manufacturing Systems", IEEE Transactions on Robotics and Automation, Vol. 16, No. 3, pp. 228-238, Junio 2000.

[Nicolás 96] J. Nicolás, et al., "Análisis y diseño con el lenguaje de especificación formal orientado a objetos OASIS de un microscopio efecto-túnel", II Jornadas sobre Tecnología de Objetos (SIMO 96), Madrid, Noviembre, 1996.

[Noordam 00] J. C. Noordam, et al., "High Speed Potato Grading and Quality Inspection Based on a Color Vision System", Proc. VIII SPIE Conference on Machine Vision Applications in Industrial Inspection, San José, California (USA), 2000.

[OPENCV] Intel Open Computer Vision Library, Información disponible en:
http://www.intel.com/research/mrl/research/opencv/

[Ortiz 03] F. Ortiz, B. Álvarez, J. A. Pastor, P. Sánchez, "A Case Study in Performance Evaluation of Real-Time Teleoperation Software Architectures Using UML-MAST", 8th Ada-Europe International Conference, Lecture Notes in Computer Science, Vol. 2655, pp. 417-428, 2003.

[Ortiz 05] F. Ortiz, "Arquitectura de Referencia para unidades de control de robots de servicio teleoperados", Tesis Doctoral, Univ. Politécnica de Cartagena, 2005.

[Page-Jones 88] M. Page-Jones, "Practical Guide to Structured Systems Design (2nd Edition)", Prentice-Hall, 1988.

[Pastor 04] J. A. Pastor, B. Álvarez, P. Sánchez, F. Ortiz, "A layered architectural component model for service teleoperated robots", IX Jornadas de Ingeniería del Software y Bases de Datos (JISBD), Malaga, Noviembre 2004.

[PCMEncyclopedia PC Magazine Encyclopedia, System Development Methodology Definition, información disponible en: http://www.pcmag.com/encyclopedia/

[Perrier 04] V. Perrier, "A look inside Electronic System Level (ESL) design", EEDesign.com, Art. Id. 18402916, 2004.

[PULSE] Metodología PuLSE™ (Frauhofer IESE), información disponible en: http://www.iese.fhg.de/PuLSE/

[Qureshi 05] S. Qureshi, "Embedded Image Processing on the TMS320C6000™ DSP: Examples in Code Composer Studio™ and MATLAB", Springer , 2005.

[Recce 96] M. Recce, et al., "High Speed Vision-Based Quality Grading of Oranges". International Workshop on Neural Networks for Identification, Control, Robotics and Signal/Image Processing, Venecia (Italia), 1996.

[Ribeiro 00] A. Ribeiro, et al., "Automatic Rules Generation By G.A. For Eggshell Defect Classification". European Congress on Computational Methods in Applied Science and Engineering (ECCOMAS'00), Barcelona (España), 2000.

[Rivas 05] S. Rivas, R. Servent, "Automated Spot Weld inspection in the Automotive Industry", The e-Journal of Nondestructive Testing, Vol. 10, Nº 3, Marzo 2005, disponible en: http://www.ndt.net/v10n03.htm

[Rosenblatt 62] Rosenblatt, F. "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms", Spartan Books, Washington DC, 1962.

[Ruiz 94] Ruiz Vargas, J. M., "La memoria humana. Función y estructura.", Alianza Psicología Minor, 1994.

[RUP] RUP: Rational Unified Process, Información disponible en: http://www-306.ibm.com/software/awdtools/rup/

[Sankaran 98] V. Sankaran, A. R. Kalukin, R. P. Kraft, "Improvements to X-ray laminography for automated inspection of solder joints", IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C, Vol. 21, Nº 2, pp. 148-154, 1998.

[Savolainen 01] J. Savolainen, et. al., "Feature Analysis", ESAPS Project, 2001. Información disponible en: www.esi.es/en/Projects/esaps/esaps.html

[Schael 01] M. Schael, "Texture Defect Detection Using Invariant Textural Features", Lecture Notes in Computer Science, Vol. 2191, pp. 17-24, 2001.

[Scheiman 94] Scheiman M. M., Rouse M. W (eds), "Optometric Management of Learning-Related Vision Problems", CV Mosby, 1994.

[Schumacher 04] M. Schumacher, "Assembly control of vehicle aggregates", International AUTOMATICA Machine Vision FORUM, track on Machine Vision in the Automotive Industry, Munich (Alemania), 15-17 Junio, 2004.

[Shapiro 01] L. Shapiro et al., "Computer Vision", Prentice Hall, 2001.

[SHERLOCK] Herramienta Sherlock de DALSA Coreco®, información disponible en: http://www.goipd.com/products/Sherlock/

[SIMULINK] Información del producto disponible en: http://www.mathworks.com/products/simulink/

[Sirta 97] J. A. Sirta et al., "Advanced Image Processing Tools for Future Satellite Image Exploitation Systems", Malasia, 20-24 Octubre, 1997.

[Sommerville 04] I. Sommerville, "Software Engineering (7th Edition)", Addison Wesley, 2004.

[Štuikys 02] V. Štuikys, R. Damaševičius, "Relationship Model of Abstractions Used for Developing Domain Generators", INFORMATICA, Vol. 13, No. 1, 2002.

[SYSTEMGENERATOR Xilinx® System Generator®, información del producto disponible en: http://www.xilinx.com/products/software/sysgen/app_docs/user_guide.htm

[SYSTEMC] SystemC, información del producto disponible en: http://www.systemc.org/

[Szyperski 98] C. Szyperski, "Component Software. Beyond Object-Oriented Programming", Addison-Wesley, 1998.

