Centro De Investigaciones Y Estudios Avanzados

Del Instituto Politécnico Nacional

Departamento de Ingeniería Eléctrica

Sección de Computación

Método de Desarrollo Arquitectónico en Grupo

Tesis que presenta

Moisés González García

para obtener el Grado de

Doctor en Ciencias

En la Especialidad de

Ingeniería Eléctrica

Directora de la Tesis:

Ana María Antonia Martínez Enríquez

México, D.F. Agosto del 2006

Centro De Investigaciones Y Estudios Avanzados

Del Instituto Politécnico Nacional

Department of Electrical Engineering

Computer Science Section

Architectural and Group Development Method

Thesis presented by:

Moisés González-García

to obtain the grade of

Doctor es Science

Electrical Engineering

Speciality

Advisor:

Ana María Antonia Martínez Enríquez

México, D.F. August, 2006

RESUMEN

Nuestra investigación se interesa en la contribución al desarrollo de software, del trabajo realizado por grupos de personas y el enfoque dirigido por modelos. La meta principal de este trabajo es crear un método de elaboración que asigne la mayor importancia al producto de software y su estructura, asociando un proceso de grupo, a cada uno de los productos-del-trabajo requeridos para construir el resultado.

Definimos el método AGD (Desarrollo Arquitectónico en Grupo) para desarrollar software, que utiliza un conjunto preestablecido de productos-del-trabajo (PS) y asocia a cada producto-del-trabajo un proceso dirigido por modelos (MDP). AGD obtiene cualquier producto-del-trabajo mediante una transformación de un modelo fuente a un modelo objetivo, o realizando en equipo un conjunto de sub-procesos. El software se desarrolla por grupos de personas colaborando, los productos se revisan en reuniones técnicas por colegas o expertos y el desempeño se monitorea en reuniones de seguimiento del equipo.

PS abarca los productos de las cuatro categorías de áreas de proceso que incluye la Integración del Modelo de Madurez de Capacidad (CMMI): 1) Administración de Proceso, 2) Administración de Proyecto, 3) Ingeniería y 4) Soporte. Los conjuntos de productos-del-trabajo de la categoría de Ingeniería, abarcan las representaciones necesarias para desarrollar el producto de software.

AGD utiliza, en la categoría de Ingeniería, un conjunto de productos-del-trabajo con los conceptos de los modelos de la Arquitectura Dirigida por Modelos (MDA): Modelos Independientes del Cómputo, Modelos Independientes de la Plataforma y Modelos de Plataforma Específica. Complementando los modelos de la MDA con los conjuntos: Modelos de Implementación y Modelos para Operación. Inicialmente los productos-del-trabajo de Ingeniería se obtienen mediante el proceso MDP, pero conforme se habiliten transformaciones, se podrá elegir obtenerlos automáticamente o mediante MDP.

MDP realiza en equipo principalmente los sub-procesos: requerimientos, análisis & diseño, implementación y prueba, que iteran continuamente de manera flexible, para obtener cada producto-del-trabajo. El software se obtiene incrementalmente en forma disciplinada, monitoreándose peculiaridades del producto y características del proceso de manera semejante a como se realiza en el "Proceso de software en equipo" (TSP). El método AGD establece que el desarrollo se lleva acabo en forma: grupal, incremental, cooperativa, adaptable y directa.

ABSTRACT

Our research concerns the contribution of work done by groups of persons, and model driven approaches for software development. The main goal of this work is to develop a method that gives the main role to the software product and its structure, associating a group-process to each component of the result.

We defined the Architectural and Group Development method (AGD) for software development, using a predefined set of work-products (Products Set, PS), and each work-product is assoiated with a Model Driven Process (MDP). AGD produces any work-product either by a transforming a target model from a source one, doing a set of sub-processes by a team. Software is developed by groups of persons collaborating, products are reviewed in technical meetings by pears or experts, and performance is monitored in follow-up meetings by team members.

PS contains the work-products of the four categories of process areas, used in the Capability Maturity Model Integration (CMMI): 1) Process Management, 2) Project Management, 3) Engineering, and 4) Support. The work-products of the Engineering category cover every required model to develop a software product.

AGD uses, in the Engineering category, a set of work-products similar to the mo-dels defined in the Model Driven Architecture (MDA): Computation Independent Model, Platform Independent Model, and Platform Specific Model. Complementing those models with the: Implementation Model, and Operation Model. Initially the work-products are obtained by the process MDP, but as transformations become available, developers will chose to get models automatically or using MDP.

MDP performs by a team, mainly, the sub-procesees: requirenments, analisys & design, implementation, and test, iterating to construct each work-product. The software product is obtaineded incrementally with discipline, monitoring product peculiarities and process characteristics, like is done in the Team Software Process (TSP). AGD stablish that development is performed: by groups, incrementall, cooperative, adaptive, and direct.

AGRADECIMIENTOS

A mis padres Ana María García Gama y José González Sandoval, porque me amaron y enseñaron a amar, proporcionándome la estabilidad y confianza necesarias para vivir plenamente.

A mi esposa Martha Montiel Rentería, por su paciencia y acicate para el logro de mis objetivos.

A mis hijos Moisés, Martha Angélica, Ana Luz y Aarón, por darme un motivo para capacitarme y por ser fuente del amor necesario para continuar. Pidiéndoles que siempre estén pendientes de su comportamiento y habilidades, dispuestos a mejorar continuamente.

A la Doctora Ana María Martínez Enríquez, por su perseverancia y trabajo infatigable, realizado en la dirección de la tesis.

A los directivos y compañeros del CENIDET[1], por darme la oportunidad de emprender esta investigación. Especialmente a los profesores René Santaolaya Salgado y Mario Guillén, así como a los alumnos que colaboraron en la experimentación realizada.

Al CONACyT[2], CENIDET y CINVESTAV, por proporcionar los recursos necesarios para realizar los estudios Doctorales y esta investigación.

A los alumnos de la maestría con especialidad en ingeniería de software, en la Universidad Autónoma de Querétaro, por la retroalimentación de los resultados obtenidos al poner en práctica los productos de la investigación.

A los miembros de la Asociación de la Industria del Software (AISAC), por haberme permitido interactuar e intercambiar ideas, así como por su colaboración y retroalimentación indispensables para mejorar los resultados.

A los colegas de la Asociación Mexicana de Calidad en Ingeniería de Software (AMCIS), por compartir sus conocimientos, facilitando la adquisición de habilidades y conceptos, útiles para nuestra investigación.

[1] CENIDET – Centro Nacional de Investigación y Desarrollo Tecnológico
[2] CONACyT – Consejo Nacional de Ciencia y Tecnología

TABLA DE CONTENIDO

LISTA DE FIGURAS

Número Página

LISTA DE TABLAS

Número Página

NOTACIÓN

La tabla siguiente muestra la notación utilizada en este documento.

Texto	Arial de tamaño 12.- Redacción normal de la tesis (en español)
Texto en Español / English Text (ej. Producto-del-trabajo / Work-product)	Siempre que se incluye un texto en Inglés, se precede por su traducción en Español separada por una diagonal " / ".} (se presenta los téminos en **español / ingles**, porque todas las publicaciones acerca de esta investigación han sido en inglés)
work-product	Arial *cursiva* de tamaño 12.- Palabra reservada para conceptos, usados en el prototipo (en ingles)
Planning	Arial Negrita de tamaño 12.- Nombre de instancia de un concepto, usado en el prototipo (en inglés)

ABREVIATURAS GENERALES

etc. - etcétera	**íd.** - ídem	**nº** o **núm.** - número	**p.ej.** o **ej.** - por ejemplo
págs. - páginas	**§** - Sección		

ABREVIATURAS ESPECÍFICAS

ABC - Ciclo del Negocio para Arquitectura / Architecture Business Cycle
AGD – Método de Desarollo Arquitectónico en Grupo / Architectural Group Development method
CIM - Modelo Independiente de Cómputo /Computation Independent Model
CMM - Modelo de Madurez de la Capacidad Integrado / Capability Maturity Model Integrated
CS - Servicio Cooperativo / Cooperative Service
CSCW – Trabajo Cooperativo Soportado por Computadora / Computer Supported Cooperative Work
CWIS - Sistema de Información con Trabajo Cooperativo / Cooperative Work Information System
CWSLR - Modelo Trabajo Cooperativo Almacenamiento Lógica y Recursos / Cooperative Work Storage Logic and Resources model
CWSP - Ambiente Proceso de Software con Trabajo Coopetrativo / Cooperative Work Software Process environment

DBMS - Sistema manejador de base de datos / Data Base Management System
DGM - Mecanismo Dinámico Gráfico / Dynamic Graphic Mechanism
GQM - Técnica Meta Pregunta medición / Goal Question Metric technique
IEEE - Instituto de Ingenieros Eléctricos y Electrónicos / Institute of Electric and Electronic Engineers
IM - Modelo de Implementación / Implementation Model
LN - Redes locales / Local Networks
MCTC - Catálogo de Tipos de Componentes Mínimos / Minimal Component Type Component cathalog
MDA – Arquitectura Dirigida por Modelos / Model Driven Architecture
MDP - Proceso Dirigido por Modelos / Model Driven Process
MOF - Mmetamodelo Facilida Meta Objeto / Meta Object Facility methamodel
MUS - Sistema para varios usuario / Múltiple-User System
ODBMS - Sistema manejador de base de datos de objetos / Object Data Base Management System
OM - Modelo de Operación / Operation Model
OMG - Grupo de Administración de Objetos / Objetc Management Group
PIM - Modelo Independiente de Plataforma / Platform Independent Model
PS - Conjunto de Productos / Product Set
PSM - Modelo de Plataforma Específica / Platform Specific Model
PSnSet - Conjunto a nivel "n" del Product Set
RCL - Biblioteca de Componentes Reusables / Reusable Component Library
RN - Redes remotas / Remote Networks
RUP - Proceso Unificado de Rational / Rational Unified Process
RUS - Sistemas para utilitarios remotos / Remote Utility System
SEI - Instituto de Ingeniería de Software / Software Engineering Institute
SPEMS - Especificación del metamodelo para ingeniería del proceso de software / Software Process Engineering Metamodel Specification
SUS - Sistema para un usuario / Single-User System
TCP/IP - "Protocolo de control de transporte / Protocolo Internet" / "Transport Control Protocol / Internet Protocol"
TSP - Proceso de software en equipo / Team Software Process
UDF - Fólder de desarrollo de unidad / Unit Development Folder
UML - Lenguaje unificado de modelado / Unified Modeling Language
μ_A - Media de valores para equipos trabajando con el métododo AGD
μ_R - Media de valores para equipos trabajando con el proceso RUP

1 Introducción

La presente investigación esta ubicada en el área del desarrollo de software producido en grupo, acorde con la necesidad actual de calidad[3] del producto de software[4] y de desempeño del proceso de software[5]. Mediante los resultados de la investigación aportamos un enfoque alterno para el desarrollo de software que pretende obtener productos de mayor calidad y simplificar el proceso.

El objetivo principal de esta investigación es definir las características de un producto de software que considere la complejidad de las aplicaciones actuales y la plataforma tecnológica existente, así como especificar el proceso[6] adaptable para el desarrollo de la ingeniería[7] del producto de software especificado.

Analizando la definición de método del glosario [IEEE-610.12], identificamos que el postulado de nuestro objetivo equivale a establecer un método cuyo proceso se dirige a desarrollar los componentes de la arquitectura de software, mediante modos de trabajo-en-grupo, para obtener productos de software de mejor calidad (con menos defectos). El producto está representado mediante un conjunto de reglas que definen su estructura y las relaciones entre sus partes, es decir, se describe mediante la arquitectura de software [ODP98]. Utilizamos la arquitectura de software del producto para establecer una referencia, para dirijir el proceso de desarrollo.

Siguiendo esta estrategia se propuso como solución el método de Desarrollo Arquitectónico en Grupo / Architectural Group Development (AGD). AGD incluye el ciclo de vida del producto[8] [IEEE-610.12] utilizando la arquitectura del producto para asociar a cada uno de sus

---

[3] Calidad, es la habilidad de satisfacer los requerimientos del cliente mediante un conjunto de características inherentes a un producto, componente del producto o proceso [SEI-CMMI-C]. En la norma ISO/IEC 9126 -1, acerca de la calidad del producto [ISO-9126], se mencionan características y sub-características, agrupadas en dos categorías: 1) Calidad en uso y 2) Calidad externa e interna. La norma 9126 determina que las características que constituyen la calidad en uso son: Eficacia, Productividad, Seguridad y Satisfacción; mientras que las características tomadas en consideración para la calidad externa e interna son: Funcionalidad, Confiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad.

[4] Producto de Software, es el conjunto de programas de computadora, procedimientos, documentación y datos, asociados [IEEE-12207].

[5] Proceso de Software, es el proceso (o procesos), usado por una organización (o proyecto), para planificar, administrar, ejecutar, monitorear, controlar y mejorar sus actividades, relacionadas con el software [ISO-15504].

[6] Proceso, es un conjunto de actividades interrelacionadas, que transforman entradas a salidas [IEEE-12207].
Nota: El término "actividades" incluye el uso de recursos.

[7] Ingeniería, es la aplicación de un enfoque sistemático, disciplinado y cuantitativo a estructuras, máquinas, productos, sistemas o procesos [IEEE-610.12].

[8] Ciclo de vida del producto, es el periodo de tiempo que comienza con la concepción del producto y termina cuando éste ya no está disponible para usarse.

componentes un proceso realizado mediante trabajo-en-grupo[9]. Los principales sub-procesos realizados en equipo son: especificación de requerimientos, análisis & diseño, implementación y prueba, que iteran continuamente para cada componente de la arquitectura del software, de tal manera que el producto se obtiene incrementalmente [IEEE-610.12]. El método AGD establece que el proceso se lleva acabo en forma: grupal, incremental, cooperativa, adaptable y directa.

Una vez definido el método AGD diseñamos y realizamos un experimento para validar nuestro enfoque. Mediante la experimentación se compararon los resultados obtenidos usando el AGD en relación a los derivados mediante el proceso unificado de Rational (RUP) [RUP03a] [RUP03b], que es una norma de facto en el desarrollo de software. Evaluamos alguna de las hipótesis estadísticas emanadas a partir de la siguiente hipótesis general, establecida para la comprobación:
Productos de software de mejor calidad (con menos defectos) pueden obtenerse mediante un proceso dirigido a desarrollar los componentes de la arquitectura de software, mediante modos de trabajo-en-grupo [GM04].

Los resultados de la experimentación mostraron que existe un efecto positivo significativo. El cliente identificó menos defectos para los productos desarrollados usando AGD, en virtud de que se cubren algunas deficiencias existentes en el desarrollo de software mediante RUP.

Además de la definición del AGD y la experimentación efectuada, se diseñó y construyó una herramienta para registro y monitoreo del proceso: el Sistema de Soporte para el Proceso de Software en equipo (SiSoProS). SiSoProS se caracteriza por: 1) asignación de la máxima importancia a los componentes de la arquitectura de software y los productos-del-trabajo asociados y 2) la subordinación al producto del proceso, estructurado mediante cuatro modos de trabajo en grupo. SiSoProS facilita la captura y presentación cuantitativa del proceso realizado y del producto obtenido.

El contenido del capítulo se estructura de la manera siguiente:

- Describimos la motivación de la investigación realizada (§ 1.1), mostrando la problemática involucrada y la especificación de los requerimientos a satisfacer en el área del producto de software y en el área del proceso de software realizado.
- Introducimos los principales conceptos técnicos del área del producto de software (§ 1.2) y del proceso de software (§ 1.3), utilizados en la solución propuesta.
- El enfoque de solución propuesto muestra la utilización de los conceptos técnicos (§ 1.4), distinguiendo las áreas del producto de software y del proceso.
- La justificación del trabajo realizado y los beneficios técnicos logrados mediante nuestra propuesta se describen en (§ 1.5).

[9] Trabajo-en-grupo, es trabajo realizado por dos o más personas, siguiendo algún protocolo para alcanzar un objetivo común.

- La contribución técnica de la solución propuesta (§ 1.6), en las áreas del producto y proceso de software, se evidencia.
- Terminamos el capítulo describiendo la organización del documento (§ 1.7).

1.1Motivación

Actualmente se requiere software más complejo en todos los sectores, para que soporte el trabajo en equipo en las organizaciones e incluya cada vez más áreas usuarias y mayor funcionalidad. Continuamente la extensión del dominio de las aplicaciones está creciendo, a esta situación se agrega la dificultad de usar la plataforma tecnológica actual, que ofrece mayor número de opciones y servicios.

Debido a la creciente complejidad de las aplicaciones y de la plataforma tecnológica, el desarrollo de los productos de software sobrepasa la capacidad de un individuo, requiriéndose el trabajo de varias personas organizadas en equipo y la simplificación de todos los aspectos (ej. proceso, uso de tecnología). Esta circunstancia adquiere mayor relevancia debido al papel protagónico de los productos de software en las áreas críticas del negocio (ej. áreas donde se generan ingresos).

En este contexto, la comunidad interesada en el desarrollo de software ha utilizado su experiencia y resultados en la especificación de un proceso genérico de desarrollo de software, los elementos que lo componen y las relaciones entre esos elementos. La descripción de este proceso genérico es la Especificación del Metamodelo de la Ingeniería del Proceso de Software / Software Process Engineering Metamodel Specification (SPEMS) [OMG-SPEMS02].

Mediante el metamodelo descrito en SPEMS se puede describir naturalmente el Proceso Unificado de Racional (RUP) [RUP03a] [RUP03b]), pues se nota su influencia en la estructura de SPEMS al ser el estándar de facto para el desarrollo de software. En SPEMS y RUP el proceso y sus características tienen mayor importancia que cualquier otro elemento. La importancia del proceso se hace patente pues la atención se centra en él, tanto en el primer nivel de abstracción formado por fases, como en el segundo nivel de abstracción constituido por disciplinas y ciclos. A los productos les otorga un papel secundario en el nivel de menor abstracción del proceso, razón por la cual los productos tienen una influencia menor en la planificación y control del desarrollo.

La mayoría de los enfoques de desarrollo de software conocidos [B85] [S95] [B99] [W00] [H00] [PF02] [OMG-SPEMS02] [RUP03a] [GMP03], con el propósito de asegurar el logro de sus metas, conceden principal importancia al proceso y otorgan a los productos-del-trabajo (modelos) menor influencia.

A partir de este hecho, distinguimos dos áreas principales en el desarrollo de software:

- *El producto de software***.** Se refiere al conjunto de programas, los procedimientos, los datos y la documentación asociada [IEEE-12207], cuya arquitectura es la estructura (o estructuras), que comprende componentes de software, las propiedades externas visibles de esos componentes y las relaciones entre ellos [BCK98].

  El producto de software, además del código, principalmente se representa mediante el Lenguaje Unificado de Modelado (UML) [CSM03], usando conocimiento acerca de las arquitecturas de software.

- *El proceso de software*. Es el proceso (o procesos), usado por una organización (o proyecto) para planificar, administrar, ejecutar, monitorear, controlar y mejorar las actividades relacionadas con el software [ISO-15504].

La mayoría de los enfoques de desarrollo considera al proceso dentro de los tres principales elementos utilizados para controlar la producción de software [B85] [S95] [B99] [W00] [H00] [PF02] [OMG-SPEMS02] [RUP03a] [GMP03] y jerarquiza los procesos especificando varios niveles de abstracción (ej. RUP). Controlan el desarrollo y organizan el trabajo mediante participantes especializados en áreas técnicas, dando poca atención a la dinámica y requerimientos del desempeño de un equipo de desarrolladores.

Los procesos de desarrollo más empleados [H99] [RUP03a] [OMG-SPEMS02] no consideran al producto dentro de los tres principales elementos utilizados para controlar la producción de software (§ 2.12), situación que nos hace dudar de la calidad de los productos resultantes.

Después de analizar el contexto del desarrollo de software, en las áreas mencionadas del producto de software y del proceso necesario para construirlo, se identificó la problemática y a partir de ésta, se identificaron un conjunto de requerimientos. Los requerimientos que siguen, sirvieron como guía para plantear la solución propuesta en esta investigación.

1.1.1 Problemática acerca del producto de software

Es común que en una organización, el trabajo se realiza mediante grupos coordinados [D88] [F94]. El trabajo coordinado en equipo involucra la interacción de varias disciplinas sociales (Ej. psicología, sociología) y técnicas, conjuntando naturalmente las áreas de investigación de sistemas de información (IS) [MS98] y de trabajo Cooperativo Soportado por Computadora (CSCW) [SB92]. De esta manera, se observa:

Conceptos de grupo. Se requiere incluir los elementos necesarios (ej. memoria de grupo, cooperación), para soportar el trabajo en grupo que se realiza en las organizaciones.

La segunda motivación surge cuando se considera la estructura del producto de software, ya que se carece de una guía que sugiera los componentes necesarios, en una aplicación soportada por computadora, para atender el trabajo en grupos con la plataforma tecnológica actual (ej. Internet, Web,…). No hay ayuda para encontrar partes faltantes, que se descubrirán en el momento de usar la aplicación generada y constatar la funcionalidad incompleta u omisión de alguna estructura de datos requerida.

La estructura del producto de software es importante cuando se consideran las secuencias de modelos necesarias para el desarrollo de una aplicación, pues conviene contar con un patrón que guíe la selección y orden de realización de los modelos.

La necesidad de referencias para la estructura del producto, sugiere dar mayor importancia a la arquitectura de software, estableciendo: 1) un modelo de referencia, es decir, una estructura que soporte el funcionamiento y el flujo de datos para el tipo de aplicación a desarrollar y 2) un estilo arquitectónico, es decir, una descripción de los tipos de componentes y un patrón del control, junto con el flujo de datos. De esta manera, uno de los requerimientos a cubrir es:

Modelos de referencia. Durante la construcción de un producto de software se requieren modelos de referencia para identificar los productos-del-trabajo a desarrollar.

La tercera situación que impulsa esta investigación es el hecho de que el lenguaje UML describe una variedad amplia de productos-del-trabajo, sin embargo, su notación puede ser ambigua, por ejemplo se puede usar para representar la misma vista del sistema, un diagrama de colaboración o un diagrama de actividades. En este sentido es beneficioso determinar un conjunto suficiente[10] de productos, para obtener un componente de software de calidad.

Además de identificar los elementos contenidos en el conjunto de productos, se necesita indicar el modelo del UML a utilizar para cada elemento del conjunto. Incluso es conveniente proponer modelos adicionales a los existentes en UML, pues se reconoce que existen más artefactos o modelos que los descritos en el lenguaje. De esta manera, uno de los requerimientos a cubrir es:

Especificación. En el desarrollo de cada producto-del-trabajo se requiere especificar el elemento del lenguaje UML a utilizar.

El desarrollo de software se realiza a través de transformaciones o procesos, generando un modelo a partir de otro, hasta obtener el producto de software terminado. Cambiando consecutivamente modelos independientes de cómputo (CIM[11]) a modelos independientes de plataforma (PIM), modelos PIM a modelos de plataforma específica (PSM) y modelos PSM a modelos de código.

La transformación de un modelo a otro, requiere definiciones de las transformaciones, que relacionan elementos constructivos de un modelo a elementos de otro modelo. La transformación es un proceso descrito por las definiciones de transformación, mediante reglas realizado por un desarrollador o automáticamente por una herramienta [KWB03], manteniendo las características siguientes:

- Entonable.- las reglas generales, de transformación, puedan ajustarse.
- Seguible.- puede seguirse un elemento en el modelo objetivo, hacia el elemento del modelo fuente, del que proviene.
- Consistencia incremental.- cuando se agrega información al modelo objetivo, dicha información debe persistir para que este modelo se pueda regenerar, a partir del modelo fuente modificado.
- Bidirecionalidad.- una transformación puede aplicarse de fuente a objetivo y también de objetivo a fuente.

De esta manera, uno de los requerimientos lo enunciamos como:

---

[10] Suficiente. Solamente se consideren los productos cuya eliminación provocaría insatisfacción de algún requerimiento.

[11] CIM, PIM y PSM son términos usados en la Arquitectura Dirigida por Modelos / Model-Driven Architecture (MDA) [OMG03A].

Transformación entre modelos. Identificar las transformaciones entre modelos con la finalidad de habilitar su realización automática.

Los autores de los procesos genéricos, como RUP, u otros descritos mediante SPEMS, requieren especialistas en el dominio de la aplicación, condición que frecuentemente no es posible cumplir, esta es nuestra quinta motivación. En los casos en que es difícil o costoso contar con expertos, hace falta tener representaciones genéricas de los procedimientos y de los participantes en áreas diversas de aplicación, en el contexto de una organización. De tal manera que estas representaciones faciliten la identificación de las entidades del negocio caracterizando a las entidades soportadas por computación y a las que no la utilizan, para ayudarse en la selección de los casos de uso y la elaboración de los diagramas UML restantes.

Aunque hay avances para el modelado conceptual en el UML 2.0, por ejemplo en los diagramas de interacción [OMG05] [FGD06], se requieren tipos de diagramas y procedimientos adicionales apropiados para modelar el negocio, a un nivel de abstracción alto (independiente de cómputo) que guíe la identificación de los componentes básicos. De esta manera, uno de los requerimientos a satisfacer es:

Modelado conceptual del negocio. Para producir un modelo de proceso del negocio es necesario tener elementos explícitos en UML, que sean parte del modelo conceptual del componente en desarrollo.

La tabla 1-1 resume los requerimientos descritos anteriormente, en la primera columna se incluye el nombre y en la segunda su descripción.

Tabla 1-1 Requerimientos orientados al producto de software.

Nombre	Descripción
Conceptos de grupo	Se requiere incluir los elementos necesarios (ej. memoria de grupo, cooperación), para soportar el trabajo en grupo que se realiza en las organizaciones.
Modelos de Referencia	Durante la construcción de un producto de software se requieren modelos de referencia para identificar los productos-del-trabajo a desarrollar.
Especificación	En el desarrollo de cada producto-del-trabajo se requiere especificar el elemento del lenguaje UML a utilizar.
Transformación entre modelos	Identificar las transformaciones entre modelos con la finalidad de habilitar su realización automática.
Modelado conceptual del negocio	Para producir un modelo de proceso del negocio es necesario tener elementos explícitos en UML, que sean parte del modelo conceptual del componente en desarrollo.

1.1.2 Problemática acerca del proceso de software

La primera motivación en el área del proceso de software es el incremento de la complejidad del proceso de software, añadiéndose el hecho de que el objetivo principal de la mayoría de las

empresas no es el desarrollo en sí de software, sino alguna otra área de negocio (fabricación de un producto de uso específico ej. engranes). Partiendo de estos hechos y en virtud de que el esfuerzo demandado para utilizar las plataformas tecnológicas sobrepasa la capacidad de un individuo para desarrollar software [S06][MS98], los productores de software están obligados a trabajar en equipo[12], estableciendo compromisos tanto acerca de la dinámica y administración del equipo, como en las asignaciones de desarrollo específicas [PB04].

El proceso de software en equipo TSP [H99], dado que está dirigido a equipos donde intervienen un número menor o igual a 20 participantes, es necesario extenderlo para soportar el desarrollo de productos complejos, en el contexto de una organización. De esta manera se requiere:

Desarrollo en equipo. El desarrollo de software requiere trabajo en equipo disciplinado y coordinado, asignando responsabilidades y tareas a los participantes.

De acuerdo con el análisis del Rational Unified Process (RUP), identificamos procesos descrito en dos niveles de abstracción, con la finalidad de controlar el desarrollo: el primero incluye una secuencia de fases: Inicio / Inception, Elaboración / Elaboration, Construcción / Construction y Transición / Transition; y el segundo un conjunto de disciplinas nucleo: Modelado del negocio / Business Modeling, Requerimientos / Requirements, Análisis & Diseño / Analysis & Design, Implementación / Implementation, Prueba / Test y Distribución / Deployment

La existencia de los dos niveles confunde a los desarrolladores, pues alguna disciplina (del segundo nivel, ej. Implementación) tienen significado similar al de alguna fase (del primer nivel, ej. Elaboración). El significado semejante (ej. Elaboración Vs. implementación) de elementos ubicados en diferentes niveles de abstracción complicando el entendimiento y la realización del proceso [H03].

Adicionalmente a las disciplinas núcleo, RUP define en el segundo nivel, disciplinas de soporte complementarias: Administración de Configuración & Cambio / Configuration & Change Management, Administración de Proyecto / Project Management y Ambiente / Environment. Situación que aumenta aún más la complejidad del proceso.

Para valorar la conveniencia del uso de fases en el primer nivel (principal del proceso) se considera que las fases son conceptos del marco de ciclo de vida en cascada y que no son tan importantes cuando los sistemas de software no se desarrollan en forma monolítica [DBC88] [G89] [C97].

---

[12] Trabajo en equipo – un modo de trabajo-en-grupo, en cuyo protocolo se establecen las reglas y roles requeridos para asegurar que se logran las metas definidas, y se cubren las áreas administrativas y de soporte indispensables para asegurar la calidad.

En el desarrollo no monolítico se realizan varios sub-productos que evolucionan de manera asíncrona e independiente. Esta situación nos sugiere la necesidad de un proceso simple que se pueda usar para desarrollar cualquier producto. De esta manera, se requiere:

Simplificación. La complejidad del proceso se reduce evitando denominaciones semejantes a niveles diferentes (Ej. nombres de fases Vs. disciplinas), mejorando la interpretación y realización incremental del proceso.

Los procesos actuales, como aquellos ajustados al modelo SPEMS, son secuenciales y rígidos, incluso los procesos descritos mediante dos niveles, ambos tiene esas características. Para lograr que el método de desarrollo sea ágil debe evitar la secuencialidad (cascada de fases) [A01], en lo global, así como en las iteraciones de mayor detalle (mini-cascada de disciplinas).

Conviene dirigir el proceso hacia la obtención de productos desarrollados independientemente, organizados en un conjunto ordenado que sugiera la serie de transformaciones requeridas. En cuanto al proceso de obtención de los productos, se requiere identificar un sub-proceso en el que se defina la estrategia de desarrollo del producto-del-trabajo en turno, mediante la realización sistemática de un conjunto de sub-procesos. De tal manera que la estrategia indique el orden más adecuado de realización, en función del contexto de desarrollo. De esta manera se necesita:

Flexibilidad. Los procesos no deben ser secuenciales ni rígidos, de tal manera que el proceso pueda modificarse, incluso aún cuando se realicen transformaciones automáticas entre modelos.

Los productores de software precisan de un proceso dinámico que entregue rápidamente resultados, con la finalidad de participar en mercados altamente competidos. Consideramos necesario introducir, en el proceso de desarrollo, los factores de calidad requeridos por los métodos ágiles [AWSR03]: a) *incrementa*l - entregas pequeñas de software, con ciclos rápidos de desarrollo; b) *cooperativo* - interacción estrecha entre el cliente y el desarrollador; c) *directo* – proceso suficientemente documentado, fácil de aprender y modificar; d) *adaptable* - facilidad para modificar el producto y reaccionar adecuadamente a cambios de último momento. De esta manera se requiere:

Agilidad. El proceso debe permitir la adecuación dinámica y realización en forma ágil: incremental, cooperativo, directo y adaptable, es decir adecuar productos, sub-procesos y formas.

La tabla 1-2 resume el análisis anteriormente descrito del área de proceso.

Tabla 1-2 Requerimientos orientados al proceso de software.

Nombre	Descripción
Desarrollo en equipo	El desarrollo de software requiere trabajo en equipo disciplinado y coordinado, asignando responsabilidades y tareas a los participantes.
Simplificación	La complejidad del proceso se reduce evitando denominaciones semejantes a niveles diferentes (Ej. nombres de fases Vs. disciplinas), mejorando la interpretación y realización incremental del proceso.
Flexibilidad	Los procesos no deben ser secuenciales ni rígidos, de tal manera que el proceso pueda modificarse, incluso aún cuando se realicen transformaciones automáticas entre modelos.
Agilidad	El proceso debe permitir la adecuación dinámica y realización en forma ágil: incremental, cooperativo, directo y adaptable, es decir adecuar productos, sub-procesos y formas.

1.2Conceptos técnicos del producto de software

El producto de software y todos los productos-del-trabajo se especifican explícitamente para guiar el proceso involucrado en el desarrollo. Para lograr este objetivo se consideran la Arquitectura Dirigida por Modelos (MDA), la Integración del Modelo de Madurez de la Capacidad (CMMI) y los elementos necesarios para soportar el trabajo cooperativo en la organización.

1.2.1 Arquitectura Dirigida por Modelos

La arquitectura dirigida por modelos / Model Driven Architecture (MDA) [OMG03A] es un marco para el desarrollo de software definido por el grupo de administración de objetos / Object Management Group (OMG). Un concepto de suma importancia para MDA es el papel preponderante de los modelos en el desarrollo del software. Con MDA el desarrollo esta dirigido mediante el modelado del sistema de software.

El proceso se parece al realizado normalmente para desarrollar, la mayor diferencia radica en la naturaleza de los productos-del-trabajo creados durante el desarrollo. Los productos son modelos formales susceptibles de entenderse por la computadora, siendo los principales: el Modelo Independiente de Plataforma (PIM), el Modelo de Plataforma Específica (PSM) y el código.

Modelo Independiente de Plataforma

El Modelo Independiente de Plataforma (PIM) tiene un grado de abstracción alto y es neutral a la tecnología de implementación. PIM describe a un sistema de software que soporta algún negocio, desde el punto de vista de qué tan bien lo sustenta. No se consideran características de la tecnología en la que se desarrollará, como el tipo de base de datos a utilizar.

Modelo de Plataforma Específica

El modelo siguiente se obtiene transformando el modelo PIM en uno o más Modelos de Plataforma Específica (PSM)s. PSM se confecciona para especificar el sistema en términos de elementos constructivos de implementación disponibles en la tecnología específica a utilizar. Para cada tecnología específica se genera un PSM por separado.

Código

El paso final del desarrollo dirigido por modelos es la transformación de cada PSM a código. Esta transformación es casi directa porque PSM se ajusta mucho a la tecnología específica.

Primordialmente MDA define el modelo PIM, el modelo PSM y el código, así como la forma en que se relacionan entre sí.

1.2.2 Integración de modelos de madurez de la capacidad

La integración de modelos de madurez de la capacidad / Capability Maturity Model Intergration (CMMI) [SEI-CMMI-S] [SEI-CMMI-C] contiene los elementos esenciales de los procesos eficaces para varios cuerpos de conocimiento (ingeniería de sistemas, ingeniería de software y desarrollo integrado de producto y proceso). Estos elementos se fundamentan en conceptos desarrollados por Crosby, Deming, Juran y Humphrey [C79] [D86] [J88] [H89], para guiar en la mejora de los procesos de la organización y la habilidad de gestionar el desarrollo y mantenimiento de productos y servicios.

Uno de los componentes principales de la CMMI, es el "área de proceso", que es un grupo de prácticas relacionadas a un área (ej. desarrollo de requerimientos, administración de configuración, …), que cuando se realizan colectivamente, satisfacen un conjunto de metas importantes para lograr mejoras significativas en dicha área.

Las 25 áreas de proceso del CMMI se pueden agrupar en cuatro categorías:

- *Administración de Proceso*. Contiene ocho áreas de proceso relacionadas con el proyecto y su planificación, monitoreo, control, manejo de proveedores, manejo de riesgos y equipamiento.
- *Administración de Proyecto*. Abarca cinco áreas de proceso enfocadas al proceso organizacional, así como su definición, innovación, capacitación, difusión y desempeño.
- *Ingeniería*. Incluye seis áreas de proceso encaminadas al manejo de requerimientos, desarrollo de la solución técnica, así como a la integración y validación del producto.
- *Soporte*. Con seis áreas de proceso ocupadas de la administración de configuración, aseguramiento de calidad, medición y análisis de las mediciones, así como el manejo de decisiones y causas.

Aunque se hayan agrupado las áreas de proceso en la forma mencionada, con frecuencia interactúan y tienen efecto unas sobre las otras sin importar los grupos a que pertenecen. Las categorías de áreas de proceso de CMMI agrupan los demás elementos del modelo: metas específicas, prácticas específicas, metas genéricas, prácticas genéricas, productos-del-trabajo típicos, sub-prácticas, notas, amplificaciones de disciplina, elaboración de prácticas genéricas y referencias.

Los modelos CMMI no son procesos o descripciones de procesos. Los procesos seguidos en la organización dependen de muchos factores, incluyendo el dominio de aplicación, la estructura y el tamaño de la organización. En particular las áreas de proceso del modelo CMMI no se mapean uno a uno con los procesos usados en la organización.

El propósito del CMMI es proveer guía para mejorar los procesos de la organización y la habilidad de administrar el desarrollo, adquisición y mantenimiento de productos y servicios. CMMI ubica enfoques probados en una estructura que ayuda a la organización a evaluar la

madurez organizacional o capacidad del área de proceso, estableciendo prioridades para acrecentar el desempeño y realizar los cambios.

1.3Conceptos técnicos del proceso de software

Se requiere supeditar el proceso al producto de software y su arquitectura, así como a los productos-del-trabajo necesarios para su construcción, reconociendo además que uno o varios grupos de personas con habilidades técnicas y administrativas lo llevan a cabo. Para lograr esto consideramos la realización del trabajo en grupo, el proceso disciplinado y el proceso ágil.

1.3.1 Trabajo en grupo

Existen varios tipos de trabajo en grupo que tienen protocolos definidos:

- Trabajo en equipo para desarrollo de software, mediante varios participantes que representan papeles diferentes de acuerdo a las necesidades de la dinámica del equipo. Descrito en la técnica proceso de software en equipo / Team Software Process (TSP) [H99].
- Trabajo para la realización de modelos o productos-del-trabajo en forma colaborativa, por dos personas. Definido como proceso de software colaborativo / the Colaborative Software Process (CSP) [W00]. De acuerdo a los conceptos del proceso personal de software / Personal Software Proces (PSP) [H95].
- Trabajo para la revisión técnica de productos-del-trabajo. En el que se cuenta con protocolos para solución de conflictos de intereses, como los del estándar IEEE Std 1028-1997: IEEE Standard for Software Reviews [IEEE-1028].
- En el estándar IEEE 1028 se definen los procedimientos para: revisiones administrativas, técnicas, inspecciones, paso-a-paso y auditorias.
- Trabajo para monitoreo y control de avances de proyectos. Descrito en el estándar IEEE 1028 como revisiones administrativas y en la técnica TSP como reunión de seguimiento.

1.3.2 Proceso disciplinado

La ingeniería de software es una disciplina joven que busca fundamentar sus procedimientos en principios de ingeniería probados. La comunidad de ingeniería de software inspirada en Deming [D86] y Juran [J88] ha determinado que se requiere una buena capacidad de desarrollo del proceso para obtener un producto de alta calidad. Estándares de proceso como el ISO 9000 y la integración de modelos de madurez de la capacidad (CMMI) se desarrollaron para alcanzar resultados más previsibles mediante guías para incorporar procedimientos probados en los procesos. Las compañías que han adoptado los estándares ISO 9000 [ISO-90003] y CMMI [SEI-CMMI-S] [SEI-CMMI-C] han logrado mejoras sustanciales.

El proceso de adopción por las empresas, de los estándares mencionados, inspiró la aparición de varias técnicas que refinan a las normas en la última década del siglo XX y en los primeros años del siglo XXI. Primero surgió la técnica Proceso Personal de Software / Personal Software Process (PSP) [H95] que proporciona los elementos necesarios para el trabajo disciplinado de un desarrollador individual; en seguida apareció la técnica Proceso de Software en

Equipo / Team Software Process (TSP) [H99] que agrega los elementos necesarios para planificar y controlar el trabajo de un equipo de desarrolladores y después surgió la técnica Proceso de Software Colaborativo (CSP) [W00] que utiliza los conceptos del PSP para el trabajo de una pareja de desarrolladores (la programación por pareja es una práctica sugerida en los métodos ágiles).

1.3.3 Proceso ágil

Los métodos ágiles son una reacción a las formas tradicionales de desarrollar software y el reconocimiento de la necesidad de una alternativa a los procesos pesados de desarrollo de software, guiados por documentación. En la implementación mediante los métodos tradicionales, el trabajo comienza con la recopilación y documentación de un conjunto completo de requerimientos, seguida por el diseño de alto nivel, desarrollo e inspección.

El proceso ágil usa la documentación, pero no cientos de páginas que nunca se mantienen y rara vez se usan. Se planifica, pero se reconocen los límites de la planificación. Las prácticas adoptadas en los métodos ágiles se resumen en el manifiesto ágil [B01] que establece:

"Se promueven mejores formas de desarrollo de software mediante práctica, ayudando a que otros las realicen. En las prácticas promovidas se valoran:

- Más a los individuos y su interacción, que al proceso y las herramientas
- Más la obtención de software corriendo en una plataforma, que completar la documentación
- Más la colaboración del cliente, que la negociación del contrato
- Más la respuesta al cambio, que seguir un plan

Damos mayor relevancia a los primeros elementos, pero reconocemos el valor de los elementos enunciados después."

1.4 Enfoque de solución y justificación

La solución propuesta se fundamenta en dirigir el proceso de desarrollo de software en función del producto a obtener. Para lograrlo definimos las características de un producto de software con la complejidad de las aplicaciones actuales y la plataforma tecnológica existente, especificando también el proceso adaptable para el desarrollo del producto de software.

Formulamos el método Arquitectónico en Grupo / Architectural and Group Development (AGD), que incluye un ciclo de vida del producto asociando a cada producto-del-trabajo un proceso que integra varios modos de trabajo-en-grupo, donde el modo en equipo agrupa los sub-procesos a realizar. El método AGD establece la realización del proceso de manera grupal, incremental, cooperativa, adaptable y directa.

Los elementos del método AGD (en negrillas) y los requerimientos que satisfacen se listan a continuación:

1. **Conjunto de Productos (PS: Product Set)**. El PS proporciona la estructura y características de los productos-del-trabajo necesarios para construir el producto de software objetivo. Usa como referencia un modelo de componentes de software que soportan el trabajo

cooperativo y un estilo arquitectónico dirigido a modelos, que posibilita la transformación entre modelos estándar.

PS resuelve los requerimientos: Conceptos de grupo, Modelos de referencia, Especificación, Transformación entre modelos y Modelado conceptual del negocio.

2. **Proceso Dirigido por modelos (MDP: Model Driven Process)**. MDP establece la estructura y características del proceso de desarrollo requerido para generar el PS específico del producto de software objetivo. El proceso integra explícitamente los modos de trabajo-en-grupo: equipo - para establecer la infraestructura administrativa requerida, colaborativo – para desarrollar los productos-del-trabajo, cooperativo – para revisar técnicamente los productos-del trabajo y control - para monitorear, prevenir y corregir el desempeño.

   El trabajo se realiza por equipos de desarrolladores que iteran continuamente los subprocesos: lanzamiento & estrategia, planificación, requerimientos, análisis & diseño, implementación, prueba y pos-término, para cada componente de la arquitectura del software, de tal manera que el producto se obtiene incrementalmente.

   MDP satisface los requerimientos: Desarrollo en equipo, Simplificación, Flexibilidad y Agilidad.

   El método AGD otorga máxima importancia al producto de software, para obtenerlo mediante un proceso realizado por grupos de participantes, que dirigen su esfuerzo a desarrollar los componentes de la arquitectura del software objetivo.

AGD proporciona mayor importancia a la arquitectura de software, en el proceso de desarrollo, mediante el PS y el MDP. Para la estructura del PS establecimos, como el nivel de mayor abstracción, los conjuntos de productos de las cuatro categorías de áreas de proceso del CMMI. El beneficio que obtenemos considerando al CMMI es asegurarnos de tener un conjunto completo de productos del trabajo. Los productos de la categoría Ingeniería se ubica como el elemento principal para el control del desarrollo.

Definimos el conjunto de productos Ingeniería utilizando los cuatro modelos de MDA, especificando como primer nivel de abstracción el conjunto ordenado siguiente: modelo independiente de cómputo (CIM: Computation Independent Model), modelo independiente de plataforma (PIM: Plataform Independent Model), modelo de plataforma específica (PSM: Platform Specific Model), modelo de implementación (IM: Implementation Model) y modelo de operación (OM: Operation Model). Ajustándonos a MDA, buscamos contar con las ventajas potenciales de la ingeniería dirigida por modelos, que nos permitirán realizar trabajos futuros orientados a desarrollar mediante transformaciones de modelos.

El proceso está en función del conjunto de productos de la categoría Ingeniería, ubicando el proceso MDP en un nivel secundario, donde se utilizan cuatro modos de trabajo en grupo que se definieron considerando los protocolos existentes de trabajo en grupo (revisiones [IEEE-1028], proceso personal [H95], proceso en equipo [H99], trabajo colaborativo [W00]). Apoyándonos en las ventajas reportadas por sus grupos de usuarios (ej. publicaciones del Instituto de Ingeniería de Software, www.sei.cmu.edu/publications).

El modo de trabajo en equipo se estructuró, para obtener los componentes de la arquitectura del software y los productos-del-trabajo asociados, siguiendo la técnica del proceso de software en equipo (TSP: Team Software Process) [H99]. De tal manera que un grupo con asigna-

ción de roles explícita itere el proceso requerido para obtener los productos solicitados, abarcando la mayoría de las áreas de proceso de CMMI.

Simultáneamente a la dinámica del modo de trabajo en equipo, se realiza:

- en forma colaborativa, el desarrollo de los productos-del-trabajo;
- mediante el modo de control, el reporte y gobierno del desempeño, usando las características importantes del producto y de los recursos utilizados en el desarrollo;
- de manera cooperativa, la revisión de los productos de ingeniería.

Buscamos la sinergia de los cuatro modos de trabajo en grupo, facilitando su comprensión al describirse en forma sencilla por separado, pero comprobando su realización concurrente mediante la ocurrencia de eventos específicos.

AGD es incremental, cooperativo, adaptable y directo, mediante los conceptos y dinámica de los procesos iterativos incrementales que abarcan a los métodos ágiles de desarrollo de software. La utilización de AGD permite la adecuación del PS y del MDP en función de las características del producto de software y de la organización que desarrollará el producto.

La importancia principal de los modelos en el método AGD permitirá contar con la opción de automatizar, siempre que sea posible, la transformación de un modelo al siguiente en el conjunto de productos (PS). Cuando se cuenta con el soporte automatizado o semi-automatizada, las transformaciones de modelos [CH03] [TEL05] sustituirían a los procesos requeridos.

AGD aporta una simplificación para obtener cualquier producto-del-trabajo mediante un nivel único de sub-procesos, adaptable a las circunstancias. Este proceso permitirá además desarrollar el software mediante transformaciones de modelos, conforme vayamos avanzando en esta dirección.

El producto de software se construye definiendo un Conjunto de Productos (PS: Product Set), elemento principal para controlar el desarrollo de software, tomando como referencia al CMMI [SEI-CMMI-S] [SEI-CMMI-C], la MDA [OMG03A] y el modelo Trabajo Cooperativo - Almacenamiento, Lógica y Recursos (CWSLR) [GJM99].

En el procedimiento realizado para obtener el PS se requería un modelo de referencia[13], razón por la cual se sintetizó el modelo para componentes de software que soportan la cooperación: Trabajo Cooperativo - Almacenamiento, Lógica y Recursos / Cooperative Work: Storage, Logic and Resources (CWSLR) [GJM99].

CWSLR proporciona sustento al desarrollo de software, constituye un modelo de referencia para el tipo de aplicaciones de ambiente cooperativo en Internet, sirviendo de base de la es-

[13] Modelo de referencia es una estructura que soporte el funcionamiento y el flujo de datos para el tipo de aplicación a desarrollar.

tructura de productos (Capítulo 3), la organización de los procesos (Capítulo 4) y la interfase con el usuario (Capítulo 6).