[Toledo 05a] A. Toledo-Moreo, "Entorno de Codiseño para Sistemas de Procesamiento de Imágenes", Tesis Doctoral, Univ. Politécnica de Cartagena, 2005.

[Toledo 05b] A. Toledo-Moreo, C. Vicente-Chicote, J. Suardíaz-Muro, S. Cuenca-Asensi, "Xilinx System Generator Based HW Components for Rapid Prototyping of Computer Vision SW/HW Systems.", Proc. 2nd Iberian Conf. on Pattern Recognition and Image Analysis (IbPRIA), Lecture Notes in Computer Science (LNCS), Vol. 3522, pp.667-674, 2005.

[Trigaux 03] J. C. Trigaux and P. Heymans, "Modelling variability requirements in Software Product Lines: A comparative survey", Tech. rep. PLENTY Project, Institut d'Informatique FUNDP, November, 2003.

[Trucco 98] E. Trucco, A. Verri, "Introductory Techniques for 3-D Computer Vision", Prentice Hall, 1998.

[UML] Especificación de la notación UML (Unified Modeling Language™) realizada por el OMG (Object Management Group), información disponible en: http://www.uml.org/

[van Gurp 01] G. van Gurp, J. Bosch, M. Svahnberg, "On the Notion of Variability in Software Product Lines", Proc. Working IEEE/IFIP Conf. on Software Architecture (WICSA), 2001.

[Vergés 02] C. Vergés Roger, "La percepción visual. Qué vemos y cómo vemos", Microcirugía Ocular, Nº 4, Dic 2002. http://www.oftalmo.com/secoir/secoir2002/rev02-4/02d-02.htm

[VERILOG] Verilog® HDL de Cadence® Design Systems Inc., información del producto disponible en: http://www.verilog.com/

[VHDL] VHDL, IEEE Standard 1076, información disponible en: http://standards.ieee.org/reading/ieee/std_public/description/dasc/1076-1993_desc.html

[VHProject] The Visible Human Project, National Library of Medicine (USA). Información disponible en: http://www.nlm.nih.gov/research/visible/visible_human.html

[Vicente 02] C. Vicente-Chicote, G. García-Mateos, A. García-Meroño, "Seguimiento de caras humanas mediante búsqueda logarítmica en rejilla", Proc. III Jornadas Regionales de Informática Gráfica (JRIG), pp. 75-86, 2002.

[Vicente 04a] C. Vicente-Chicote, C. Fernandez-Andrés, P. Sánchez-Palma, "Desarrollo de Sistemas de Inspección Visual Automatizada a partir de la Descripción de un Patrón Arquitectural Genérico", NOVATICA, No. 171, pp. 63-65, 2004.

[Vicente 04b] C. Vicente-Chicote, C. Fernandez-Andrés, P. Sánchez-Palma, "Sistemas de Inspección Visual Automatizada: De la Arquitectura Software Genérica a la Generación de Prototipos Ejecutables", Proc. IX Jornadas de Ingeniería del Software y Bases de Datos (JISBD'04), pp. 515-522, 2004.

[Vicente 05a] C. Vicente-Chicote, A. Toledo-Moreo, C. Fernandez-Andrés, "Heterogeneous COTS Product Integration to Allow the Comprehensive Development of Image Processing Systems", Proc. IV Int'l Conf. on COTS-Based Software Systems (ICCBSS), Lecture Notes in Computer Science (LNCS), Vol. 3412, pp. 8, 2005.

[Vicente 05b] C. Vicente-Chicote, A. Toledo-Moreo, P. Sánchez-Palma, "Image Processing Application Development: From Rapid Prototyping to SW/HW Co-Simulation and Automated Code Generation", Proc. 2nd Iberian Conf. on Pattern Recognition and Image Analysis (IbPRIA), Lecture Notes in Computer Science (LNCS), Vol. 3522, pp. 659-666, 2005.

[Vicente 05c] C. Vicente-Chicote, A. Toledo-Moreo, P. Sánchez-Palma, C. Fernández-Andrés, "Generación Automática de Aplicaciones Mixtas Sw/Hw mediante la Integración de Componentes COTS", Proc. X Jornadas de Ingeniería del Software y Bases de Datos (JISBD'05), 2005.

[Viebig 04] K. Viebig, "Application examples of image analysis systems in the foundry of an automotive plant", International AUTOMATICA Machine Vision FORUM, track on Machine Vision in the Automotive Industry, Munich (Alemania), 15-17 Junio, 2004.

[VISIQUEST] AccuSoft Corporation™ VisiQuest®, Información del producto disponible en: http://www.accusoft.com/imaging/visiquest/

[Vitrià 05] J. Vitrià, M. Bressan and P. Radeva, "Bayesian classification of cork stoppers using class-conditional independent component analysis", IEEE Transactions on Systems, Man, and Cybernetics (in press).

[Wählby 03] C. Wählby , "Algorithms for Applied Digital Image Cytometry", PhD Thesis, Uppsala University (), 2003.

[Wallnau 01] K. C. Wallnau, S. A. Hissam, R. C. Seacord, "Building Systems from Commercial Components", The SEI Series in Software Engineering, Addison-Wesley, 2001.

[Whitney 96] J. Whitney, "Investment Analysis of Software Assets for Product Lines", (CMU/SEI-96-TR-010), Software Engineering Institute (SEI), Carnegie Mellon University, 1996.

[Winter 05] S. Winter, B. Brendel, C. Igel, "Registration of bone structures in 3D ultrasound and CT data: Comparison of different optimization strategies", Proc. Computer Assisted Radiology and Surgery (CARS), International Congress Series, Vol. 1281, pp. 242-247, 2005.

[WIT] Herramienta WIT de DALSA Coreco®, información disponible en: http://www.coreco.com/wit