PS proporciona la información necesaria para sugerir: a) un orden de construcción, b) los modelos a utilizar y c) representaciones apropiadas para el negocio. Esta información posibilita la realización de transformaciones entre modelos, facilitando la organización del desarrollo dirigiéndolo a subproductos que evolucionan de manera asíncrona e independiente.

Otra referencia usada para definir el PS fue la MDA, que aporta a la arquitectura de software un estilo que se coloca en línea con elementos aceptados por OMG: Object Management Group, como son: CORBA[14] [OMG02A] y XMI[15]; proporcionando la estructura estándar que permitirá sustituir procesos mediante transformaciones de un modelo a otro.

PS aporta una guía para identificar cuando y como utilizar cada modelo del lenguaje unificado de modelado (UML: Unified Modeling Language) y utilizar como núcleo base al UML nos beneficia ya que utilizamos elementos estandarizados.

En el área de procesos el método AGD propone el Proceso Dirigido por Modelos (MDP: Model Driven Process), realizado mediante la sinergia de cuatro modos de trabajo-en-grupo: Equipo, Colaborativo, Cooperativo y de Control. MDP guía la elaboración de un producto de software de mayor calidad mediante patrones de proceso que Incluyen los mecanismos necesarios para su realización flexible y ágil.

Mediante un proceso de un sólo nivel de abstracción se disminuye la complejidad como resultado de eliminar uno de los dos niveles (fases y disciplinas) que consideran la mayoría de los procesos de software actuales. En el caso del RUP, eliminaríamos sus cuatro fases y los inconvenientes de ambigüedad que se presentan entre el significado de las fases y las disciplinas.

Todo modelo o producto-del-trabajo se desarrolla mediante la dinámica del modo de proceso en equipo, usando ciclos de siete sub-procesos. Reconociendo los ciclos, como un proceso de ingeniería de sistemas, que se aplicará sistemáticamente para resolver el problema planteado por la realización de cualquier modelo del PS. La ejecución de los sub-procesos es flexible, siendo la única restricción, efectuar siempre el primer y el último sub-procesos, permitiendo que los cinco sub-procesos restantes se realicen en cualquier orden o incluso omitir alguno en caso de justificarlo.

---

[14] CORBA – (Common Objet Request Broker Architecture), arquitectura e infraestructura, independiente-de-proveedor, que las aplicaciones computacionales usan, para trabajar juntas sobre las redes.

[15] XMI – (XML Metadata Intergchange), modelo de intercambio de información abierta, que proporciona a los desarrolladores que trabajan con tecnología de objetos, la habilidad de intercambiar datos de programación sobre el Internet de manera normalizada.

1.5Contribución

Productos de software con menos defectos se pueden elaborar usando el método AGD, que se dirige a generar la arquitectura del software objetivo, fundamentada en el modelo de referencia para componentes CWSLR y el marco para desarrollo MDA, mediante el trabajo coordinado de grupos de participantes.

La disminución de defectos se propicia por:

1. El fundamento de la arquitectura de software, que:
   - evita la omisión, en la arquitectura, de algún elemento especificado en los modelos de referencia y
   - proporciona un conjunto ordenado de productos-del-trabajo que guían el proceso de desarrollo, ya sea realizado manualmente o usando transformaciones entre modelos.
2. Los modos de trabajo-en-grupo, que consideran explícitamente la realización coordinada de cuatro modos de trabajo-en-grupo, que inducen: administración del trabajo en equipo, realización del trabajo colaboratívamente, revisión de los productos y control del desempeño de los recursos disponibles.

1.6Organización del Documento

Primero describimos y analizamos los trabajos relacionados especificando su conexión con el AGD, así como sus similitudes y diferencias (capítulo 2).

En seguida explicamos el método propuesto como solución: Método de Desarrollo Arquitectónico en Grupo (AGD), y su componente principal, el Conjunto de Productos (PS) (capítulo 3). PS se ajustó a los modelos:

- "Trabajo Cooperativo: Almacenaje, Lógica y Recursos / Cooperative Work: Storage, Logic and Resources" (CWSLR),
- "Arquitectura Dirigida por Modelos / Model Driven Architecture" (MDA) y
- "Integración del Modelo de Madurez de la Capacidad / Capability Maturity Model Integration" (CMMI);

proporcionando una cadena definida de modelos a utilizar, para el desarrollo de un componente de software, incluyendo los lineamientos para el estilo de arquitectura de software a utilizar.

El segundo componente importante del método AGD, es el "Proceso Dirigido por Modelos / Model Driven Process" (MDP) (capítulo 4). MDP depende del PS y reconoce en general que el desarrollo de software se realiza mediante grupos de productores, abstrayendo cuatro modos de trabajo-en-grupo: equipo, colaborativo, cooperativo y de control.

Describimos después los resultados de la experimentación realizada usando el proceso RUP y el método AGD (capítulo 5). Mediante esta experiencia observamos el comportamiento de varias mediciones realizadas por equipos durante el desarrollo de software, utilizando ambos enfoques.

El prototipo de la herramienta "Sistema para el Soporte del Proceso de Software en Equipo" (SiSoProS) (capítulo 6), muestra la utilización del AGD.

Las conclusiones (capítulo 7), analizan las contribuciones y los trabajos a realizar.

Incluimos la información complementaria para la comprensión de los resultados de esta investigación, en dos anexos:

- Anexo A: los patrones de proceso, que describen a tres modos de trabajo-en-grupo: trabajo Colaborativo, trabajo Cooperativo y trabajo de Control.
- Anexo B: el modelo para componentes de software CWSLR, que es una referencia para estructurar un componente o aplicación que soporta el trabajo cooperativo, en el contexto tecnológico actual de Internet. Proporciona una base para estructurar varios elementos del AGD. CWSLR es uno de los elementos que dirigen al método AGD hacia la arquitectura del software.

2 Análisis de Trabajos Relacionados

2.1Introducción

Después de revisar las contribuciones en las áreas del desarrollo de software incremental, iterativo y ágil, se formuló el método de Desarrollo Arquitectónico en Grupo (AGD), con la finalidad de mejorar el desempeño del proceso, tomando como referencia los conceptos de los trabajos relacionados descritos a continuación:

Trabajo cooperativo soportado por computadora (CSCW) [G94], utilizado para determinar los elementos requeridos en el PS para soportarlo y especificación del modo de trabajo cooperativo para la toma de decisiones durante las revisiones técnicas, en presencia de situaciones conflictivas;

Ingeniería Dirigida por Modelos (MDE) [S06], que seguimos estableciendo un conjunto de productos (PS) susceptible de obtenerse mediante transformaciones de un modelo a otro, habilitando la automatización del proceso correspondiente;

Integración del Modelo de Madurez de Capacidad, (CMMI) [C04] del que seguimos su estructura de áreas de proceso para organizar completar el conjunto de productos-del-trabajo (PS);

Desarrollo de Software Orientado-a-Objetos (EOS) [H97], de este tomamos su desarrollo mediante fractales de proceso asociados a los componentes del software, estableciendo análogamente en AGD la relación del conjunto de sub-procesos del trabajo en equipo con cada componente del PS;

Proceso de Software en Equipo (TSP) [H99] y Personal (PSP) [H95], adecuando sus conceptos para estructurar el trabajo en equipo mediante sub-procesos y recursos orientados al control de la dinámica del equipo de desarrollo;

Proceso de Software Colaborativo (CSP) [W00], aprovechamos de él los avances en la integración del trabajo en parejas con un proceso disciplinado. En AGD al menos dos participantes colaboran;

Desarrollo iterativo e incremental [LB03] (ej. RUP [RUP03a], CCPDS-R [R98]), se usa como fundamento de la dinámica del método AGD, para asociar el PS con el proceso dirigido por modelos (MDP);

Algunas técnicas del Manifiesto ágil [HC01], para establecer un proceso de desarrollo: incremental, cooperativo, directo y adaptable;

Modelo de proceso para la industria del software (MOPROSOFT) [O03], del que tomamos su patrón de proceso y

Metamodelo para ingeniería del proceso de software (SPEMS) [OMG-SPEMS02], usado como referencia de algunos conceptos y para contrastar su organización con nuestra propuesta, pues resume las características del proceso de software utilizado en la industria.

En las secciones siguientes, cuando se comenta cada trabajo incluimos el número de renglón que le corresponde en la tabla 2-1, donde resumimos los elementos utilizados para monitorear y corregir desviaciones en el desempeño deseado.

2.2Trabajo Cooperativo Soportado por Computadora

En el método AGD tomamos en consideración los requerimientos del trabajo cooperativo tanto para fundamentar al conjunto de productos, como para definir el proceso a seguir en la obtención de los productos. Este método combina herramientas de dos áreas de investigación (de acuerdo con la taxonomía de Jonathan Grudin [G94]) resultando el área que denominamos sistemas de información que soportan trabajo cooperativo / Cooperative Work Information System (CW-IS) [GJM99] [B02] [K96]. Las áreas de conocimiento fusionadas son el Trabajo Cooperativo Soportado por Computadora (CSCW: Computer Supported Cooperative Work), y los Sistemas de Información (IS: Information Systems).

CSCW es un dominio interdisciplinario del campo de la ingeniería y del campo de las ciencias sociales. Desde la perspectiva de la ingeniería se explica, a continuación, la interpretación de Soportado por Computadora (CS).

El soporte mediante computadora se organiza en dos niveles de abstracción: a) el nivel de contenido, con enfoque en la creación y manejo de la estructura de la información compartida – por ejemplo las bases de datos y bases de conocimiento que se caracterizan por una orientación fuerte en el contenido y b) el nivel de procesos, con enfoque en el proceso de producción de la información – por ejemplo los servicios de mensajería y de video conferencias.

Para organizar mejor las actividades del trabajo cooperativo (CW), se requiere integrar conocimiento de sociología, psicología, antropología, telecomunicaciones, teoría organizacional y administración de la información. La interpretación del Trabajo Cooperativo, en el término CSCW, se explica desde la perspectiva de las ciencias sociales mediante varios aspectos: a) un proceso de trabajo donde se involucran varias personas, b) participantes realizando actividades dirigidas hacia un objetivo predefinido y decidido de común acuerdo, c) trabajo organizado sin estructuras jerárquicas y con un alto grado de autonomía de los individuos (incluyendo el tratamiento de intereses antagónicos y la escasez de información necesaria para la toma de decisiones).

En la concepción del AGD consideramos el punto de vista de la teoría de la cooperación, donde el problema básico es el compromiso entre lo que es bueno para el actor individual a corto plazo y lo que es bueno para el grupo a largo plazo [A00], con tres preguntas teóricas centrales:

1. ¿En que condiciones puede emerger y mantenerse la cooperación entre actores egoístas?
2. ¿Qué consejo se puede ofrecer a un participante en una situación dada acerca de la mejor estrategia?
3. ¿Qué consejo se puede ofrecer a un reformador que quiere modificar los términos mismos de la interacción para promover la cooperación?

La interpretación del CW impone factores sociales de los miembros del grupo, por ejemplo: los miembros requieren un cierto nivel de privacidad y confidencialidad acerca de su producción. En cierto momento un miembro tiene interés de comunicarse con otros, mientras en otro momento quiere concentrarse en su producción y no quiere que lo distraigan (disponibilidad). Se considera la dimensión tiempo para la producción y se regula la asignación de roles (ej. productor, revisor, facilitador, etc.).

Para integrar en forma balanceada las características técnicas de ingeniería (interpretación de CS) y los factores sociales del grupo (interpretación de CW), en el contexto de una organización multidisciplinaria y multiregional, CSCW se define [SB92] como:

> "Disciplina encaminada a comprender la naturaleza y requerimientos del trabajo cooperativo con intención de diseñar tecnologías computacionales orientadas a facilitar los preparativos del trabajo cooperativo".

Los Sistemas de Información (IS), incluidos en el término CW-IS, son un campo de investigación que inicia en 1960, cuyo objetivo principal es proporcionar soporte a la organización incluyendo: la selección de personal, división del trabajo, monitoreo, control y productividad, abarcando dentro de su alcance a las computadoras y a la planificación del trabajo.

Barbara McNurling y Ralph Sprague en 1998 [MS98], identifican como misión de los IS: "mejorar el desempeño de los participantes en una organización a través del uso de la tecnología de información". Es notorio que en la actualidad el trabajo en una organización se realiza mediante grupos coordinados que efectúan tareas cooperativamente [D88] [F94].

AGD para dar soporte a la definición del conjunto de productos (PS) genera y utiliza al modelo CWSLR[16] [GJM99]. Este modelo proporciona, en el contexto de AGD, una referencia para el dominio de los sistemas de información con la finalidad de soportar trabajo cooperativo, representando la división de la funcionalidad y el flujo de datos de los elementos que lo constituyen [GJM00]. Además durante el desarrollo de productos de software mediante AGD se incorpora el conjunto de elementos y las características necesarias para sustentar la cooperación.

2.3Ingeniería Dirigida por Modelos

AGD es un método ubicado en lo general dentro del ámbito de la Ingeniería Dirigida-por-modelos / Model-Driven Engineering (MDE) (1 en tabla 2-1). MDE [S06] [BGK06] es un enfoque para tratar la complejidad de las plataformas y la inhabilidad de los lenguajes de tercera generación en cuanto a aminorar dicha complejidad y expresar eficazmente conceptos del dominio de la aplicación.

MDE combina las tecnologías siguientes:

- Lenguajes de modelado de dominio-específico / Domain-Specific Modeling Languages (DSML) [CSN05]. Cuyo sistema tipo formaliza la estructura de la aplicación, el comportamiento y los requerimientos dentro de un dominio particular, tal como cómputo en mi-

[16] CWSLR - modelo Trabajo Cooperativo: Almacenamiento, Lógica y Recursos / Cooperative Work: Storage Logic and Resources model [GJM99]

siones de aviación [BCG95], servicios financieros en línea [CGM99] o plataformas "middleware" [BCP01]. DSMLs se describen mediante metamodelos, que definen las relaciones entre conceptos en el dominio y especifican precisamente la semántica y restricciones asociadas a los conceptos del dominio.

- Máquinas y generadoras de transformaciones que analizan ciertos aspectos de los modelos y sintetizan varios tipos de artefactos, tales como código, entradas para simulación, descripciones para distribución en Lenguaje de Marcado eXtensible / Extensible Markup Language (XML), u otros modelos. La síntesis de artefactos a partir de los modelos asegura la consistencia entre la implementación y la información contenida en los modelos, de la funcionalidad y los requerimientos de calidad.

Los generadores de herramientas MDE no requieren ser muy complicados ya que sintetizan artefactos que se mapean a interfaces de aplicación (APIs) de plataformas estándar middleware de alto nivel y marcos, en lugar de mapearse a APIs de sistemas operativos de bajo nivel.

La Arquitectura Dirigida por Modelos / Model-Driven Architecture (MDA) es una variante del desarrollo dirigidos por modelos, del Grupo de Administración de Objetos / Object Management Group (OMG). MDA usa el Lenguaje Unificado de Modelado / Unified Modeling Language (UML) y sus perfiles para transformar los modelos a artefactos que se ejecutan en varias plataformas (ej. EJB, .NET y CCM).

En MDE se combinan los DSMLs con las máquinas y generadores de transformaciones; mientras que en AGD tomamos como referencia a MDA y proporcionamos modelos apropiados para recabar información del dominio y organizar los componentes resultantes con ayuda de un catálogo de componentes mínimos (MCTC), que se pueden clasificar por tipo de aplicación. En cuanto a las transformaciones entre modelos se definió una secuencia de productos que se utiliza como referencia, junto con los diagramas del lenguaje UML, para caracterizar e identificar las transformaciones posibles. AGD es un método dirigido a modelos que ayuda a recopilar las entidades y relaciones del dominio de aplicación mediante diagramas y permite el desarrollo del software realizando procesos o usando transformaciones entre modelos.

2.4Integración del Modelo de Madurez de Capacidad

Para ubicar al AGD en el contexto de la mejora de la capacidad de desarrollo de software, organizamos al conjunto definido de productos del trabajo (PS) de tal manera que coincida con las áreas de proceso de la "Integración del Modelo de Madurez de Capacidad / Capability Maturity Model Integration (CMMI)" (2 en tabla 2-1) [SEI-CMMI-C] [SEI-CMMI-S] [C04]. CMMI es un conjunto compuesto de modelos, para varias disciplinas: Ingeniería de Sistemas, Ingeniería de Software, Desarrollo de Producto y Proceso y Fuente de Proveedores, para dirigir en una organización los esfuerzos de mejora de la capacidad a través de las disciplinas mencionadas. CMMI es un marco que acomoda a varias disciplinas y es flexible para soportar dos tipos de representación diferente: por etapas y continua.

La representación por ***etapas*** (utilizada originalmente por CMM [PC91]), tiene las funciones siguientes: 1) Proveer una secuencia probada de mejoras, que comienza con prácticas administrativas básicas y progresa a través de una trayectoria predefinida y probada de niveles sucesivos, cada uno sirviendo de sustentación al siguiente; 2) Permitir la comparación dentro y entre organizaciones, utilizando niveles de madurez.

En CMMI, la representación ***continua*** tiene los atributos siguientes: 1) Permite seleccionar el orden de mejora que coincida con los objetivos de negocio de la organización, disminuyendo las áreas de riesgo de la organización; 2) Permite la comparación entre organizaciones en cada área de proceso, o cotejando los resultados transformando a etapas (niveles de madurez) equivalentes.

CMMI utiliza los conceptos siguientes: a) Niveles de madurez (ej. inicial, administrado,…), b) Áreas de proceso (ej. planificación del proyecto, desarrollo de requerimientos,…) organizadas en categorías (ej. Ingeniería, Soporte,…), c) Metas genéricas y específicas (ej. manejo de requerimientos, institucionalizar la administración de proyectos,…), d) Prácticas genéricas y específicas (ej. manejo de cambios a requerimientos, capacitación de participantes,…) y e) Características comunes (ej. compromiso con el desempeño, habilidad de realización,…).

AGD utiliza las categorías de áreas de proceso de la CMMI para estructurar su conjunto de productos, asegurándose de cubrir todas las categorías del modelo (Administración de Proyecto, Administración de Proceso, Ingeniería y Soporte). Además AGD cubre la mayoría de los conceptos que utiliza CMMI pues incorpora los conceptos de las técnicas de Proceso de Software en equipo (TSP) y Proceso Personal de Software (PSP) que se ha comprobado abarcan gran parte de las componentes del CMMI [DMH02].

2.5Desarrollo de Software Orientado-a-Objetos

En AGD se establece un conjunto de sub-procesos asociado a la realización de cada producto-del-trabajo requerido, en forma similar a como lo sugiere el Desarrollo de Software Orientado-a-Objetos (EOS, del Alemán: Objektorientierten Software-Entwicklung) (3 en tabla 2-1) [H97] [H03]. EOS administra el proceso mediante la realización de sub-procesos similares que forman una estructura de fractales de proceso, organizados en 4 fases: análisis, diseño, implementación y operación.

Los procesos de desarrollo se ligan a la estructura del sistema y sus unidades arquitectónicas principales (Sistemas, Componentes y Módulos). EOS modela el proceso de software mediante cinco sub-procesos: desarrollo, administración de calidad, administración de configuración y soporte, uso y evaluación así como administración de proceso.

De manera semejante AGD especifica un conjunto de sub-procesos que iteran sistemáticamente para obtener cada uno de los productos-del-trabajo. Los sub-procesos siguen los pasos definidos por la técnica de Proceso de Software en Equipo (TSP), aunque el orden en las iteraciones es flexible. Esta flexibilidad significa determinar el orden de ejecución de los cinco sub-procesos siguientes durante el primer sub-proceso (Lanzamiento & Estrategia) y evaluar el resultado obtenido mediante dicho orden, en el último sub-proceso (Pos-término).

2.6Proceso de Software en Equipo e Individual

AGD establece el trabajo en grupo como característica primordial del desarrollo de software y en particular propone organizar la realización en equipo en forma semejante al Proceso de Software en Equipo / Team Software Process (TSP) (4 en tabla 2-1) [H99]. TSP provee guía específica de cómo ingenieros de software pueden trabajar eficazmente en un equipo de alto desempeño. Utiliza los conceptos siguientes: a) metas del proyecto, b) roles (ej. líder del equi-

po, administrador del desarrollo,…), c) proceso de equipo, d) plan del proyecto, e) plan balanceado, f) análisis de riesgo, g) comunicación entre participantes del equipo, h) coordinación de equipo, e i) seguimiento de estado y reportes del proyecto.

En TSP se utiliza un proceso de equipo mediante varios ciclos de desarrollo para construir el producto final. Cada ciclo inicia con un lanzamiento seguido de siete pasos: estrategia, planificación, requerimientos, diseño, implementación, prueba y post-mortem.

Usando TSP se considera una disciplina para el trabajo individual, semejante a la del Proceso Personal de Software / Personal Software Process (PSP) (5 en tabla 2-1) [H95]. PSP provee guía específica de cómo el ingeniero de software, individualmente, puede mejorar continuamente su desempeño. Utilizando las prácticas siguientes: a) métricas personales, b) disciplina de proceso, c) estimación y planificación y d) administración de calidad.

AGD considera al proceso en equipo como un conjunto de siete sub-procesos susceptibles de utilizarse para realizar cualquiera de los productos-del-trabajo incluidos en el conjunto de productos (PS). Los sub-procesos son: Lanzamiento & estrategia, Planificación, Requerimientos, Análisis & Síntesis, Implementación, Prueba y Pos-término.

Cuando se utiliza AGD hay flexibilidad en la ejecución de los sub-procesos, ya que para cada producto se determina el orden y recursos a utilizar durante el primer sub-proceso (lanzamiento & estrategia). Las asignaciones de trabajo durante los sub-procesos se realizan en modo de trabajo colaborativo, y cuando se requiere de proceso individual, éste se estima como un caso especial del modo colaborativo.

2.7Proceso de Software Colaborativo

Uno de los cuatro modos de trabajo en grupo que integra el AGD es el trabajo colaborativo, que sigue el enfoque del Proceso de Software Colaborativo / The Colaborative Software Process (CSP) (6 en tabla 2-1) [W00]. CSP busca un mejor desempeño integrando la programación en pareja con roles de piloto y copiloto, usando una sola computadora y el proceso definido y repetible del PSP. CSP es una extensión del PSP y utiliza sus fundamentos conceptuales.

Mientras CSP se enfoca en dar un marco de disciplina al trabajo por parejas, para desarrollar un producto monolítico, el AGD combina el trabajo colaborativo de dos o más participantes con el PSP, para utilizarlo en un contexto de trabajo en grupo, con el objetivo de desarrollar sistemas con varios niveles de agregación parte-ensamble.

2.8Desarrollo iterativo e incremental

AGD se plantea como un método orientado a la obtención de un conjunto de productos mediante un proceso iterativo incremental, en forma similar al planteamiento de los trabajos relacionados siguientes:

1. Proceso Unificado de Racional / Rational Unified Process RUP (7 en tabla 2-1) [RUP03a] [RUP03b]. RUP es un proceso para ingeniería del software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades a los participantes, dentro de una organización de desarrollo. Es un proceso que utiliza al Lenguaje Unificado de Modelado (UML) para modelar el software y describe cómo aplicar mejores prácticas

de ingeniería de software mediante guías, plantillas, y guías estructuradas de herramienta, en todas las actividades críticas del ciclo de vida del software. Las mejores prácticas del RUP son: 1) desarrollo iterativo del software; 2) administración de los requerimientos; 3) utilización de arquitecturas basadas en componentes; 4) modelado visual del software, 5) verificación de la calidad del software y 6) control de los cambios del software.

Los creadores del RUP le atribuyen las características siguientes: Basado en modelo, Dirigido por riesgo y por casos de uso, Centrado en arquitectura, Orientado a objetos y Configurable. Su ciclo de vida consta de las fases siguientes:

- Origen / Inception – se define el alcance del proyecto.
- Elaboración / Elaboration – el proyecto se planifica, se especifican las características, se obtiene la arquitectura base.
- Construcción /Construction – se construye el producto.
- Transición / Transition – se pasa el producto al usuario final.

RUP maneja iteraciones de secuencias de actividades que cuentan con un plan y criterios de evaluación; entregando un producto ejecutable (interno o externo). También usa disciplinas, constituidas por una secuencia de actividades que produce un resultado de valor observable para un actor del negocio.

Las disciplinas núcleo del RUP son las siguientes: 1) Modelado del sistema / Business modeling, 2) Requerimientos / Requirements, 3) Análisis & Diseño / Análisis & Design, 4) Implementación / Implementation, 5) Prueba / Test y 6) Despliegue / Deployment. Además de las disciplinas mencionadas añade las disciplinas de soporte siguientes: a) Configuración & Administración del cambio / Configuration & Change management, b) Administración de proyectos / Project management y c) Ambiente / Environment.

2. Reemplazo del Sistema de Proceso y Despliegue de Comandos / Command Processing and Display System Replacement CCPDS-R (8 en tabla 2-1) [R98]. La TRW[17] lanzó el proyecto de cuatro años para el reemplazo del sistema de proceso y despliegue de comandos. Este sistema utilizó un proceso de desarrollo iterativo e incremental descrito por Walter Royce [R90] que consistió de seis iteraciones acotadas en el tiempo, que se tardaron cada una en promedio seis meses. Este enfoque concuerda con el proceso que se convirtió en "Rational Unified Process", al que contribuyó Royce.
3. Entrega Evolutiva Vs. Modelo en Cascada / Evolutionary Delivery Vs. Waterfall Model (9 en tabla 2-1) [G85]. Promueve una estrategia agresiva, recomendando la entrega frecuente (con ciclos de pocas semanas) de resultados útiles a los interesados en el producto desarrollado.

[17] TRW, compañía de EUA relacionada con el Departamento de Defensa, que provee productos de tecnología avanzada para el dominio aeroespacial, sistemas de información y mercados automotrices.

4. Modelo en Espiral del Desarrollo y Mejora del Software / A Spiral Model of Software Development and Enhancement (10 en tabla 2-1) [B85]. En este modelo Barry Boehm promueve que el equipo asigne prioridades durante los ciclos de desarrollo basándose en el riesgo. Formalizando así el enfoque de iteraciones-guiadas-por-riesgo y el uso de un paso del proceso dedicado a evaluar el riesgo, en cada iteración.
5. Prototipado Evolutivo / Evolutionary Prototyping (11 en tabla 2-1) [SB82]. Se argumenta que la especificación y el diseño se intercalan promoviendo el enfoque iterativo y evolutivo para la ingeniería de requerimientos y el desarrollo.

AGD se organiza mediante la realización concurrente de cuatro modos de trabajo en grupo y uno de ellos, específicamente el modo de trabajo en equipo, se forma mediante siete sub-procesos iterados sistemáticamente para la obtención de cada uno de los productos-del-trabajo que constituyen al conjunto de productos de Ingeniería, predefinido en PS.

RUP estructura su proceso en dos niveles de abstracción, el primero constituido por fases realizadas para avanzar en el desarrollo del producto de software deseado. Los objetivos de cada fase se logran mediante el segundo nivel de proceso, en el que se efectúan una secuencia de disciplinas núcleo y otras disciplinas de soporte.

AGD en contraste estructura los productos del conjunto Ingeniería, para que contenga todos los productos-del-trabajo necesarios y propone que cada uno de estos se obtenga con la repetición de un sólo nivel de proceso. Mediante trabajo en equipo se realiza, en forma flexible y disciplinada, un conjunto de siete sub-procesos: Lanzamiento & estrategia, Planificación, Requerimientos, Análisis & Síntesis, Implementación, Prueba y Pos-término.

2.9Manifiesto ágil

AGD utiliza algunas de las formas de desarrollo de software integradas en el Manifiesto Ágil (12 en tabla 2-1) [HC01] (www.agilealliance.org). Este manifiesto muestra y ayuda a otros a que utilicen las mejores formas de desarrollo. Favoreciendo a) la interacción entre individuos, b) el software funcionando, c) la colaboración del cliente y d) la respuesta al cambio. A la creación del manifiesto ágil contribuyeron los trabajos relacionados siguientes:

1. Familia Crystal / Crystal Family (13 en tabla 2-1) [C98]. Crystal es un conjunto de metodologías que incluye varios métodos de donde se escoge el más adecuado para cada proyecto específico. También contiene principios para adecuar los métodos a circunstancias variadas de proyectos diferentes. Cada método se marca con un color específico y se sugiere escoger el adecuado en función del tamaño y de que tan crítica es una falla. Mientras mas grandes sean los proyectos se requiere mejor coordinación y métodos más pesados. Los métodos están abiertos a cualquier práctica de desarrollo, herramientas o productos del trabajo.
2. Método Scrum / Scrum Method (14 en tabla 2-1) [S95]. Scrum se desarrolló para administrar el proceso de realización de software en un ambiente volátil. Es un enfoque práctico que se sustenta sobre flexibilidad, adaptabilidad y productividad. Scrup permite a los desarrolladores escoger las técnicas específicas de desarrollo de software, así como los métodos y las prácticas para el proceso de implementación. Incluye actividades adminis-

trativas enfocadas a la identificación sistemática de inconsistencias o impedimentos en el proceso de desarrollo y en las prácticas utilizadas.

3. Programación Extrema / Extreme Programming XP (15 en tabla 2-1) [B99]. XP es un conjunto de prácticas de ingeniería de software bien conocidas, cuyo propósito es permitir el desarrollo de software aún cuando los requerimientos sean vagos o que cambien constantemente. XP se distingue por la forma en que las prácticas se organizan para interactuar entre ellas. Algunas de su características principales son: 1) iteraciones cortas con entregas pequeñas y retroalimentación rápida; 2) participación cercana del cliente; 3) comunicación y coordinación constantes; 4) reensamble continuo, 5) integración y prueba continuos, 6) propiedad colectiva del código y 7) programación por parejas.
4. Desarrollo de software adaptable / Adaptive Software Development ASD (16 en tabla 2-1) [H00a]. ASD promueve un paradigma de desarrollo incremental e iterativo, para construir prototipos constantemente. ASD proporciona un marco con suficientes ayudas para evitar que los proyectos caigan en el caos, pero no demasiadas, pues podrían suprimir la creatividad.
5. Programación pragmática / Pragmatic Programing PP (17 en tabla 2-1) [H00]. PP introduce un conjunto de mejores prácticas. PP es una colección de 70 consejos enfocados a los problemas cotidianos. Estas prácticas se dirigen a: a) desarrollo incremental e iterativo, b) prueba rigurosa y c) diseño centrado en el usuario.
6. Proceso vivo para desarrollo de software / Living Software Development Process LSDP (18 en tabla 2-1) [GMP03]. LSDP utiliza un marco de referencia para proceso basado en conceptos de patrones de proceso y artefactos del trabajo. El marco de referencia permite a los ingenieros de software definir, evolucionar y aplicar un proceso de desarrollo flexible, considerando las necesidades diarias del proyecto. LSDP permite realizar una mejora evolutiva del proceso junto con el ajuste estático y dinámico de los modelos de proceso.
7. Modelado ágil / Agile Modeling AM (19 en tabla 2-1) [A02]. AM es un enfoque para realizar actividades, con énfasis en prácticas de modelado y principios culturales (ej. comunicación, estructura de la organización, formas de trabajo). Se pretende que los desarrolladores produzcan los modelos avanzados suficientes para satisfacer necesidades de diseño detallado y documenten al producto. La meta es mantener la cantidad de modelos y documentación en un nivel bajo.
8. Arquitectura basada en modelos e ingeniería de software / Model-Based Architecting and Software Engineering process Mbase (20 en tabla 2-1) [BP01]. Mbase introduce extensiones al modelo en espiral: para evitar las incompatibilidades (clash) de modelos (mediante modelos: de éxito del proyecto, de producto, de proceso y de propiedad) y para fomentar la relación ganar-ganar de los interesados en el proceso; introduciendo también consideraciones acerca de riesgos en los puntos de medición de avance “anchor”.
9. Desarrollo Internet-veloz / “Interner-speed development ISD (21 en tabla 2-1) [BLP01] [CY99]. ISD se refiere al software que se requiere entregar rápido, con ciclos de desarrollo cortos. Mediante un marco que incluye: a) manejadores de tiempo, b) dependencias de la calidad y c) ajustes al proceso. En este contexto “ajuste de proceso” significa enfocarse en gente competente en lugar de en el proceso. Las personas maduras y talentosas requieren menos de un proceso.

   El fundamento teórico del ISD es el desarrollo ametódico [BTT92] [TRT01] en donde el desarrollo de software consiste en un conjunto de procesos aleatorios y oportunistas go-

bernados por accidente. Los procesos son simultáneos, se traslapan y tienen huecos y el desarrollo ocurre en formas únicas e ideográficas. El desarrollo es negociado con concesiones y caprichoso, lo opuesto de predefinido, planificado y acordado en forma compartida.

10. Desarrollo guiado por características / Feature-driven development FDD (22 en tabla 2-1) [PF02] [CLL00]. FDD es un método orientado al proceso, para realizar aplicaciones críticas para el negocio. Abarca desarrollo iterativo y prácticas efectivas en la industria. FDD hace énfasis en aspectos de calidad durante el proceso, incluye entregas frecuentes de resultados tangibles, y monitoreo preciso del progreso en el proyecto.
11. Método Dinámico para Desarrollo de Sistemas / Dynamic System Development Method DSDM (23 en tabla 2-1) [S97-a]. DSDM en lugar de fijar la cantidad de funcionalidad en un producto y después ajustar el tiempo y los recursos para lograr la funcionalidad preestablecida, prefiere fijar el tiempo y los recursos y ajustar la cantidad de funcionalidad de acuerdo a ellos.
12. Sincroniza y estabiliza / Synch-and-stabilize de Microsoft (24 en tabla 2-1) [CS97]. Sincroniza y estabiliza es para desarrollar software que cambia rápidamente, hasta en forma caótica, en organizaciones emergentes [TBK99]. El desarrollo se realiza mediante equipos pequeños de 3 a 8 participantes trabajando en paralelo y programadores individuales que se integran como si constituyeran un equipo grande.

    Cada equipo es libre de desarrollar sus diseños en forma incremental, sincronizando sus cambios frecuentemente con los equipos restantes. Sus conceptos principales son: un producto diario, cero defectos y puntos de verificación de avance (milestones).
13. Desarrollo de software RADical / RADical software development (25 en tabla 2-1) [BH94]. RADical identifica qué elementos, como en RAD, son el catalizador de una transformación radical o reingeniería, de cómo desarrollar software. Fomenta que una de las primeras ideas a cambiar es la noción tradicional de los ciclos de vida del desarrollo de sistemas.

    El desarrollo RADical se enfoca en tres áreas amplias: el cliente, el producto y el proceso, a diferencia de los enfoques predominantes que se dirigen hacia alguna técnica específica (ej. prototipado, timebox). Proponiendo que el desarrollo de software RADical es un ciclo de vida dirigido hacia el cliente que: a) entrega soluciones de calidad, b) es un proceso evolutivo, c) usa técnicas de ingeniería de aplicación continua, d) se realiza por un equipo profesional, e) se administra el proyecto con tiempo-restringido (timebox), f) lo habilitan herramientas poderosas de desarrollo y g) tiene como resultado beneficios profundos en productividad.
14. Desarrollo Rápido de Aplicaciones / Rapid Application development RAD (26 en tabla 2-1), inspirado en las enseñanzas de James Martín [M91]. RAD define un proceso iterativo obtenido a partir de la reunión de Enero de 1994, donde un grupo de 16 practicantes del desarrollo rápido se reunieron en el Reino Unido. Éste proceso se convierte en el método DSDM.

    RAD establece un proceso para desarrollo de software que permita construir sistemas utilizables en un periodo de 60 a 90 días. Frecuentemente establece compromisos acerca de aspectos económicos y/o de calidad. Sugiriendo la negociación de temas económicos, fechas de entrega y calidad, ya que una solución del 80% se puede realizar en el 20 % del tiempo requerido para realizar la solución total.

AGD cuenta con los mecanismos para la adecuación de las características del desarrollo de software para cada proyecto individual, proporcionando alternativas claras y explícitas principalmente para el conjunto de productos (§ 6.4.1; ej. PS1, PS2,…,PSn, donde PS1 es el primer conjunto de productos, PS2 es el segundo conjunto alternativo, etc.), los niveles estructurales del producto (§ 6.4.2; ej. S1, S2,…, Sn, donde S1 es el primer conjunto de niveles estructurales, S2 es el segundo conjunto alternativo, etc.) y el conjunto de sub-procesos (§ 6.5.4; ej. P1, P2,…, Pn, donde P1 es el primer conjunto de sub-procesos, P2 es el segundo conjunto alternativo, etc.).

La definición de AGD proporciona flexibilidad para utilizar cualquier práctica de desarrollo, herramientas o productos del trabajo. Establece una configuración inicial, en cada una de las áreas mencionadas, que se puede modificar de acuerdo a las exigencias de la situación, considerando un conjunto de prácticas de ingeniería de software bien conocidas, para el desarrollo incremental e iterativo encaminado a construir prototipos constantemente.

2.10 Modelo de proceso para la industria del software

Los modos del trabajo en grupo del AGD se especifican mediante el patrón de proceso del modelo mexicano para el proceso de software.

El Modelo de Proceso para la Industria del Software (MOPROSOFT) (27 en tabla 2-1) [O03]. MOPROSOFT se desarrolló en el contexto del Programa para el Desarrollo de la Industria de Software (PDIS- ProSoft) de la Secretaría de Economía (México). ProSoft estableció siete estrategias y dentro de su estrategia 6: "Alcanzar niveles internacionales en capacidad de procesos", se obtuvo MOPROSOFT. Con el objetivo de definir un modelo de procesos y de evaluación apropiado para la industria mexicana de software.

MOPROSOFT organiza el desarrollo mediante tres categorías de procesos: de Alta dirección (Gestión del Negocio), de Gestión (de Proceso, de Proyecto y de recursos) y de Operación (Administración de Proyectos Específicos, Desarrollo y Mantenimiento de Software). A nivel del proceso de desarrollo maneja los conceptos de: Ciclos del Desarrollo, Fases de un Ciclo y Actividades de una Fase. Utilizando un patrón para los procesos, que incluye: una Definición general del proceso, Prácticas y Guías de ajuste.

La utilización del patrón de proceso de MOPROSOFT proporcionó una estructura homogénea a la descripción de los modos de trabajo en grupo del AGD.

2.11 Metamodelo para ingeniería del proceso de software

Para proponer una forma de desarrollo de software alterna se requirió analizar el proceso comúnmente seguido en la industria del software, para este propósito una de las fuentes es SPEMS de la OMG, ya que agrupa a una muestra representativa de las prácticas seguidas por las compañías que desarrollan software.

El metamodelo para ingeniería del proceso de software / "Software Process Engineering Metamodel SPEMS (28 en tabla 2-1) [OMG-SPEMS02]. SPEMS describe un proceso de desarrollo de software concreto, o una familia de procesos de desarrollo de software relacionados. Se utiliza un enfoque orientado a objetos para modelar una familia de procesos de software mediante el lenguaje UML.

SPEMS usa la arquitectura de OMG de cuatro niveles de abstracción: en el inferior M0 está el proceso de producción del mundo real; en el nivel M1 ubica la definición del proceso. (Ej. Proceso Unificado de Racional, DMR Macroscope, …); en el nivel M2 incluye el metamodelo y sirve como patrón para el nivel M1; en el nivel M3 esta el meta-metamodelo del Meta Object Facility (MOF). La especificación del SPEMS se estructura como un perfil UML, y provee un metamodelo soportado-en-MOF.

El SPEMS lo utilizamos para contrastar la forma de desarrollo de software propuesta en el AGD y como referencia de algunos conceptos utilizados.

2.12 El control del desarrollo en los trabajos relacionados

En el análisis de la información existente acerca de los conceptos y trabajos relacionados, recopilamos información acerca de la forma en que controlan el proceso de desarrollo. Esta información la incluimos en la columna Control de la tabla 2-1, mediante el nombre de los elementos utilizados en el monitoreo y determinación de acciones correctivas.

En la tabla 2-1 las tres primeras columnas contienen datos de identificación (Número en la 1, Referencia en la 2 y Nombre del Elemento en la 3), la columna 4 lista los elementos utilizados para el control y en la columna 5 anotamos las interrelaciones entre trabajos (Influenciado por).

En la columna Control, se incluyen dos guiones "—", en caso de no haber un elemento del que se recopilen mediciones de algunas de sus característica, para utilizarlas explícitamente en el control y la toma de decisiones tendientes a mejorar el desempeño del proceso.

El renglón 2 del CMMI se marcó con "—" en la columna Control, porque menciona el uso de información relacionada con el proceso, el proyecto y el producto, pero es muy general y no propone alguna forma específica de control.

En la columna Influenciado por, los dos guiones "—", indican que no se encontró mención explícita de que el trabajo correspondiente hubiera sido influenciado por algún otro elemento.

Del renglón siete al once, los trabajos referenciados se consideran como instancias del enfoque de desarrollo iterativo e incremental. En estos trabajos, todos los casos utilizan características del proceso para controlar el desempeño.

Los trabajos listados en los renglones doce al veintiséis son representativos de los métodos ágiles. En estos, para monitorear y controlar: 4 casos usan características de los recursos (12, 21, 23 y 24), 1 caso considera características del producto de software (16) y 10 casos restantes continúan usando características del proceso.

Los valores de las características de los elementos incluidos en la columna Control se recolectan, evalúan y utilizan para proponer cambios en el desarrollo que repercutan en la calidad del producto de software o en el desempeño del proceso realizado (ej. modificar la cantidad de tiempo dedicado a cada tipo de actividad, en función del tratamiento de la información acerca del proceso y sus resultados).

Mediante la información incluida en la columna Control de la tabla 2-1, determinamos que en el 75% de los elementos (21 casos) se utilizan principalmente características del proceso (ej. fases, disciplinas, actividades, pasos), en el 14% de los elementos (4 casos) se usan características de los recursos (ej. tiempos, personas), en 4% de los elementos de la tabla (1 caso)

se da el papel principal a características del producto de software (ej. prototipos) y 4% de los elementos de la tabla (1 caso) se enfocan a modelos (ej. metamodelos y transformaciones).

Es evidente en la práctica actual (representada por los trabajos relacionados que se analizaron) la tendencia de dar la máxima importancia al proceso y sus características, pues las mediciones se acumulan por fases o actividades, para tomar decisiones durante la planificación, monitoreo y adecuación del proceso de software.

Tabla 2-1 Elementos relacionados al Desarrollo Iterativo Incremental y Ágil.

Num.	Referen-cia	Nombre del E1emento	Control	Influen-ciado por
1	[S06]	Ingeniería Dirigida por Modelos / Model-Driven Engineering MDE (2006)	- Metamodelo de dominio específico - Generador de transformaciones	--
2	[C04] [PC91]	Modelo de Madurez de Capacidad, integrado / Capability Maturity Model, Integration CMMI (2002) y Modelo de Madurez de Capacidad / Capability Maturity Model CMM (1991)	--	--
3	[H97]	Desarrollo de software Orientado a Objetos / Object Oriented Software Development EOS (1997)	- 5 sub-procesos - Componentes del producto - Desarrollo mediante fases, organizadas en fractales	--
4	[H99]	Proceso de Software en Equipo / Team Software Process TSP (1999)	- Ciclo - Pasos-de-proceso - Ciclo de pasos y Tareas	5, 2
5	[H95]	Proceso Personal de Software / Personal Software Process PSP (1995)	- Paso de proceso - Ciclo de pasos	2
6	[W00]	Proceso de Software Colaborativo CSP (2000)	- Paso de proceso - Ciclo de pasos	15, 5
7	[RUP03a]	Proceso Unificado de Racional / Rational Unified Process RUP (1990s)	- Fases (proceso grueso) - Iteración de disciplinas (proceso detallado) - Mejores prácticas	8

(Continua tabla 2-1) Elementos relacionados al Desarrollo Iterativo Incremental y Ágil

Num.	Referencia	Nombre del E1emento	Control	Influenciado por
8	[R98]	Reemplazo del Sistema de Proceso y Despliegue de Comandos / Command Processing and Display System Replacement CCPDS-R (1987)	- Fases (proceso grueso) - Iteración de flujos-de-trabajo (proceso detallado)	--
9	[G85]	Entrega Evolutiva Vs. Modelo en Cascada / Evolutionary Delivery Vs. Waterfall Model (1985)	- Ciclos de pocas semanas - Entrega frecuente de resultados	--
10	[B85]	Modelo en Espiral del Desarrollo de Software / A Spiral Model of Software Development and Enhancement (1985)	- Iteraciones guiadas por riesgo - Pasos (proceso detallado)	--
11	[SB82]	Prototipado Evolutivo / Evolutionary Prototyping (1984)	- Iteraciones de requerimientos y desarrollo - Prototipo evolutivo	--
12	[HC01]	Manifiesto ágil / Agile manifesto (2001)	- Interacción entre individuos - Software funcionando - Colaboración del cliente y Respuesta al cambio	6, 16, 15, 14, 13
13	[C98]	Familia Crystal / Crystal Family (1998)	- Elección de métodos - Adecuación de métodos - Abierto a cualquier práctica	--
14	[S95]	Método Scrum / Scrum Method (1986 – 1999)	- Elección de técnicas, métodos y prácticas - Identificación sistemática de inconsistencias	--

(Continua tabla 2-1) Elementos relacionados al Desarrollo Iterativo Incremental y Ágil

Num.	Referen-cia	Nombre del E1emento	Control	Influen-ciado por
15	[B99]	Programación Extrema / Extreme Programming XP (1999)	- Prácticas de ingeniería - Interacción entre prácticas	9, 11
16	[H00a]	Desarrollo de software adaptable / Adaptive Software development ASD (2000)	- Producción de prototipos constantemente - Ayudas para evitar el caos	25
17	[H00]	Programación pragmática / Pragmatic Programing PP (2000)	- Mejores prácticas - 70 consejos - Prueba rigurosa y diseño centrado en el usuario	--
18	[GMP03]	Proceso vivo para desarrollo de software / Living Software Development Process LSDP (2003)	- Marco de referencia del proceso - Patrones de proceso - Artefactos del trabajo	23, 15
19	[A02]	Modelado ágil / Agile Modeling AM (2002)	- Actividades - Modelado	7
20	[BP01]	Arquitectura basada en modelos e ingeniería de software / Model-Based Architecting and Software Engineering Mbase (2001)	- Modelo de proyecto - Modelo de producto - Modelo de proceso	26
21	[CY99]	Desarrollo Internet-veloz / Internet Speed Development ISD (1999- 2001)	- Manejador de tiempo (Desarrollo ametódico) - Dependencias de calidad - Ajustes al proceso	24

(Continua tabla 2-1) Elementos relacionados al Desarrollo Iterativo Incremental y Ágil

Num.	Referencia	Nombre del E1emento	Control	Influenciado por
22	[PF02]	Desarrollo Guiado por Características / Feature Driven Development FDD (1999)	- Proceso - Desarrollo iterativo - Prácticas industriales	--
23	[S97-a]	Método Dinámico para Desarrollo de Sistemas / Dynamic System Development Method DSDM (1997)	- Fija tiempo y recursos disponibles - Ajusta cantidad de funcionalidad	26
24	[CS97]	Sincroniza y estabiliza / Synch-and-stabilize (1995 – 1997)	- Equipos pequeños - Trabajo paralelo - Independencia de diseños, sincronizando cambios	26
25	[BH94]	Desarrollo de software RADical / RADical software development (1994)	- Ciclo de vida dirigido hacia el cliente - Producto de calidad - Proceso evolutivo y técnicas de ingeniería	--
26	[M91]	Desarrollo Rápido de Aplicaciones / Rapid Application Development RAD (1994)	- Proceso iterativo - Compromisos económicos y de calidad - Solución de 80% en 20% del tiempo	9, 10, 11
27	[O03]	Modelo de Proceso para la Industria del Software MOPROSOFT (2003)	- Ciclo - Fase - Actividad	2
28	[OMG-SPEMS02]	Metamodelo de Ingeniería para el Proceso de Software / Software Process Engineering Metamodel SPEMS (2002)	- Fase - Disciplina-Iteración - Actividad-Paso	7

El enfoque propuesto para el método AGD representa un cambio del elemento al que se otorga mayor importancia, pues se dirige a la obtención del producto de software mediante la construcción de un conjunto de productos-del-trabajo, predefinido. Usando AGD los resultados se obtienen realizando un proceso disciplinado que recolecta mediciones y las acumula por producto-del-trabajo obtenido (no por fase o actividad). Así las decisiones se toman evaluando los recursos utilizados para la construcción de los diferentes tipos de producto-del-trabajo (ej. fólder de requerimientos, fólder de diseño de alto nivel,…).

2.13 Conclusión

El método AGD sigue la tendencia de evolución del desarrollo de software amalgamando los conceptos y contribuciones de los trabajos relacionados.

La utilización de los conceptos y técnicas que forman parte de los trabajos relacionados y la inclusión de complementos pertinentes, se realizó para definir las características de un producto de software y especificar el proceso adaptable para la ingeniería del producto de software especificado

Los trabajos relacionados descritos en las áreas de desarrollo de software ágil, desarrollo iterativo e incremental y mejora del proceso de software, proporcionan la mayoría de las características deseables para un método de desarrollo de software apropiado para las aplicaciones y plataformas tecnológicas actuales.

Del análisis de las características utilizadas para el control del desarrollo de software se hace evidente que en general en los enfoques usados para desarrollar software se da mucha mayor importancia a las características del proceso que a las características del producto.

3 Método de Desarrollo Arquitectónico en Grupo

3.1Introducción

En este capítulo se describe primero en un nivel de abstracción alto, al método para desarrollo de software Arquitectónico en Grupo (AGD), con el propósito de explicitar las relaciones entre sus componentes principales (§ 3.2). En seguida incluimos también la descripción detallada de su componente principal, el PS (§ 3.3)

El método AGD sintetiza una estructura diferente para obtener un cambio profundo respecto la forma común de desarrollar software descrita mediante el metamodelo para la ingeniería del proceso de software (SPEMS) y su principal instancia el proceso unificado de Rational (RUP) (§ 3.2). El componente principal del AGD es el conjunto de productos (PS) que contiene a todos los productos-de-trabajo necesarios para desarrollar un software. PS se definió apegado a modelos de referencia (§ 3.2.1). Para desarrollar el PS se realiza el Proceso Dirigido por Modelos (MDP), que se integró mediante modos de trabajo-en-grupo ajustables (§ 3.2.2).

En el capítulo siguiente se describe detalladamente el segundo componente principal del método AGD, el proceso MDP.

3.2Vista General del Método AGD

El método de Desarrollo Arquitectónico en Grupo / Architectural Group Development (AGD) [GM04] tiene como elemento principal el conjunto de productos resultantes (PS); de esta manera los productos dirigen el proceso requerido para obtenerlos (ubicado en un segundo plano). Esta disposición ocasiona que el método se dirija hacia los productos-del-trabajo, de los cuales la arquitectura del software es un elemento central.

El enfoque dirigido hacia productos simplifica el proceso de desarrollo propuesto en los documentos normativos (SPEMS, RUP), pues el PS reemplaza a uno de los dos niveles de proceso. De los dos niveles de proceso que forman a los mencionados procesos estándar: 1) las fases y 2) las disciplinas (workflows), se elimina el nivel de fases. Esta simplificación evitará una terminología confusa, que actualmente no parece ayudar a los practicantes.

AGD está constituido por dos dimensiones, mostradas en la figura 3-1:

1. Conjunto de Productos / Product Set (PS) [GJM00] es una jerarquía de sub-conjuntos de modelos que desarrolla productos de la categoría de Ingeniería (de las áreas de proceso del CMMI[18] [C04]). El PS se muestra en las columnas de la figura 3-1 y se mide en unidades de tamaño, ej. Líneas de código, líneas de texto, párrafos o figuras.
2. Proceso Dirigido por Modelos / Model Driven Process (MDP) [GM02], está compuesto de cuatro modos de trabajo-en-grupo: un trabajo en equipo y tres modos de trabajo-en-grupo complementarios. Los modos se muestran en los renglones de la figura 3-1 y el trabajo realizado se mide en unidades de esfuerzo (horas-hombre).

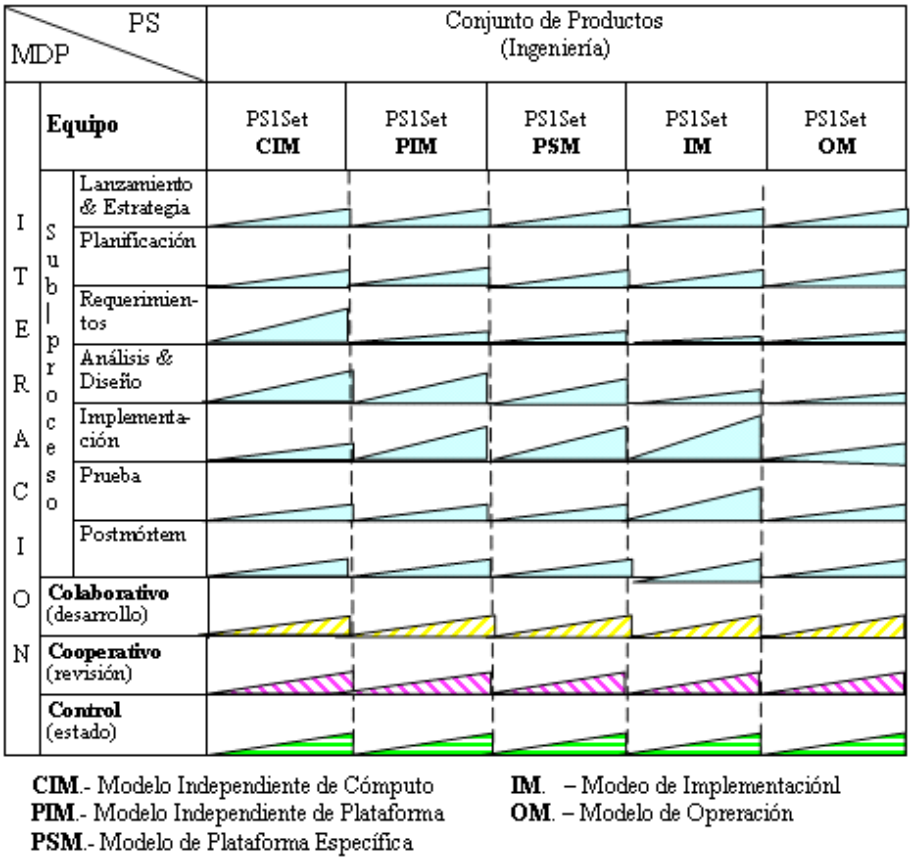
El método AGD asigna y maneja, el desarrollo de los productos-de-trabajo organizados en el Conjunto de Productos (PA: *Product Set*), así como la responsabilidad asociada en cada tarea resultante. Se utiliza, para este fin, el Proceso Dirigido por Modelos (MDP: Model Driven Process), que tiene como finalidad producir software de alta calidad que satisfaga las necesidades de los usuarios finales.

MDP es un proceso de buen desempeño (cuyo calendario es predecible y se mantiene dentro del presupuesto especificado), centrado en arquitectura debido a que se enfoca al PS constituido por modelos que se ajustan: a un modelo de referencia para sistemas de información y a un estilo arquitectónico explícito.

AGD establece cómo un equipo aplica un enfoque pragmático para desarrollar software. Las prácticas principales del método, son: realizar mediante iteraciones, administrar los requerimientos, usar un estilo arquitectónico dirigido por modelos, construir ajustado a modelo de referencia, especificar y verificar la calidad del software, controlar los cambios aplicados al software y usar la sinergia de cuatro modos de trabajo-en-grupo: Equipo / *Team,* Colaborativo / *Collaborative*, Cooperativo / Cooperative y de Control / *Control*.

Este método permite la ingeniería[19] de productos siguiendo un proceso ordenado, sistemático e iterativo de trabajo-en-grupo, organizado en sub-procesos de forma que facilite el trabajo en equipo. Cuidando que el proceso resultante abarque los elementos de las técnicas de Proceso de Software Personal / Personal Software Process (PSP) y Proceso de Software en Equipo / Team Software Process (TSP).

[18] CMMI - Capability Maturity Model, Integrated / Modelo de Madurez de Capacidad, integrado.
[19] Ingeniería es la aplicación de un enfoque sistemático, disciplinado y cuantitativo a estructuras, máquinas, productos, sistemas o procesos. [IEEE-610.12].



Constituido por el conjunto de productos (PS, con 5 PS1Sets, en las columnas) y el proceso dirigido por modelos (MDP, con 4 modos de trabajo-en-grupo, en los renglones: Equipo, Colaborativo, Cooperativo y Control). El Trabajo en Equipo se desglosa en sus 7 sub-procesos.

Figura 3-1 Método para el Desarrollo Arquitectónico en Grupo (AGD).

Para cada sub-proceso se describe quien lo realiza y como proceder (§ 4.4.6). La cantidad y alcance de los productos realizados en cada sub-proceso difieren en cada iteración y dependen del conjunto de productos realizado (ej. construcción del Modelo de Plataforma Específica / Platform Specific Model, **PSM**). En la figura 3-1 el área mayor para la columna PSM se ubica en el renglón correspondiente a Implementación, indicando que la mayor parte del modelo se obtiene dedicando mayor esfuerzo durante este sub-proceso.

1.
2.

La cantidad de modelos en cada PS1Set (ej. CIM, PIM, …) inicia siempre en cero y se representa por el punto donde se unen la hipotenusa y el cateto horizontal. La longitud del cateto vertical muestra la cantidad y alcance final de los resultados obtenidos. El triángulo con el cateto vertical más grande representa el sub-proceso que produce más modelos y cuyo alcance es mayor.

Para lograr que el método AGD se dirija hacia la arquitectura de software, los productos PS y los procesos MDP se ajustan a:

- Modelo de Referencia - CWSLR[20] [GJM99], para sistemas de información de tecnología actual ej. una aplicación Web que incluye la memoria organizacional[21] y el trabajo-en-grupo y
- Un Estilo Arquitecónico - basado en la MDA[22] [OMG03A], del grupo OMG, con sus modelos: Independiente de Cómputo, Independientes de Plataforma, y de Plataforma Específica.

CWSLR y MDA los ubicamos en el tercer nivel (N3), más abstracto, del entorno de desarrollo: Proceso de Software mediante Trabajo Cooperativo / Cooperative Work Software Process (CWSP) [GM02], organizado en tres niveles de abstracción (ver figura 3-2).

En el nivel N1 se ubican las herramientas que soportarán al método AGD. La primera herramienta a desarrollar es el Sistema De Soporte Para El Proceso De Software En Equipo (SiSoProS).

En el nivel N2 localizamos al AGD con sus dos componentes principales: el PS y el MDP. AGD utiliza las abstracciones del nivel N3 y a su vez proporciona conceptos y estructura a las herramientas del nivel N1.

En el nivel N3 se muestra EA(MDA, …) un estilo arquitectónico[23] fundamentado en la arquitectura dirigida por modelos (MDA) [OMG03A]. Este Estilo Arquitectónico (EA) se utiliza como referencia junto al modelo CWSLR, en la definición del conjunto de modelos PS del método AGD.

---

[20] CWSLR - modelo Trabajo Cooperativo: Almacenamiento, Lógica y Recursos / Cooperative Work: Storage Logic and Resources model [GJM99]

[21] Memoria organizacional – información acerca de decisiones tomadas y la forma de resolver problemas [WU91]

[22] MDA – Arquitectura Dirigida a Modelos / Model Driven Architecture [OMG03A].

[23] Estilo arquitectónico es una descripción de tipos de componentes y un patrón de transferencias de control y/o datos, llevado a cabo durante la ejecución. Es un conjunto de restricciones sobre una arquitectura –restricciones en los tipos de componentes y sus patrones de interacción-- y estas restricciones definen un conjunto o familia de arquitecturas que las satisfacen.

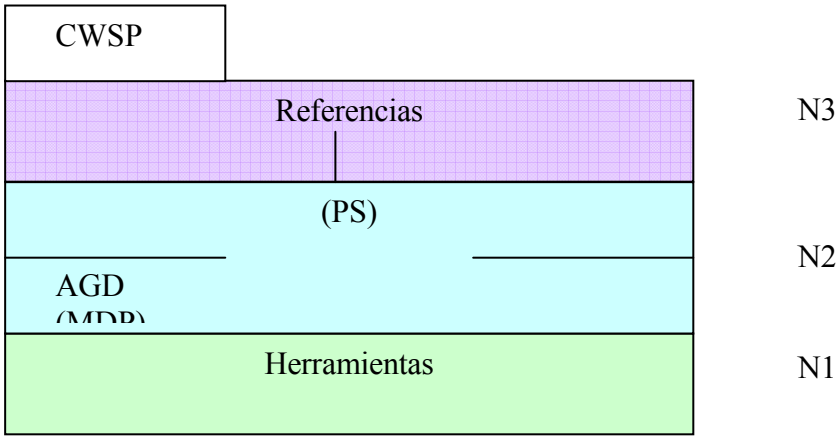


Figura 3-2 Niveles del entorno de desarrollo: Proceso de Software mediante Trabajo Cooperativo / Cooperative Work Software Process (CWSP)

En las siguientes secciones se completa una visión general del método AGD y sus elementos.

3.2.1 Conjunto de Productos (PS)

El producto de software a construir incluye un modelo ejecutable, que es parte del Conjunto de Productos / Product Set (PS) del método AGD. El modelo ejecutable es el que se carga en la memoria de las computadoras para interpretar sus instrucciones, modificar el estado de la memoria de datos y operar los dispositivos de entrada y salida requeridos.

PS tiene una estructura jerárquica en forma de árbol invertido (§ 4.2), que abarca los productos de las cuatro categorías de Áreas de Proceso que incluye la Integración del Modelo de Madurez de Capacidad / Capability Maturity Model Imtegration (CMMI) [SEI-TR-2002-011] [SEI-TR-2002-012]: 1) Administración de Proceso / Process Management, 2) Administración de Proyecto / Project Managenet, 3) Ingeniería / Engineering y 4) Soporte / Support.

Primordialmente nuestro trabajo se enfocó a definir los conjuntos de modelos de la categoría de Ingeniería, de áreas de procesos del CMMI. La categoría Ingeniería, abarca los modelos necesarios para la concepción del producto de software y su desarrollo hasta alcanzar la operación del modelo ejecutable, que forma parte del producto de software terminado. Cuando mencionemos al Conjunto de Productos (PS) se entenderá la rama que cubre la categoría de Ingeniería. Cuando haya necesidad de referirse al conjunto completo (abarcando tres categorías adicionales a la de Ingeniería) o una área de proceso en una categoría distinta, se hará explícitamente.

PS se definió tomando como base dos modelos que proporcionan vistas diferentes de una arquitectura: La MDA [OMG03A]; y el modelo CWSLR [GJM99] que corresponden a un producto o sistema de software.

Durante la determinación de los modelos a incluir en PS, se propuso un orden para el primer nivel de la categoría de Ingeniería (el nivel más abstracto, donde se ubican los PS1Sets). Este nivel contiene los tres conjuntos de modelos de la Arquitectura Dirigida por Modelos (MDA) (figura 3-3): CIM, PIM y PSM, donde cada uno de estos conjuntos agrupa modelos con perspectivas diferentes. PS, igual que MDA, pretende usar modelos para dirigir el curso de la comprensión, diseño, construcción, distribución, operación, mantenimiento y modificación de un producto de software.

MDA provee un enfoque abierto, independiente del proveedor para enfrentar el reto del cambio en el negocio y en la tecnología. Los modelos independientes de plataforma (que incluyen el comportamiento) pueden ser una propiedad intelectual separada del código específico para una plataforma, ayudando a aislar las aplicaciones de la evolución tecnológica. Además las aplicaciones, libres de cuestiones tecnológicas, serán capaces de evolucionar al ritmo propio del cambio en el negocio.

Concentrándose primero en los Modelos Independientes del Cómputo / Computation Independent Model (CIM), en segundo lugar se consideran a los Modelos Independientes de la Plataforma / Platform Independent Model (PIM) y en tercer término se incluyen los Modelos de Plataforma Específica / Platform Specific Model (PSM). En este nivel de PS1Sets, se complementan los tres conjuntos mencionados, con los conjuntos: Modelos de Implementación / Implementation Model (IM) y los Modelos para Operación / Operation Model (OM). El conjunto completo de PS1Sets es: CIM, PIM, PSM, IM y OM.

PS es un conjunto de productos-de-trabajo donde todos sus elementos se conocen, y se pueden desarrollar en el orden que se requiera. El conjunto de productos (para la categoría de Ingeniería), esta organizado en cuatro niveles: PS1Set, PS2Set, PS3Set y Productos-de-Trabajo / Work-Products. Estos niveles proponen un orden para construir los modelos (según su colocación); pero no se exige que forzosamente se realicen en dicho orden. Se recomienda desarrollar todos los modelos, con la flexibilidad de elegir su orden.

Cuando se conocen de antemano, los modelos incluidos en cada sub-conjunto. Si se da el caso de que falte desarrollar algún modelo, los implementadores tienen conciencia de ello y lo pueden considerar en los sub-procesos de Lanzamiento & Estrategia y de Planeación del siguiente ciclo del desarrollo.

La Arquitectura Dirigida por Modelos (MDA) utiliza la estructura de cuatro niveles (figura 3-3) del Grupo de Administración de Objetos / Object Management Group (OMG). Los niveles M1 (modelos), M2 (meta-modelos), y M3 (meta-meta-modelos) se muestran; pero el nivel M0 no se muestra. M0 contiene el universo de discurso o dominio del negocio [J95].

Todos los modelos de MDA están relacionados porque se basan en un meta-metamodelo muy abstracto, el Meta Object Facility o MOF, utilizado en el nivel M3. Todos los modelos utilizados en MDA están definidos en términos de componentes del MOF. Esto garantiza que todos los modelos en MDA se pueden comunicar unos con otros por ajustarse al MOF. En la figura 3-3, CIM y PIM son modelos MOF.

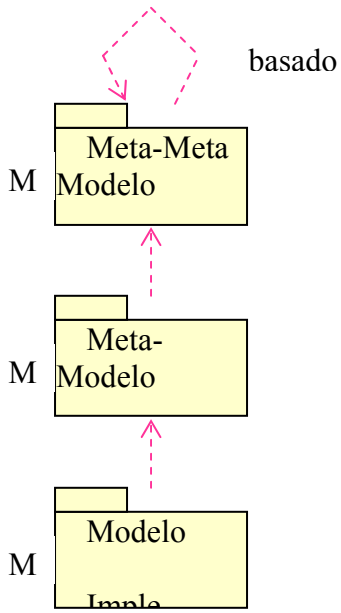


Figura 3-3 Los modelos de MDA en la Arquitectura de cuatro niveles del OMG.

La MDA es flexible, pues permite definir un mayor número de meta-modelos en el nivel M2 y meta-meta-modelos del nivel M3 [NW03]. Dicha facilidad de cambio se utilizó en esta investigación, cuando definimos al PS del método AGD, dado que se incluyeron los meta-modelos complementarios: conjunto de implementación IM y conjunto de operación OM.

El Conjunto de Productos PS *se completa*, por acuerdo del cliente con el equipo de desarrollo: asignándole el estado "*PS* Completo". Para tomar esta determinación, antes de iniciar cada proyecto, se define cuales modelos se realizarán para cada nivel del PS, dando oportunidad de cambiar la lista durante el sub-proceso SP-Lanzamiento & Estrategia, en cada iteración.

Los productos terminados se organizan para su presentación y reporte, como se muestra en la figura 3-4, estructurándose en carpetas. El conjunto de carpetas utilizado, incluye nombres usados comúnmente por los desarrolladores, que agrupan a modelos coin propósito de facilitar la utilización del método AGD.

Experimento: Fólder de Requerimientos	
AGD: Fólder de Visión/Planificación (FV) CIM-(CWSP Inicialización) CIM-(Planificación/Visión) CIM-(Alcance/Prioridad) CIM-(Resultados de Revisión de FV) *AGD: Fólder de Requerimientos (FR)* CIM-(Negocio existente) CIM-(Identificación de componentes)	CIM-(Integración del prototipo del sistema) CIM-(Requerimientos del sistema) PIM-(Ingeniería inversa) PIM-(Ingeniería directa) PIM-(Integración del prototipo del software) PSM-(restricciones de requerimientos) PSM-(Especificación de prueba de aceptación) PSM-(Requerimientos del software) PSM-(Resultados de Revisión de FR)
Experimento: Fólder de Diseño	**Experimento: Fólder de Usuario**
AGD: Fólder de Diseño (FD) PSM-(Arquitectura del componente) PSM-(Especificación prueba integración) PSM-(Análisis de arquitectura) PSM-(Estructura de division del trabajo y asignación de unidades) PSM-(Resultados de Revisión de FD)	*AGD: Fólder de Operación* IM-(Operación inicial) IM-(Resultados prueba de integración) OM-(Identificación de recursos) OM-(Distribución) OM-(Liberación de producto) OM-(Resultado de prueba de aceptación) OM-(Resultados de evaluación)
Experimento: Fólder de Implementación	**Experimento: Métricas (formas)**
AGD: Fólder de desarrollo de unidad Selección de requerimientos de software (desde FR) Selección de diseño general (desde FD) IM-(Diseño detallado) IM-(Especificación de prueba de unidad) IM-(Resultado de revisdión diseño detallado) IM-(Lista de funciones) IM-(Código fuente) IM-(Resultado de prueba de unidad) Reporte de defectos IM-(IM-(Resultados de Revisión de UDF)	*AGD: Formas* Bitácora de registro de tiempos (MDP-LOGT) Bitácora de registro de defectos (MDP-LOGD) Estrategia (MDP-STRAT) Liberación de producto y retroalimentación (MDP-PDF) Reporte de estado Semana/Ciclo (MDP-WEEK) Programa de tareas (MDP-SCHEDULE) Solicitud de cambio en configuración (MDP-CCR) Reporte de estado de la configuración (MDP-CSR) Reporte de inspección-detalle (MDP-INS-DET) Reporte de inspección-resumen (MDP-INS-SUM)

Figura 3-4 Coincidencia entre Carpetas de Experimentación y Carpetas del AGD.

La correspondencia de la MDA con las carpetas se realiza incluyendo únicamente dos niveles del *PS*: el nivel que incluye los PS1Sets y el nivel que incluye a los PS2Sets (PS1Set-(PS2Set) (ej. CIM-(Planificación/Visión)). Los fólderes de la figura 3-4 que agrupan a los PS1Sets y PS2Sets son las siguientes: Fólder de Planificación/Visión, Fólder de Requerimientos, Fólder de Diseño, Fólder de Desarrollo de Unidad y Fólder de Operación.

Se Incluyen en la figura 3-4 los nombres de los fólderes definidos en el método AGD, a los que se antepone el prefijo "AGD:"; y los que se definieron para reportar los resultados durante la experimentación realizada (§ 5.3), a los que se añade el prefijo "Experimento:".

En cuanto a la forma en que se desarrollan los modelos incluidos en PS, se propone el Proceso Dirigido por Modelos, que es sistemático e iterativo, descrito en forma general en la siguiente sección.

Se propone una forma flexible de organizar el trabajo para desarrollar los modelos incluidos en PS, ya que los participantes que elaboran los PS2Sets están en libertad de producirlos en una o varias iteraciones y conforme se requiera. Pudiendo incluso planificar una secuencia especial requerida por las características del producto.

Por ejemplo construir PIM-(Integración del Prototipo de Software) antes de desarrollar PIM-(Ingeniería Directa). Esta secuencia especial comúnmente se conoce como desarrollo de prototipo-rápido [G89], el cual es frecuente debido a que actualmente se tienen disponibles varios lenguajes de cuarta generación para llevar a cabo la implementación.

3.2.2 Proceso Dirigido por Modelos (MDP)

La forma natural para el proceso genérico Proceso Dirigido por Modelos / Model Driven Process (MDP) es el trabajo-en-grup, para realizar un proyecto de desarrollo de software. MDP especifica cuatro modos de trabajo-en-grupo, enlistados a continuación:

- Modo colaborativo / Collaborative mode [SWN03] [WK*00*] [N98].- desarrolla los modelos (productos), trabajando dos o más personas conjuntamente
- Modo cooperativo / Cooperative mode [A84] [A97-B] [IEEE-1028].– revisa y evalúa los modelos (productos)
- Modo Control */ Control mode* [SK01] [IEEE-1028].– reporta información del desempeño, determina acciones correctivas y planifica los esfuerzos de desarrollo y de administración necesarios para producir los modelos (productos)
- Modo en Equipo */Team mode* [H99].– desempeño del grupo como un equipo y proporciona un contexto (ej. inicio/terminación de sub-procesos y ciclos), a los otros modos de trabajo-en-grupo. Este modo usa un conjunto de sub-procesos que son una versión modificada de los Pasos de Proceso del Team Software Process (TSP) (ver figura 3-1, mostrando los siete sub-procesos del modo en Equipo, como renglones en el eje vertical, ej. Requerimientos).

Los cuatro modos de trabajo en grupo definen cada uno un conjunto de roles a los participantes asociados con las responsabilidades asignadas (§ 4.3). Aquí solamente mencionamos los roles del modo Equipo: 1)Miembro del equipo, 2) Líder del Equipo , 3) Administrador de Planeación, 4) Administrador de Soporte, 5) Administrador de Desarrollo y 6) Administrador de Calidad/Proceso.

Estos modos de trabajo-en-grupo pueden realizarse concurrentemente como se muestra en la carta de estados de la figura 3-5, separando a los estados mediante líneas punteadas (ej. TG-Desarrollo y TG-Sub-procesos).

Mediante diagramas de estado como el incluido en la figura 3-5, el modelo de proceso MDP se describe dinámicamente por medio de eventos. Por ejemplo, una inspección, hecha utilizando el modo Cooperativo se dispara (programa y realiza) cuando uno de los dos eventos siguientes ocurre: a) la terminación de un producto de trabajo (que se hizo utilizando el modo Colaborativo), b) la terminación de una iteración (la ejecución de los siete sub-procesos del modo de trabajo en Equipo).

Las reuniones de seguimiento del estado del proyecto se realizan en modo de Control. Estas reuniones se pueden programar periódicamente en un día y a una hora específica de cada semana.

El modo de Trabajo en Equipo es flexible[24], pues usa dos sub-procesos fijos y cinco sub-procesos asíncronos. Los sub-procesos fijos son: 1) el inicial en el que determina el orden y alcance de los modelos a realizar; y 2) el sub-proceso final en el que se evalúa el trabajo efectuado.

Los cinco sub-procesos asíncronos no requieren realizarse en una secuencia predeterminada, aunque se sugiere efectuarlos todos. Esta estructura del MDP permite que el proceso global sea flexible y busca explícitamente que sea incremental[25], cooperativo[26], directo[27] y adaptable[28].

En el sub-proceso inicial nombrado SP-Lanzamiento&Estrategia, el grupo de trabajo establece los modelos a obtener, y el orden en que se realizarán los otros subprocesos. Para establecer la estrategia específica, se considera que el producto de software tiene la estructura general siguiente: 1) El sistema es un ensamble de partes producto, 2) Los productos son un ensamble de partes componente, 3) Los componentes son un ensamble de partes módulo y 4) Los módulos son un ensamble de partes objeto.

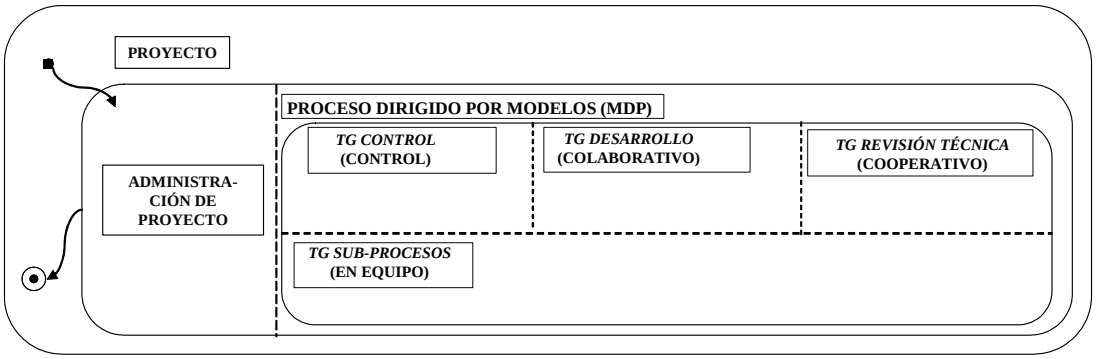


Figura 3-5 El proceso Model Driven Process (MDP).

[24] Flexible - proporciona un determinismo suficiente, para establecer relaciones variables entre la mayoría de los elementos.
[25] Incremental - entregas pequeñas de software, con ciclos de desarrollo rápidos.
[26] Cooperativo - interacción estrecha entre el cliente y el desarrollador.
[27] Directo - fácil de aprender y de modificar, contando con documentación suficiente.
[28] Adaptable - habilidad para hacer cambios y reaccionar adecuadamente a modificaciones de último momento.

En esta forma los términos ensamble y parte se usan recursivamente, por lo que se requiere especificar a que tipo de elemento se refieren (sistema, producto, componente, modulo u objeto).

Cada iteración de sub-procesos termina entregando un producto externo o interno (preferentemente ejecutable), que se refina (crece en detalles) de iteración en iteración hasta que se obtiene el producto de software final. El sub-proceso que se realiza al término de cada iteración es SP-PostTérmino, cuyo propósito es: 1) recopilar, analizar y registrar los datos del proyecto, 2) evaluar a los miembros del equipo de acuerdo con el rol desempeñado, 3) identificar la forma de mejorar el siguiente ciclo y 4) producir un reporte del ciclo que termina.

Con el objetivo de controlar la calidad del producto y del proceso, el método AGD define 23 formas de captura de datos (ver el fólder "Formas" en la figura 3-4), que se agrupan el conjuntos de resultados que coinciden con las áreas de proceso sugeridas por CMMI en las categorías: Administración de Procesos, Administración de Proyecto, y Soporte.

Los productos adicionales requeridos por las categorías de áreas de proceso diferentes a la de Ingeniería, AGD los permite agregar al PS para que se obtengan también utilizando el proceso MDP. La mejor forma de obtener los resultados definidos para cada área de proceso, se planifica en el contexto del modo de trabajo en Equipo. De esta forma se obtienen instancias de los productos del trabajo para cada área.

Para cubrir los requisitos adicionales (ej. del categoría Soporte), la solución de la mayoría de las normas y modelos relacionados con los procesos de software, es añadir nuevos procesos. Así, se agregan nuevos procesos en los casos siguientes: 1) RUP y sus Disciplinas de Soporte [RUP03] y 2) La normas ISO 12207 con cinco categorías y 25 tipos de proceso [ISO-15504] [IEEE-12207].

La descripción del MDP, completa la visión general del método AGD, ubicado en el nivel N2 del ambiente CWSP. A continuación se incluye una descripción a mayor detalle del conjunto de productos del AGD.

3.3Refinamiento del PS

Para nuestro trabajo de investigación, establecimos el Conjunto de Productos / Product Set (PS) [GM02], para satisfacer los siguientes requerimientos:

Nombre	Descripción
Conceptos de grupo	Se requiere incluir los elementos necesarios (ej. memoria de grupo, cooperación), para soportar el trabajo en grupo que se realiza en las organizaciones.
Modelos de Referencia	Durante la construcción de un producto de software se requieren modelos de referencia para identificar los productos-del-trabajo a desarrollar.
Especificación	En el desarrollo de cada producto-del-trabajo se requiere especificar el elemento del lenguaje UML a utilizar.
Transformación entre modelos	Identificar las transformaciones entre modelos con la finalidad de habilitar su realización automática.

Modelado conceptual del negocio	Para producir un modelo de proceso del negocio es necesario tener elementos explícitos en UML, que sean parte del modelo conceptual del componente en desarrollo.

Como estrategia para satisfacer estos enunciados (explicados en § 1.1.1), se otorgar importancia a la arquitectura del software, sobre otros elementos durante el desarrollo de software, estableciendo como meta principal la obtención de los productos, siendo estos los que dirigen a cualquier mecanismo o procedimiento involucrado (ej. el proceso).

PS incluye los productos-del-trabajo / Work Products (WP) requeridos en las áreas de proceso involucradas en el desarrollo del software. Utilizando como lenguaje principal para desarrollar estos WP el Lenguaje de Modelado Unificado / Unified Modeling Language (UML) [CSM03] [OMG05] [FGD06].

PS es un árbol invertido que consta de cinco niveles (figura 3-6), cuya raíz la denominamos PS y el siguiente nivel, incluye a las categorías en que se agrupan las áreas de proceso de la Integración del Modelo de Madurez de Capacidad / Capability Maturity Model Integration (CMMI) [C04]: 1) Administración del Proyecto, 2) Administración del Proceso, 3) Soporte y 4) Ingeniería. Cada nivel del PS es un conjunto ordenado, a su vez de otros conjuntos ordenados.

La categoría Ingeniería es una jerarquía de cuatro niveles, de los modelos técnicos (WPs), construidos en el desarrollo de componentes de software: PS1Sets, PS2Sets, PS3Sets y ProductosDelTrabajo /Work Products. El trabajo realizado en esta investigación se enfocó en la categoría de Ingeniería. En esta sección se tratan los WP requeridos y la secuencia explícita sugerida de los tipos de diagrama utilizados.

Para identificar los modelos técnicos producidos, cuando se sigue el método AGD, se usaron como referencia los elementos del modelo CWSRL (presentados en el anexo B), así como el estilo arquitectónico orientado a modelos de la Arquitectura Dirigida por Modelos / Model Driven Architecture (MDA) [MM03].

Las categorías de las áreas de proceso del primer nivel del PS y la forma en que se organizan se describe en § 3.3.1. Las categorías de áreas de proceso administrativas y de soporte se tratan conjuntamente en § 3.3.2. El primer nivel de la Categoría de Ingeniería, incluye los modelos de la MDA en los PS1Sets, explicándose en § 3.3.3. Los PS2Sets, son los productos principales a obtener en el desarrollo de software, detallándose en § 3.3.4. El contenido de los productos principales se incluye en los PS3Sets, especificándose en § 3.3.5. PS incluye algunos modelos adicionales a los incluidos en el UML, describimos los principales complementos en § 3.4.

3.3.1 Primer nivel del PS: Categorías de Áreas de Proceso

Seguimos el orden de lo más general, arriba: raíz, a lo más específico, abajo: quinto nivel. La raíz consiste de un sólo Conjunto de Productos / Product Set (PS) que contiene, en el primer nivel jerárquico un conjunto ordenado de cuatro categorías de áreas de proceso que concuer-

dan con las definidas en el CMMI: Administración de Proyecto, Administración de Proceso, Soporte e Ingeniería, como se muestra en la figura 3-6.

CMMI y sus categorías de áreas de proceso forman el nivel de mayor abstracción del PS, su primer nivel. Las cuatro categorías de áreas de proceso, incluidas en CMMI, contienen las áreas específicas que se incluyen en la figura 3-7. En total, las categorías contienen 25 áreas de proceso, distribuidas como sigue: ocho en la categoría de Administración de Proyecto, cinco en la categoría de Administración de Proceso, seis en la categoría de Soporte, y seis en la categoría de Ingeniería.

Para el segundo nivel del PS, nos interesó principalmenteb la categoría de áreas de proceso Ingeniería. En la categoría de Ingeniería, decidimos apegarnos a los modelos de MDA:

- Modelo Independiente de Cómputo / Computation Independent Model CIM,
- Modelo Independiente de Plataforma / Platform Independent Model PIM,
- Modelp de Plataforma Específica / Platform Specific Model PSM,

pues se requiere orientación a modelos.

Especificando que el orden de los elementos en el primer nivel de la categoría de Ingeniería, de izquierda a derecha, sugiere la secuencia más frecuente de realización de los modelos. En PS, la categoría Ingeniería esta subdividida en 5 PS1Sets: CIM, PIM, PSM, IM y OM, y es evidente que estos difieren de la lista de las áreas de proceso incluidas en la categoría de Ingeniería del CMMI de la figura 3-7.

En la definición del primer nivel de la categoría Ingeniería (PS1Sets) tomamos como referencias principalmente a la arquitectura dirigida por modelos MDA y al modelo CWSLR. Aunque el conjunto resultante de PS1Sets (figura 3-6) no tienen una relación directa con las áreas de proceso contenidas en Ingeniería de la CMMI (figura 3-7), se tuvo cuidado de incluir, en los PS2Sets del segundo nivel de la categoría de Ingeniería, los productos realizados en las áreas de proceso de la categoría de ingeniería del CMMI.

La ubicación en PS de los resultados requeridos por CMMI, se distribuyeron de acuerdo a la estructura de MDA:

- CIM – resultados independientes del cómputo
- PIM – resultados independientes de la plataforma
- PSM – resultados para plataforma específica
- IM – resultados de implementación (contiene al código)
- OM – resultados de operación

La estructura del PS para la categoría Ingeniería, de abajo hacia arriba en la figura 3-6, incluye en el nivel más bajo (detallado) a los productos-del-trabajo / *work-products (WP).* A su vez los WPs se agrupan en conjuntos *PS3-Sets*, incluidos en los conjuntos *PS2Sets* y estos reunidos en *PS1Sets,* del primer nivel jerárquico.

Dándose por terminado el desarrollo de un conjunto de productos (ej. un *PS1Set),* cuando todos los modelos del conjunto correspondiente se han realizado. En ese momento ocurre una revisión técnica formal de los resultados, mediante una reunión de trabajo-en-grupo.

3.3.2 Productos de las Categorías de Administración y Soporte

Los productos de las áreas de proceso incluidas en las categorías de 1) Administración de Proyecto, 2) Administración de Proceso y 3) de Soporte, se incluyen en las formas de captura utilizadas en el método AGD. Dichas formas se usan cuando se realiza el proceso dirigido por modelos MDP, como parte de la dinámica del modo de trabajo-en-grupo: "Equipo". El proceso MDP (capítulo 4) se utiliza para desarrollar cualquier producto del PS, principalmente los modelos de la categoría de Ingeniería.

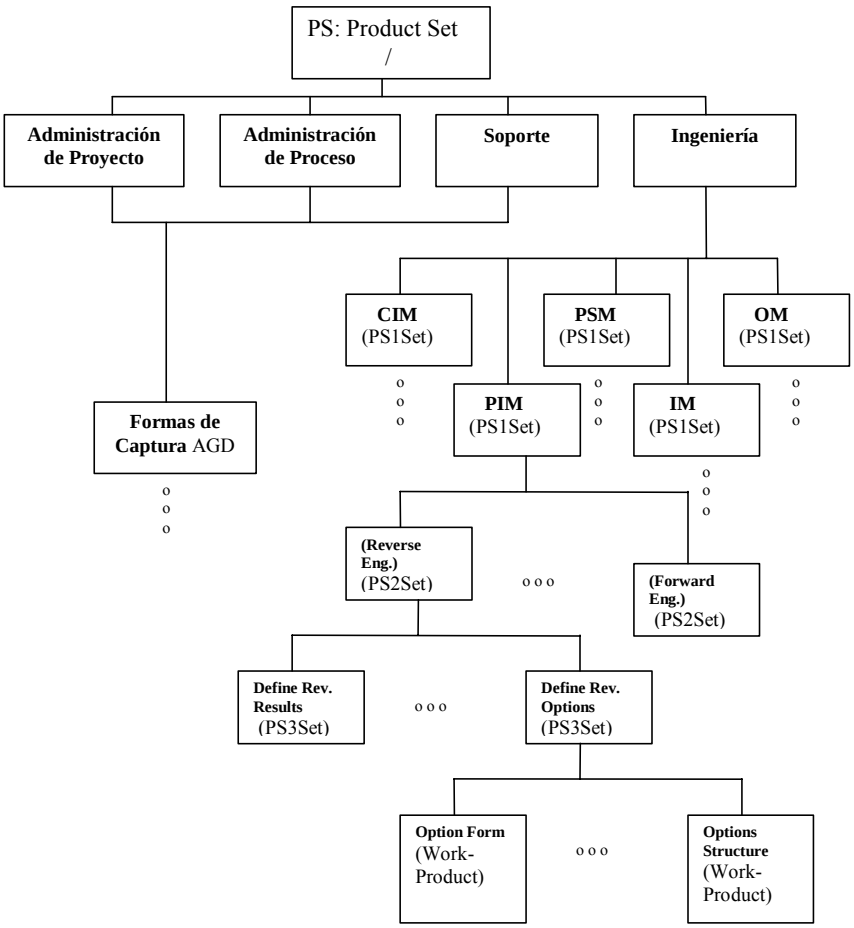


Figura 3-6 *Product Set* con sus cinco niveles jerárquicos.

Las formas de captura utilizadas en el método AGD se enlistan en la tabla 3-1. Aquellas corresponden mayormente a las formas utilizadas en el Proceso de Software en Equipo / Team Software Process (TSP) [H99], modificadas para incluir la estructura y conceptos del método AGD.

La tabla 3-1 completa la presentación del PS, aunque el énfasis de este capítulo se centra en los productos ubicados en la categoría Ingeniería. Cara renglón de la tabla incluyen informa-

ción de las formas de captura: en la primera columna encontrará el evento o eventos asociados a su utilización, en la segunda columna se ubica el nombre descriptivo de la forma, en la tercera columna se coloca el nombre corto con el que se identifican, y en la cuarta columna se describe la función que tiene cada forma.

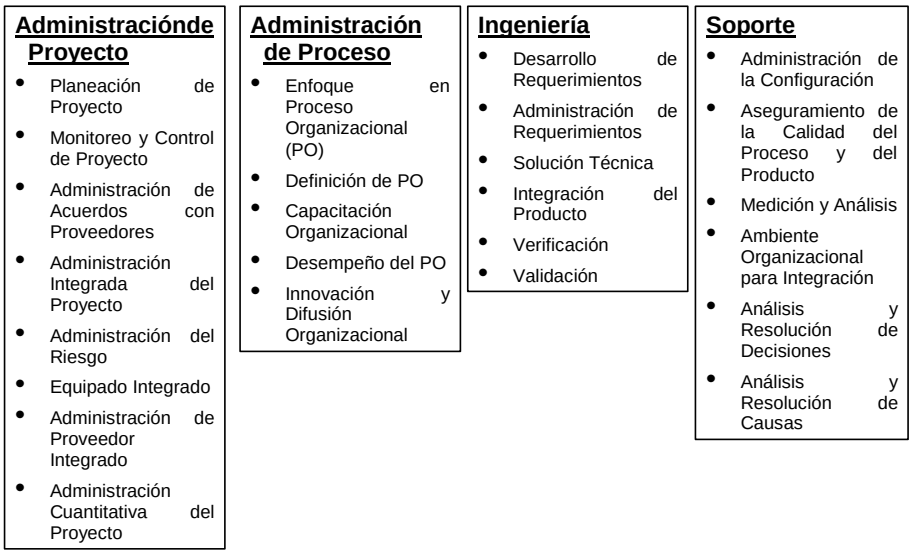


Figura 3-7 Áreas de proceso de las categorías del CMMI, incluidas en el primer nivel del PS.

Tabla 3-1 Formas de Captura Principales del método AGD

Evento asociado	Forma	Abrev.	Uso
Sin evento asociado (Uso continuo**)**	Bitácora: Registro del tiempo	MDP-LOGT	Medición del tiempo trabajado
	Bitácora: Registro de defectos	MDP-LOGD	Localización y corrección de defectos
	Bitácora de pruebas	MDP-LOGTEST	Registro de defectos encontrados y corregidos, en pruebas
Semana Inic. - Fin	Reporte de estado semanal	MDP-WEEK	Reporte de estado semanal
	Forma para programar esfuerzo	MDP-SCHEDULE	Registro de horas planificadas y reales ocupadas a la semana
	Reporte de estado de configuración	MDP-CSR	Estado concerniente al sistema de administración de configuración
	Bitácora seguimiento de	MDP-ITL	Registro de los riesgos y

	asuntos/riesgos		asuntos
	Entrega de Producto y Retroalimentación del Cliente	MDP-PDF	Medición tiempos y recopilar comentarios del cliente
Ciclo Inic. - Fin	Forma de Estrategia	MDP-STRAT	Presentar planificación
	Evaluación de Equipo y Compañeros	MDP-PEER	Productores y Observadores: Motivación, detección y solución de conflictos
	Propuesta de mejora de proceso	MDP-PIP	Registrar los problemas del proceso e ideas de mejora
	Forma de planificación de tareas	MDP-TASK	Estimar el tiempo de desarrollo, el valor planeado y la fecha de terminación para cada tarea
	Forma de Información del Participante	MDP-INFO	Describir el interés y experiencia del participante
	Resumen de Defectos Insertados	MDP-SUMDI	Resume defectos insertados en las partes de ensambles
	Resumen de Defectos Eliminados	MDP-SUMDR	Resume datos acerca de defectos eliminados de las partes de los ensambles
	Resumen de Planificación	MDP-SUMP	Mantiene información planificada y real para los ensambles de programas
	Plan de Calidad	MDP-SUMQ	Mantiene los datos de calidad planificados y reales de partes o ensambles
	Resumen de Tamaños	MDP-SUMS	Resume los datos acerca del tamaño de los productos
	Resumen de Tiempos de desarrollo	MDP-SUMT	Resume los datos acerca del tiempo ocupado en el desarrollo de los productos de un ensamble
	Resumen de Tareas	MDP-SUMTASK	Facilitar las actividades de planificación y seguimiento de tareas
Término de	Reporte de revisión -	MDP-INS-	Recolectar datos deta-

Producto / Ciclo	detallado	DET	llados de la inspección/revisión
	Reporte de revisión - resumen	MDP-INS-SUM	Analizar y resumir la inspección/revisión
	Solicitud de Cambio	MDP-CCR	Información acerca del cambio al Comité de Control de Configuración, para realización o rechazo

Las formas en la tabla 3-1 se agruparon según los eventos asociados a su uso: a) uso continuo, b) reporte y control en reunión semanal, c) inicio/final de ciclo y d) término de un producto-del-trabajo.

En las subsecciones siguientes se describe la estructura jerárquica de los cuatro niveles de la categoría Ingeniería y se explican los productos y/o modelos requeridos durante el desarrollo del software.

3.3.3 PS1Sets: Primer Nivel de la Categoría Ingeniería

Los cinco PS1Sets de la categoría Ingeniería son: Modelo Independiente del Cómputo / Computation Independent Model **(CIM)**, Modelo Independiente de la Plataforma / Platform Independent Model **(PIM)**, Modelo de Plataforma Específica / Platform Specific Model (**PSM)**, Modelo de Implementación / Implementation Model **(IM)**, y Modelo de Operación / Operation Model (**OM)**. Cada uno de los *PS1Sets* mencionados agrupa a uno o más *PS2Sets*, éstos a su vez se construyen mediante conjuntos de productos PS3Sets, los cuales producen uno o varios productos-del-trabajo / work_products (WP).

En la figura 3-8 se muestran los cinco PS1Sets y los veinte PS2Sets que contiene la categoría de Ingeniería en PS del método AGD. La estructura jerárquica del PS, refina la información de alguna de las dimensiones del modelo de Trabajo Cooperativo: Almacenamiento, Recursos y Lógica / Cooperative Work: Storage, Resource y Logic (CWSLR) [GJM99]. Se estableció explícitamente que al menos hubiera un PS3Set que correspondiera a cada uno de los 15 niveles del modelo CWSLR (nueve estructurales y seis funcionales, como se explica en el anexo B).

Como se mencionó cuando se explicó el método AGD en forma general, en la sección 3.2, cada producto requiere un número diferente de tareas asociadas a un sub-proceso específico del modo de trabajo en equipo. Al sub-proceso donde se dedica el mayor esfuerzo a cada producto lo denominamos su *sub-proceso principal*.

La relación entre cada conjunto de productos y su sub-proceso principal se muestra en la figura 3-8. Así por ejemplo, el sub-proceso, que esta relacionado principalmente con PS2Set “4 (Identificación de Componentes)”, es el subproceso Requerimientos / Requirements.

Sub Procesos	Conjunto de Productos (PS)					
		CIM	**PIM**	**PSM**	**IM**	**OM**
	LANZAMIENTO & ESTRATEGIA	**1 (CWSP Inicialización)**				
	PLANIFICACIÓN	**2** (P**lanificación**) Alcance/Prioridad (2)		Estructura de División del Trabajo (12) Asignación de Unidad (12)		
	REQUERIMIENTOS	**3** (**Negocio Existente**) **4** (**Identificación de Componentes**) **6 (Requerimientos de Sistemas)**	**7** (**Ing. Inversa**)	**10** (**Requerimientos de Software)** **11** (**Restricciones en Requerimientos**)		**17** (**Identificación de Recursos**)
	ANALISIS & DISEÑO		**8** (**Ing. Directa**)	**12** (**Arquitectura del Componente**)	**14 (Diseño-Detallado**)	
	IMPLEMENTACIÓN	**5 (Prototipo de Sistemas)**	**9 (Prototipo de Software)**		**15** (**Codificación**) **16** (**Operación Inicial**)	Aceptación (16) **18** (**Distribución**)
	PRUEBA			Especificación de Aceptación (10) Especificación de Integración (12)	Especificación de Unidad (14) Unidad (15) Integración (16)	**19** (**Entrega del Producto**)
	POST-MORTEM	Retroalimentación del Cliente (5)	Retroalimentación del Cliente (9)	**13** (**Análisis de Arquitectura**)	Retroalimentación del Cliente (16)	**20 (Evaluación**)

CIM – Modelo Independiente de Cómputo
PIM – Modelo Independiente de Plataforma
PSM – Modelo De Plataforma Específica
IM - Modelo de Implementación
OM - Modelo de Operación

Figura 3-8 Categoría Ingeniería, del *Product Set,* completa con sus 20 PS2Sets, y los sub-procesos principales asociados.

3.3.4 PS2Sets: Segundo Nivel de la Categoría Ingeniería

Los PS1Setc y PS2Sets de la categoría de Ingeniería del PS se muestran en las tablas 3-2 y 3-3. Estas tablas incluyen, en la primera columna, los cinco *PS1Sets* del método con sus respectivos *PS2Sets* y en la segunda columna los tipos principales de modelos generados. En subsección 3.3.5 se muestra una tabla para cada PS2Set con el detalle de los PS3Sets que contiene y los WPs incluidos en cada PS3Set.

La primera columna de las tablas 3-2 y 3-3 muestra la hiperliga a seguir, para ver un ejemplo del producto, cuando se selecciona la cadena de opciones: **Configura > Manejo de Productos del trabajo > Despliegue /Impresión > PS1 > Product Set**, en la interfase del prototipo del SiSoProS incluido en el CD del anexo C. Se incluye también en esta columna, en renglones sin numeración, las características del producto que se asocian con la terminación de cada SP2Set.

Las características que se asocian a cada PS2Set son: administrado / **managed**, esencial / **essential**, factible / **feasible**, funcional / **functional**, trabajable / **workable**, correcto / **correct**, operacional / **operational**, aceptado / **accepted**, terminado / **finished**). La asociación explícita de estas cualidades con la construcción de los PS2Sets se hace para dar contexto al esfuerzo de desarrollo, de esta manera se resaltan los atributos que se deben alcanzar al terminar cada WP.

Tabla 3-2 Resumen del *PS* (continúa en la tabla 3-3)

PS1Set-(PS2Set)	ProductoDelTrabajo /Work Products
Administrado / Managed	
1 CIM-(CWSP Inicialización) / CIM-(CWSP Initialization)	Formas Web, Cuaderno ISW
Esencial / Essential, Factible / Feasible	
2 CIM-(Planificación) / CIM-(Planning)	Carpeta de Planeación (factibilidad), Entidades, Jerárquico, Casa de Calidad, Formas Web
Funcional / Functional	
3 CIM-(Negocio Existente) / CIM-(Existing business)	Entidades, Formas Web
4 CIM-(Identificación de Componentes) / CIM-(Component Identification)	Requerimientos de Sistema, Entidades (M), Línea de Producción , Distribución Lógica de Datos, Estructura Warnier/Orr , Entrada/Resultado ejemplo, Diccionario de Datos, Carpeta de Viabilidad, Resultados Requeridos del Sistema, Carpeta de Formas Web
5 CIM- (Prototipo del Sistema) / CIM- (System Prototype)	Formas Web
6 CIM- (Requerimientos del Sistema) / CIM- (System Requirements)	Conjunto de *result*s previos
Trabajable / Workable	

7 PIM-(Ing. Inversa) / PIM-(Reverse Eng.)	Casos de Uso, Clases , Modelo-de-Objetos Ideal y Real, Cartas de estado, Diagramas de Colaboración, Estructura Warnier/Orr, Entidades (M), Formas Web
8 PIM-(Ing. Directa) / PIM-(Forward Eng)	Casos de Uso, Clases, Modelo-de-Objetos Ideal y Real, Cartas de estado y Colaboración, Estructura Warnier/Orr, Distribución Lógica de Datos, Opciones ejemplo, Diccionario de Datos, Entidad-Relación, Entidades (M), Formas Web
9 PIM- (Prototipo del software) / PIM-(Software Prototype)	Formas Web
Correcto / Correct	
10 PIM- (Requerimientos del Software) / PIM- (Software Requirements)	Conjunto de *result*s previos

Las tablas 3-2 y 3-3 incluyen en la segunda columna, además de los trece tipos de diagrama del lenguaje UML 2.0 (Clases, Objetos, Componente, Distribución, Casos de Uso, Secuencia, Colaboración, Cartas de Estado, Actividad, Paquete, Estructura compuesta, Visión general de interacción y Coordinació) otros tipos de WP visuales adaptados para obtener un Modelo de Proceso del Negocio.

Los tipos de WP adaptados son: Diagramas de Entidades, Diagramas de Warnier/Orr, Formas para Texto, Diagramas de Línea de Producción [O81, BKOMGM85], Diagramas Jerárquicos y Diagramas de Casa de Calidad [B91]. Complementando además, el diseño general del componente, con el diagrama Ciclo de Negocio de Arquitectura / Architecture Business Cycle (ABC) [BCK98, MKKA03].

Para comprender las tablas 3-2 y 3-3, damos un ejemplo. En el caso del renglón numerado con el dígito 4, identificamos en la primera columna el PS1Set **CIM**, y el PS2Set (Identificación de Componentes / Component Identification), que usa (según se muestra en la segunda columna) los tipos principales de WP: "Requerimientos de Sistema, Entidades (M), Línea de Producción, Distribución Lógica de Datos, Estructura Warnier/Orr, Ejemplo de Entrada/Resultado y Diccionario de Datos".

Tabla 3-3 Resumen del *PS* (Viene de la tabla 3-2)

PS1Set-(PS2Set)	ProductoDelTrabajo /Work Products
11 PSM-(Restricciones en Requerimientos) / PSM-(Requirement Restrictions)	Carpeta de Requerimientos de Software, Entidades (M), Estructura Warnier/Orr (M), Entidad-Relación (M), Casos de Uso (M), Clases, Objetos, Componentes, Documento estructurado de prototipo, Resultados Requeridos de Componentes, Formas Web

12 PSM-(Arquitectura del Componente) / PSM-(Component architecture)	Arquitectura de software, Especificación prueba integración, Entidad-Relación, Ciclo de Negocio para Arquitectura (ABC), Jerárquico, Entidades (M), Línea de producción (M), Casos de Uso (M), Clases (M), Objetos (M), Componentes (M), Resultados Requeridos del Componentes (M), Statecharts (M) y Colaboración (M), Formas Web
13 PSM-(Análisis de la Arquitectura) / PSM-(Architecture analysis)	Escenarios, Entidades (M), Línea de Producción (M), Casos de Uso (M), Clases (M), Objetos (M), Componentes (M), Cartas de estado (M), y Colaboración (M), Carpeta de Diseño, Formas Web
Operacional / Operational	
14 IM-(Diseño-Detallado) / IM-(Detailed-Design)	Definición de procesos, Barras de Gant, Carpeta de Desarrollo de Unidad (UDF), Resultados Requeridos de Unidad, Diccionario de datos (M), Entidad-Relación (M), Formas Web
15 IM-(Codificatción) / IM-(Codification)	Archs. Fuente, Archs. Objeto, Archs. Ejecutable, UDF (M), Resultados Requeridos de Unidad (M), Formas Web
16 IM-(Operación Inicial) / IM-(Initial Operation)	Archivos de Construcción, Documentación del usuario, Resultados prueba de integración, UDF (M), Archs. Fuente (M), Archs. Objeto (M), Archs. Ejecutable (M), Componentes (M), Resultados requeridos de Componente (M), Carpeta de Construcción, Formas Web
Aceptado / Accepted	
17 OM-(Identificación de Recursos) / IM-(Initial Operation)	Plano de red, Mapa del campus, Plano del edificio, Plano del cubículo, Formas Web
18 OM-(Distribución) / OM-(Distribution)	Distribución, Línea de producción (M), Plano del edificio (M), Plano de la red (M), Componentes (M), Formas Web
19 OM-(Entrega del Producto) / OM-(Product Release)	Prueba aceptación, Archs. Fuente (M), Archs. Objeto (M), Archs. Ejecutable (M), Componentes (M), Resultados Requeridos del Sistema (M), Resultados Requeridos de Componentes (M), Documentación de Usuario (M), Formas Web
Terminado / Finished	
20 OM-(Evaluación) / OM-(Assessment)	Reporte de Evaluación Proceso, Formas Web, Gráficas Estadísticas

Nótese que en esta lista aparece el WP "Entidades (M)", donde la (M) indica que este WP se generó previamente, y que en el caso de este PS1Set-*(PS2Set)* se puede **M**odificar.

Los demás WPs, que no incluyen (M), se producen por primera vez como parte de (Identificación de Componentes / Component Identification).

Se relacionan los PS2Sets de la categoría Ingeniería con el modelo CWSLR, ayudando a construir *PS3Sets* que añadan incrementalmente la información detallada que se ubican en cada uno de los niveles del modelo CWSLR. Cuando se termina, de identificar los modelos para cada elemento del modelo CWSLR, consideramos completa la lista de los componentes del PS.

Además los modelos resultantes (WPs) se ubican en una estructura de directorios, ordenados de acuerdo a las Carpetas de Desarrollo de Unidad / Unit Development Folders (UDF [I87]), donde se organizan para su presentación. Este ordenamiento facilita la consulta, pues utiliza nombres comunes en la práctica de ingeniería de software.

El detalle para cada renglón de las tablas 3-2 y 3-3 se presenta por separado en una tabla, similar a la tabla 3-4. Dado que se presentan veinte renglones numerados, se incluyen veinte tablas en § 3.3.5. En el prototipo, incluido en el disco compacto del anexo C, se presenta un caso de estudio mediante los WPs que lo describen, donde se utiliza el método AGD.

3.3.5 PS3Sets: Tercer Nivel de la Categoría Ingeniería

En esta sección mostramos la secuencia detallada de PS3Sets que refina la información contenida en las tablas 3-2 y 3-3, para sus renglones numerados: PS2sets, PS3Sets y ProductosDelTrabajo / Work-Products. Incluyéndose:

- PS3Sets del PS1Set: CIM,
- PS3Sets del PS1Set: PIM,
- PS3Sets del PS1Set: PSM,
- PS3Sets del PS1Set: IM y
- PS3Sets del PS1Set: OM.

La estructura de la tabla 3-4 se explica como ejemplo de las 20 tablas que detallan los PS3Sets correspondientes a los renglones de las tablas 3-2 y 3-3.

La tabla 3-4 está asociada al primer renglón de la tabla 3-2. En la tabla 3-4 la columna uno incluye el PS2Set asociado, en este caso **CIM-(CWSP Inicialización) / CIM-(CWSP Initialization).** La columna dos incluye el nombre de los tres PS3Sets (ej. Identificación del Proyecto / Project Identification), que constituyen el PS2Set indicado. La columna tres inserta el tipo de producto-del-trabajo (ej. Forma Web), asociado a cada PS3Set. La marca "x" en la columna cuatro indica cuando el tipo de WP es susceptible de obtenerse con soporte automatizado utilizando el Mecanismo Dinámico-Gráfico de captura y despliegue (DGM) [GM02]. Por último en la columna cinco se incluye la dimensión / *dimension* y nivel / *level* del modelo CWSLR, donde se ubica el WP construido. (ej.Lógica: Organización / Logic: Organization).

En las sub-secciones siguientes comentamos las características que consideramos importantes para cada tabla. En general en la columna "ProductosDelTrabajo / *Work-product*s" se presentan los modelos producidos cuando realizamos la actividad correspondiente. Se marcan con una M encerrada entre paréntesis (M) los modelos que habiéndose generado anteriormente se pueden modificar.

3.3.6 PS3Sets del PS1Set: CIM

La información de los renglones numerados del 1 al 6 de la tabla 3-2 se muestra desglosada en las tablas 3-4 a 3-9. Incluyendo a los modelos incluidos en CIM-(CWSP Inicialización) / CIM-(CWSP Initialization), CIM-(Planificación) / CIM-(Planning), CIM-(Negocio Existente) / CIM-(Existing business), CIM-(Identificación de Componentes) / CIM-(Component identification), CIM- (Prototipo del Sistema) / CIM- (System Prototype) y CIM–(Requerimientos del Sistema) / CIM–(System Requirements).

La Tabla 3-4 presenta el *PS2Set* CIM-(CWSP Inicialización) / CIM-(CWSP Initialization) ubicada, de acuerdo a la quinta columna, en la "Dimensión: Nivel" del modelo CWSRL "Logic: Organization". Este PS2Set no incluye actividades de desarrollo que construyan modelos, pero es indispensable para iniciar el trabajo en un proyecto. El desarrollador proporciona los datos de identificación del proyecto a iniciar y establece si existen relaciones de dependencia con otros proyectos.

Una jerarquía de proyectos está definida hacia arriba (de quien depende este proyecto), hacia abajo (proyectos dependientes) o en el mismo nivel (proyectos correlacionados). Además en el *PS2Set* "CWSP Inicialización" se proporcionan los datos de todos los miembros del equipo de proyecto creándose los cuadernos de ingeniería de software para cada participante. Este cuaderno es el espacio estructurado de trabajo utilizado, durante el desarrollo, por cada miembro del equipo.

Tabla 3-4 *PS2Set* CWSP Inicialización del PS1Set CIM, definiendo el proyecto en el ambiente CWSP

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
1 CIM-(CWSP Inicialización) / CIM-(CWSP Initialization)	Project Identification (Identificación del Proyecto)	Forma Web		Logic: Organization
	Dependencies With Other Projects (Dependencias con Otros Proyectos)	Forma Web		Logic: Organization
	Project Team Definition	Forma Web		Logic: Organization

	(Determinación del Equipo de Proyecto)	Cuaderno de Ingeniero SW, con Bitácora de Tiempos (para cada miembro del equipo)		

Los resultados necesarios para identificar problemas u oportunidades dentro de la organización, se ubican en relación con la estrategia y características del negocio y se presentan en la tabla 3-5 con información acerca del ***PS2Set*** **(Planificación-Visión)** incluido en el **PS1Set CIM**. Se respeta la voz del cliente (sin cambiar su terminología) y determinando el alcance (la funcionalidad y áreas organizacionales incluidas) así como las prioridades para las áreas usuarias y clientes. Toda la información recopilada se estructura en una Carpeta de Planificación donde se indica la factibilidad del proyecto.

Tabla 3-5 *PS2Set* Planificación-Visión del *PS1Set* CIM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
2 CIM-(Planificación-Visión) / CIM-(Planning-Vision)	Understand Strategy (Entender Estrategia)	Forma Web		Logic: Organization
	Understand Business (Entender Negocio)	Colaboración (Entidades: negocio), Organigrama, Forma Web, Casa de Calidad	x	Logic: Organization
	Problems/Opportunities (Problemas/Oportunidades)	Forma Web multimedia		Logic: Component
	Get customer demands (Obtener Solicitudes del Cliente)	Colaboración (Entidades: cliente), Casa de Calidad, Forma Web	x	Logic: Organization
	Benchmarking New company (Comparación Cuantitativa de Compañía Nueva)	Forma Web, Tablas comparativas		Logic: Organization
	Users/areas (Äreas de Usuarios)	Colaboración (Entidades: usuarios), Jerárquico, forma	x	Logic: Component

		Web		
	Scope/Priority (Alcance/Prioridades)	Forma Web, Colaboración (Entidades: aplicación), Casos de Uso, Casa de Calidad, Carpeta de Planeación (visión, factibilidad),	x	Logic: Component

La tabla 3-6, despliega los elementos del *PS1Set-(PS2Set)* **CIM-(Negocio Existente)** donde se obtienen imágenes estáticas de la percepción de los usuarios acerca del proceso o su forma de trabajo por medio de diagramas de entidades. Se utilizan técnicas muy sencillas (ej. Diagramas de Entidad, Diagramas de Línea de Producción) propuestas originalmente por Ken Orr, dentro del marco de los métodos estructurados, para obtener una visión abstracta (sin detalles), pero con los elementos suficientes para conceptuar la forma de trabajo. Los resultados obtenidos para este PS2Set, son versiones modificadas de los modelos originalmente propuestos por Orr, dando Orientación a Objetos (OO) a las técnicas mencionadas.

Los diagramas de entidades propuestos por Ken Orr, permiten no tener que recurrir a un experto en el dominio de la aplicación. Dichos diagramas facilitan realizar una primera propuesta de los Casos de Uso y el modelo de Clases, facilitando la interacción con los especialistas en el dominio de aplicación para recabar su opinión, en lugar de que ellos indiquen que y como hacerlo.

Actualmente, la mayoría de los métodos con orientación a objetos depende de las habilidades y experiencias previas, de los especialistas en el dominio de la aplicación, para obtener la primera versión de los Casos de Uso y el Modelo de Clases. La integración de las técnicas,estructuradas de Ken Orr, a nuestro método, es una aportación en el ámbito de los métodos OO, dado que permiten representar la forma en curso de realizar los procesos.

En la columna DGM, los renglones incluyendo una "x" marcan el uso del Mecanismo Dinámico Gráfico [Gm02] para obtener los modelos. El uso del mecanismo DGM facilita la transformación de un modelo al siguiente en la secuencia mediante un procedimiento interactivo.

Tabla 3-6 *PS2Set* Negocio Existente del *PS1Set* CIM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
3 CIM-(Negocio Existente) / CIM-(Existing business)	User-Level-Entity-Diagram (Diagrama de Entidades Nivel-usuario)	Colaboración (Entidades: usuario), Forma Web con relaciones y sus atributos	x	Logic: Organization
	Combined user-level entity	Colaboración (Entidades:	x	Logic: Organization

	diagram (Diagrama de Entidades Combinado Nivel-usuari)	combinado), Forma Web con relaciones y sus atributos		
	Application-Level Entity Diagram (Diagrama de Entidades Nivel-aplicación)	Colaboración (Entidades: aplicación), Forma Web definiendo secuencias de relaciones	x	Logic: Organization

En las tablas 3-7 y 3-8 se presenta el *PS1Set-PS2Set* **CIM-(Identificación de Componentes)** en el que se recopila información, de los resultados obtenidos en el *PS2Set* anterior. A partir del "Diagrama de Entidades a Nivel-aplicación", se identifican con ayuda de un proceso muy sencillo, los principales resultados, entradas y opciones del sistema. Pretendiendo identificar el mínimo de estos elementos, pues si se elimina alguno, el componente o sistema resultante no sería útil.

Tabla 3-7 *PS2Set* Identificación de Componentes (menú a 1º nivel) del *PS1Set* CIM (Continúa en tabla 3-8)

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
4 CIM (Identificación de Componentes) / CIM (Component Identification)	Principal results compilation (Compilación de Resultados Principales)	Colaboración (Entidades: aplicación) (M), Forma Web definiendo atributos	x	Storage: Basis
	Principal inputs compilation (Compilación de Entradas Principales)	Colaboración (Entidades: aplicación) (M), Forma Web definiendo atributos	x	Storage: Basis
	Results and inputs dependencies (Dependencias de Resultados y Entradas)	Línea de producción, Forma Web	x	Storage: Process
	Principal options compila-	Página Web e hiperligas		Storage: Basis

	tion (Compilación de Opciones Principales)			
	Options' dependencies (Dependencias de Opciones)	Página Web e hiperligas con claves jerárquicas, Mapa de Navegación		Storage: Process
	Objectives (Objetivos)	Colaboración (Entidades: aplicación) (M), Forma Web	x	Logic: Component Architecture
	Functional Scope (Alcance Funcional)	Línea de producción (M), Página Web con claves jerárquicas (M), Mapa de Navegación (M)		Logic: Component Architecture
	Define Results (Definición de Resultados)	Ver tabla 3-8		
	Define Inputs (Definición de Entradas)	Ver tabla 3-8		
	Define Options (Definición deOpciones)	Ver tabla 3-8		
	Define System Data Dictionary (Definición del Diccionario de Datos del Sistema)	Entidad-Relación de objetos de datos, Objetos de datos, Forma Web Definición General de ODs		Storage: Basis

A partir de la identificación de los principales resultados y entradas se sintetizan directamente los objetivos de la aplicación. El objetivo es recibir todas las entradas y generar todas las salidas oportunamente.

Si bien es muy importante identificar los resultados, entradas y opciones principales, también la definición detallada de cada uno de esos elementos es indispensable. Proporcionando, los work-products incluidos en la tabla 3-8, obtenemos la información necesaria para determinar los datos y las estructuras de datos requeridas por el componente de software.

Tabla 3-8 *PS2Set* Identificación de Componentes (menús a 2º nivel) del *PS1Set* CIM (Viene de tabla 3-7)

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
	Define Results			
	Output form (Formato de Resultados)	Distribución Lógica de Datos, Forma Web		Storage: Basis
	Output content (Contenido del Resultados)	Ejemplo (resultado), Forma Web		Storage: Basis
	Output structure (Estructura del Resultados)	Estructura Warnier/Orr, Forma Web		Storage: Basis
	Output data elements (Elementos de datos del Resultados)	Objetos de datos, Forma Web Definición General de ODs		Storage: Basis
	Outputs organizational cycle (Ciclos Organizacional de Resultados)	Página Web con tabla de frecuencias		Storage: Process
	Define Inputs (Define Entradas)			
	Input form (Formato de Entradas)	Distribución Lógica de Datos, Forma Web		Storage: Basis
	Input content (Contenido de Entradas)	Ejemplo (resultado), Forma Web		Storage: Basis
	Input structure (Estructura de Entradas)	Estructura Warnier/Orr, Forma Web		Storage: Basis
	Input data elements (Datos de Entradas)	Objetos de datos, Forma Web Definición General de ODs		Storage: Basis
	Input organizational cycles (Ciclos Organizacionales de Entradas)	Página Web con tabla de frecuencias		Storage: Process

	Define Options (Define Opciones)			
	Option Form (Formato de Opciones)	Distribución Lógica de Datos, Forma Web		Storage: Basis
	Option Content (Contenido de Opciones)	Ejemplo (resultado), Forma Web		Storage: Basis
	Options Structure (Estructura de Opciones)	Estructura Warnier/Orr, Forma Web		Storage: Basis
	Data Elements at HMI (Elementos de Datos en Interfase Hombre Máquina)	Objetos de datos, Forma Web Definición General de ODs		Storage: Basis

En la tabla 3-9 se presenta el *PS1Set-(PS2Set)* **CIM-(Prototipo del Sistema)**. El prototipo se construye utilizando una página Web con hiper-ligas cuyo texto son los nombres de las opciones definidas y como página Web asociada: una instancia ejemplo del contenido del resultado a enviar o de la entrada a recibir, según sea el caso.

Tabla 3-9 *PS2Set* Prototipo del Sistema de la *PS1Set* CIM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
5 CIM (Prototipo del Sistemsa) / (System Prototype Integration)	System Prototype Integration (Integración del prototipo del sistema)	Página Web con hiperligas, Ambiente visual		Logic: Component

En la figura 3-9 vemos la ubicación, de los *Work-Product(s)* para obtener la Carpeta de Requerimientos del Sistema, localizados mayormente en el sub-proceso **Requirements. I**ndicando preponderantemente la realización de operaciones de análisis.

Describimos a continuación la distribución de los *resultado*s:

- Primero, en el sub-proceso de Lanzamiento se inicia la obtención de la Carpeta de Requerimientos del Sistema, estableciendo el nombre del sistema a desarrollar, los responsables, el proceso a seguir, el contenido deseado y los recursos necesarios para realizarla

- Segundo, en el sub-proceso de Requerimientos analizamos el negocio existente: tomando instantáneas de cada usuario del sistema y de las relaciones con otros usuarios, organizaciones o sistemas. Las instantáneas o retratos individuales se combinan en un sólo diagrama y se delimita el alcance resultando en un diagrama de la aplicación

	SubProceso(s)						
	LANZA-MIENTO & ESTRATEGIA	PLANI-FICA-CIÓN	REQUERIMIENTOS	ANÁLISIS & DISEÑO	IMPLEMEN-TACIÓN	PRUEBA	POST-MOR-TEM
P S	6 CIM Requeri-mientos del Sistema		3 CIM Negocio Existente Diagrama de Entidades Nivel-Aplicación Diagrama de Entidades Combinado a Nivel-Usuario Diagrama de Entidades a Nivel-Aplicación 4 CIM Identificación de Componentes Compilación de resultados principales Compilación de entradas principales Dependencias de resultados a entradas Compilación de opciones principales Dependencias entre opciones	Objetivos Alcance funcional Definición de Resultados Definición de Entradas Definición de Opciones definición del Diccionario de Datos del Sistema	5 CIM Prototipo del Sistema		

Figura 3-9 *ProductosDelTrabajo*, incluidos en la Carpeta de Requerimientos del Sistema, mediante sub-procesos específicos

- Tercero, se analiza el proceso deseado identifican los componentes (elementos): enlistando los resultados principales, enlistando las entradas principales, formando líneas de producción con las entradas y salidas, enlistando las opciones necesarias y jerarquizándolas
- Cuarto, se sintetizan componentes (elementos): listando los objetivos, definiendo el alcance funcional, diseñando los resultados, entradas y opciones, identificando los datos involucrados
- Quinto, se construye un prototipo del sistema considerando la funcionalidad determinada. Utilizando hipertexto y navegadores Web

3.3.7 PS3Sets del PS1Set: PIM

La información para los renglones numerados del 7 al 9 de la tabla 3-2 se muestra en tablas 3-10, 3-11 y 3-12 a mayor detalle, abarcando el **PIM-**(**Ing. Inversa**) / **PIM-**(**Reverse Eng.**), el **PIM-**(**Ing. Directa**) / **PIM-**(**Forward Eng**) y el **PIM-(Prototipo del Software) / PIM-(Software Prototype)**.

La tabla 3-10 presenta el *PS1Set-(PS2Set)* **Ingeniería Inversa** con que iniciamos el ***PS1Set* PIM.** Comenzando un análisis de la forma en que se lleva el proceso actual, cuando existe uno o más sistemas computacionales que soporta las operaciones.

Para construir los modelos incluidos en la tabla 3-10 se identifican, a partir de los diagramas de entidades, los casos de uso y el diagrama de clases, así como los modelos de objetos. Se incluyen los casos en que no se pueden realizar las tareas de los procesos y las acciones a realizar para dar solución a estos casos de excepción. Haciéndose también evidentes las tareas de control, planeación, monitoreo y reportes ad-hoc dentro del proceso soportado por computadora para la aplicación.

Tabla 3-10 *PS2Set* Ing. Inversa. del *PS1Set* PIM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
7 PIM-(Reverse Eng.) / PIM-(Ing. Inversa)	Define Rev. Results (Resultados en Reverse Eng.)	Igual a tabla 3-8		
	Define Rev. Inputs (Entradas en Reverse Eng.)	Igual a tabla 3-8		
	Rev. Functional Flow and Persistence	Entidades, página Web		

	(Reverse Eng. Flujo Funcional y Persistencia)			
	Define Rev. Options (Opciones en Reverse Eng.)	Igual a tabla 3-8		
	Data Dictionary (Diccionario de Datos)	Entidad-Relación de objetos de datos, Objetos de datos, Forma Web Definición General de Ods		
	Use Cases (Casos de Uso)			
	Functional Processes (Procesos Funcionales)	Modelo de Clases Ideal o de análisis, Página Web	x	Logic: Organization
	Control Feedback (Control de Retroalimentación)	Modelo de Clases Real o de diseño, Forma Web	x	Logic: Organization
	Tasks And Procedures (Tareas y Procedimientos)	Statechart y Colaboración, Forma Web	x	Logic: Organization
	Cooperation (Cooperación)	Statechart y Colaboración, Forma Web	x	Logic: Cooperation
	Management Control (Control Administrativo)	Statechart y Colaboración, Pagina Web	x	Logic: Cooperation
	Organization Memory (Memoria Organizacional)	Página Web		Storage: Memory
	Cycles System Diagram (Diagrama de Ciclos del	Estructura Warnier/Orr, Página Web con tabla		Logic: Organization

	Sistema)	de frecuencias		

En la tabla 3-11 se presenta el *PS1Set-(PS2Set)* **PIM-(Ingeniería Directa)** en el que se analiza y sintetiza la forma en que se llevará el proceso propuesto soportado por sistemas computacionales. Los casos de uso y el diagrama de clases, así como los modelos de objetos se identifican, a partir de los resultados del *PS2Set* anterior y de los diagramas de entidades. Incluyendo los casos en que no se pueden realizar las tareas de los procesos y las acciones a realizar para solucionar estos casos de excepción.

Tabla 3-11 *PS2Set* Ing. Directa del *PS1Set* PIM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
8 PIM-(Ing. Directa) / PIM-(Forward Eng.)	Data Dictionary (Diccionario de Datos)	Entidad-Relación de objetos de datos, Objetos de datos, Forma Web Definición General de Ods, Def. datos elementales, Def. estructuras de datos		
	Functional Flow and ersistence (Flujo funcional y persistencia)	Entidades (M), Forma Web definiendo objetos de datos persistentes, Resultados adicionales por ODS persistentes	x	Logic: Organization
	Software Results (Resultados Software)	Igual a tabla 3-8		
	Software Inputs (Entradas Software)	Igual a tabla 3-8		
	Software Options (Opciones Software)	Igual a tabla 3-8		
	Scope with SW (Alcance incluyendo Software)	Línea de producción, lista jerárquica de opciones		

	Use Cases (Casos de Uso)	Casos de uso, Forma Web especificación de caso de uso		
	Functional Processes (Procesos Funcionales)	Modelo de Clases Ideal o de análisis, Página Web	x	Logic: Organization
	Control Feedback (Control de Retroalimentación)	Modelo de Clases Real o de diseño, Forma Web	x	Logic: Organization
	Processes (Procesos)	Statechart y Colaboración, Forma Web	x	Logic: Organization
	Cooperation (Cooperación)	Statechart y Colaboración, Forma Web	x	Logic: Cooperation
	Presentation (HCI) (Presentación IHC)	Mapa, Contextos y Clases navegacionales, Página Web		
	Management Control (Control Administrativo)	Statechart y Colaboración, Pagina Web	x	Logic: Cooperation
	Organization Memory (Memoria Organizacional)	Página Web		Storage: Memory
	System Cycles Diagram (Diagrama de Ciclos del Sistema)	Estructura Warnier/Orr, Página Web con tabla de frecuencias		Logic: Organization

Se hacen también evidentes las tareas de control, planeación, monitoreo y reportes ad-hoc dentro del proceso que se propone para el área de aplicación. Este PS2Set define la estructura de las opciones soportadas por computador que se darán al usuario. Identificando para la interfase de usuario su distribución y estructura lógica.

En la tabla 3-12 se presenta el *PS1Set-PS2Set* **PIM: Prototipo de Software**. El prototipo se obtiene utilizando una página HTML con hiper-ligas que tienen como texto los nombres de las opciones definidas y como página Web asociada: la definición del contenido del resultado que se envía o de la entrada que se recibe, según sea el caso.

Tabla 3-12 *PS2Set* Prototipo de Software del *PS1Set* PIM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
9 PIM: (Prototipo del Software) / PIM: (Software Prototype)	Software Prototype Integration (Integración del prototipo del sistema)	Página Web con hiper-ligas, Ambiente visual		Logic: Component

3.3.8 PS3Sets del PS1Set: PSM

La información para los renglones numerados del 10 al 13 de las tablas 3-2 y 3-3 se muestra en las tablas 3-13 a 3-16 con mayor detalle. Incluyendo a los modelos incluidos en **PIM-(Requerimientos del Software) / PIM-(Software Requirements), PSM-(Restricciones en Requerimientos) / PSM-(Requirement Restrictions), PSM-(Arquitectura del Componente) / PSM-(Component Architecture)** y **PSM-(Análisis de la Arquitectura) / PSM-(Architecture Analysis)**.

En la tabla 3-13 se presenta el *PS1Set-(PS2Set)* de **PIM-(Requerimientos de Software)**. Indicando que el resultado es una Carpeta de Requerimientos de Software que se implementa desplegando el contenido de la carpeta en una página HTML con hiper-ligas hacia los resultados de todas las actividades previas del método AGD. Pudiéndose imprimir la Carpeta de Requerimientos de Software utilizando las opciones: Configura, Manejo de Productos del Trabajo, PS1 y PIM: SW Requirements.

Tabla 3-13 *PS2Set* Requerimientos del Software del *PS1Set* PIM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
10 PIM-(Requerimientos del Software) / PIM: Software Require-	Software Requirements (Requerimiento del Software)	Conjunto de productos-del-trabajo previos		Logic: Component

ments	Acceptance test specification (Especificación de Prueba de Aceptación)	Resultados Requeridos de Componentes, Forma Web		Logic: Component

Este ***PS2Set*** agrupa los resultados de actividades seleccionadas realizados previamente. Su contenido se asemeja al del documento Requerimientos del Sistema, incluyendo además los requerimientos originados en la naturaleza computacional del software. Los *ProductosDelTrabajo* agrupados se muestran en el ejemplo 3-1.

**********Ejemplo 3-1 (Inicia) **********
PIM: Software Requirements

Planificación
(Contenido de CIM: Planificación)

Negocio Existente
(Contenido de CIM: Requerimientos del Sistema)

Identificación de Componentes
(Contenido de CIM: Requerimientos del Sistemas,
agregando los requerimientos no funcionales)

Ingeniería Inversa
(Incluida sólo si existe algún sistema computacional previo)
(Contenido con los mismos nombres que Ingeniería Directa,
detallando al sistema computacional previo)

Flujo Funcional y Persistencia
Inv-Salidas
Inv. forma de salida (distribución lógica de datos)
Inv. contenido de salida (ejemplo o simulación)
Inv. estructura de salida (diagrama Warnier/Orr)
Inv. elementos de datos de salida
Inv. ciclo organizacional de las salidas
Inv. entradas
(Contenido para entradas igual al de Definir Inv-Salidas)
Inv. Opciones (HMI)
(Contenido para opciones igual a los cuatro primeros sub-productos de Definir Inv.-Salidas)
Casos de Uso
Procesos Funcionales (modelo de objetos ideal)
Retroalimentación (modelo de objetos real)
Tareas y Procedimientos (diagramas de Interacción)
Cooperación
Planificación administrativa
Memoria organizacional
Salidas Vs. Frecuencia
Diagrama de ciclos del sistema
Diccionario de Datos

Ingeniería Directa
(Contenido con los mismos nombres que Ingeniería Inversa,
detallando al sistema propuesto)
Diccionario de Datos
Flujo Funcional y Persistencia
Salidas-Software
Forma de salida (distribución lógica de datos)
Contenido de salida (ejemplo o simulación)
Estructura de salida (diagrama Warnier/Orr)
Elementos de datos de salida
Ciclo organizacional de las salidas
Entradas-Software
(Contenido para entradas, igual al de Salidas-Software)
Opciones-Software
(Contenido para opciones, igual a los cuatro primeros sub-productos de Salidas-Software)
Alcance funcional abarcando software
Casos de Uso
Procesos Funcionales (modelo de clases ideal)
Procesos Funcionales con Retroalimentación (modedlo de clases real)
Tareas y Procedimientos (diagramas de iteración)
Cooperación
Presentación
Control administrativo

Planificación administrativa
Memoria Organizacional
Salidas Vs. Frecuencia
Diagrama de ciclos del sistema
Prototipo
Especificación de pruebas de aceptación
**********Ejemplo 3-1 (Termina) **********

En la tabla 3-14 se presenta el *PS1Set-(PS2Set)* de **Restricciones en Requerimientos** en la que el desarrollador indicará las restricciones o condiciones conocidas hasta este momento, ya sea impuestas por el contexto tecnológico, por el cliente/usuario o los miembros del equipo de desarrollo. Principalmente se especifica la forma de distribuir a los elementos del componente en desarrollo. Definiéndose las necesidades o presiones en cuanto a la residencia de los resultados, entradas, interfase de usuario y funcionalidad del componente

.

Tabla 3-14 *PS2Set* Restricciones en Requerimientos del *PS1Set* PSM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
11 PSM-(Restricciones en Requerimientos) / PSM-(Requirements' restrictions)	Result Used By Roll (Resultado Usado por Rol)	Colaboración (Entidades: aplicación) (M), Forma Web	x	Resource: CompNetwork
	Result Produced At Processor And Output Devices (Resultado Producido en Procesador y Dispositivo de Salida)	Colaboración (Entidades: aplicación), Forma Web	x	Resource: CompNetwor
	Input Introduced At (Entrada Capturada en)	Colaboración (Entidades: aplicación), Forma Web	x	Resource: CompNetwor
	Functional Options Processed At (Opciones Funcionales Procesadas en)	Estructura Warnier/Orr (M), Forma Web		Resource: CompNetwor
	Data Stored At (Datos Almacenados en)	Entidad-Relación (M), Forma Web	x	Resource: CompNetwor
	Constraints (Restricciones)	Forma Web		Logic: Component
	Alternative solutions (Soluciones Alternas)	Casos de Uso (M), Clases (M), Componentes, Forma Web	x	Logic: Component
	Risks/Benefits (Riesgos/Beneficios)	Forma Web		Logic: Component

	Recommended solution (Solución Recomendada)	Casos de Uso (M), Clases (M), Componentes (M), Forma Web	x	Logic: Component

En la tabla 3-15 se presenta el *PS1Set-(PS2Set)* de **PSM-(Arquitectura del Componente)** el cual genera la Arquitectura de Software del componente. Es decir, la estructura (o estructuras) del componente, que comprende a los sub-componentes de software, las propiedades visibles de los sub-componentes y las relaciones entre estos.

Se comienza por identificar el Ciclo de Negocio para la Arquitectura (ABC: Architecture Business Cycle): donde se identifica al cliente y usuario final, la organización que desarrolla, los atributos de calidad deseados, el ambiente tecnológico y la experiencia del arquitecto. Con ABC identificado, se establece una especificación de calidad para el componente, jerarquizando los atributos de calidad deseados para el componente. Esto establece un "árbol de calidad para el proyecto / Project Quality Tree".

Se determina el estilo arquitectónico apropiado para el árbol de calidad establecido, y se organizan las estructuras necesarias para describir al componente. Eligiendo alguna combinación de acuerdo al significado de sus relaciones: Utiliza; Llama; Flujo de Datos; Flujo de Control; y/o Clases.

Tabla 3-15 *PS2Set* Arquitectura del Componente del *PS1Set* PS

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
12 PSM (Arquitectura del Componente) / PSM (Component Architecture)	Architecture Business Cycle (Ciclo de Negocio para Arquitectura)	Ciclo de Negocio para Arquitectura (ABC); Forma Web	x	Logic: Component
	Project quality tree (Árbol de Calidad para Proyecto)	Jerárquico	x	Logic: Component
	Component quality speci-	Forma Web		Logic: Component

	fication (Especificación de Calidad del Componente)			
	Quality Vs. Architecture Style (Calidad Vs. Estilos Arquitectónicos)	Forma Web		Logic: Component
	Architecture's Structures (Estructuras de la Arquitectura)	Modules + Conceptual + Process + Physical + Use + Calls + Data flow + Control flow + Classes, Forma Web	x	Logic: Component
	Architecture Folder (Folder de Arquitectura)	Conjunto de *work-products* previos		Logic: Component
	Integration test specification (Especificación de Prueba deIntegración)	Resultados Requeridos de Integración, Forma Web		Logic: Component

En la tabla 3-16 se presenta el PS1Set-(PS2Set) de **PSM-(Análisis de la Arquitectura)** el cual obtiene una evaluación de la arquitectura de software propuesta. La evaluación considera el esfuerzo y los recursos necesarios para su mantenimiento cuando se presenten varios escenarios sugeridos por un árbol de atributos de calidad establecido para el componente.

Se inicia analizando el documento de Diseño General en el que se presenta la arquitectura de software del componente para extraer los estilos arquitectónicos evidentes. A continuación, considerando el árbol de calidad especificado para el componente, se elaboran varios escenarios que plantean modificaciones o adiciones a la funcionalidad del componente.

Con los estilos arquitectónicos y los escenarios definidos se estima el esfuerzo y recursos necesarios para que el componente se adecue a los nuevos requerimientos. Reportándose estos resultados, para su evaluación, haciendo comparaciones con datos históricos de componentes similares.

Tabla 3-16 *PS2Set* Análisis de la Arquitectura del *PS1Set* PSM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
13 PSM- (Architecture analysis) / PSM- (Architecture analysis)	Busyness drivers (Directores del Negocio)	Forma HTML		Logic: Component
	Architecture Styles Extraction and Analysis (Extracción y Análisis de Estilos Arquitectónicos)	Modules + Conceptual + Process + Physical + Use + Calls + Data flow + Control flow + Classes (M), Modelo de Objetos (M), Componentes (M), Statechart (M) y Colaboración (M), Forma Web	x	Logic: Component
	Scenario Generation (Generación de Escenarios)	Página Web con tabla de descripción	x	Logic: Component
	Scenarios Vs. Architecture Styles (Escenarios Vs. Estilos Arquitectónicos)	Forma HTML		Logic: Component
	Architecture analysis report (Reporte del Análisis de Arquitectura)	Carpeta de Diseño, Forma HTML		Logic: Component

3.3.9 PS3Sets del PS1Set: IM

La información de los renglones numerados del 14 al 16 de las tablas 3-2 y 3-3 se extienden en las tablas 3-17 a 3-19. Incluyendo los modelos incluidos en **IM-(Diseño Detallado**) / **IM-(Detailed-Design**), **IM-**(**Codificación**) / **IM-**(**Codification**), **IM-**(**Operación Inicial**) / **IM-**(**Initial Operation**).

La tabla 3-17 presenta el PS1Set-(PS2Set) de **IM-(Diseño Detallado)** en la que se incluye la definición de las *Unit*s: unidades o Módulos resultantes de dividir el trabajo de desarrollo el componente, a partir de su arquitectura de software. El resultado final, del diseño detallado, será la especificación de cómo se obtendrá la funcionalidad requerida de las clases en el contexto tecnológico y de normas a seguir para el desarrollo técnico.

El resultado esperado es la especificación detallada de: cada método o proceso y los atributos de los datos utilizados por dichos procesos. Es decir la especificación de cada clase y objeto a utilizar. Este PS2Set obtiene también la especificación de la prueba de las unidades / *units*.

Tabla 3-17 *PS2Set* Diseño Detallado del *PS1Set* IM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
14 IM- (Diseño Deatallado) / IM-(Detailed-Design)	Work Breakdown structure (Estructura de División del Trabajo)	Barras de Gant, módulos		
	Unit creation and maintenance (Creación y Mantenimiento de Unidades)	Carpeta de Desarrollo de Unidad (UDF), Resultados Requeridos de Unidad, Forma Web		Logic: Component
	Unit assignment (Asignación de Unidades)	UDF, Forma Web		Logic: Component
	Unit folder creation and maintenance (Creación y Mantenimiento de Carpeta de Unidad)	UDF, Forma Web		Logic: Component
(Detailed-Design:	Services To Be Used	UDF, Forma Web		Resource:

Services-Standards) (Diseño detallado:Servicios-Normas)	(Servicios a Utilizar)			Services
	Services To Be Used By Functions (Servicios a Usar por Funciones)	UDF, Forma Web		Resource: Services
(Detailed-Design: Metrics) (Diseño detallado: Métricas)	Schedule Metric Snapshots (Programar Eventos de Monitore)	UDF, Forma Web		Resource: Standards
(Detailed-Design: Processes-Data (object)) (Diseño detallado: Datos de proceso (objeto))	Standards For Processes (Normas para Procesos)	UDF, Forma Web		Resource: Standards
	Computational Standards (Normas Computacionales)	UDF, Forma Web		Resource: Standards
	Processes (Methods) (Procesos)	Descripción detallada. Archivo de procesos actualizado, Warnier		Storage: Processes
	Data Dictionary (Diccionario de Datos)	Atributos de datos de clases, Diccionario de Datos (M), Entidad-Relación (M), Forma Web	x	Storage: Basis
	Unit test specification (Especificación de Prueba de Aceptación)	Resultados Requeridos de Unidad, Forma Web		Logic: Component

En la tabla 3-18 se presenta el PS1Set-(PS2Set) de **IM(-Codificación)** en el que se codifican los archivo fuente en el lenguaje escogido a partir del PS2Set de **CIM-(Planificación)**. Se deja la compilación libre de errores y se realiza la prueba de unidad (Unit-test). La documentación de estas tareas queda en la Carpeta de unidad: UDF.

Tabla 3-18 *PS2Set* Codificación del *PS1Set* IM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
15 IM- (Codificación) / IM- (Codification)	Code Unit (Codificar Unidad)	UDF, Archs. Fuente, Forma Web		Logic: Component
	Compilation (Compilación)	Archs. Objeto, Archs. Ejecutable, Forma Web		Storage: Basis
	Linkage (Ligado)			
	Load (Carga)			
	Unit-Test (Prueba de Unidad)	UDF, Resultados Requeridos de Unidad (M), Forma Web		Logic: Component

En la tabla 3-19 se presenta el PS1Set-(PS2Set) de **IM-(Operación Inicial)** en donde se integran los archivos objeto, teniendo principal atención en las interfases utilizadas y adecuando los archivos fuente como se requiera, regenerando los archivos objeto. Para probar la integración se generan los datos y elementos necesarios para la funcionalidad de los componentes integrados. Se realizan las pruebas de integración y se genera la documentación que necesitará el usuario para utilizar los componentes.

Tabla 3-19 *PS2Set* Operación Inicial del *PS1Set* IM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
16 IM-(Operación Inicial) / IM-(Initial Operation)	Code Integration (Integración de Código)	UDF, Archs. Fuente (M), Archs. Objeto (M), Archs. Ejecutable (M), Forma Web		Logic: Component
	Structure Components (Estructurar Componentes)	Componentes (M), Forma HTML		Storage: Processes
	Integration Test (Prueba de Integración)	Resultados Requeridos de Componentes (M), Carpeta de Construcción, Forma Web	x	Logic: Component

	Elaborate User Documentation (Elqbora documentación de usuario)	User Documentation		

3.3.10 PS3Sets del PS1Set: OM

La información para los renglones numerados del 17 al 20 de las tablas 3-2 y 3-3 se muestra en las tablas 3-20 a 3-23 con mayor detalle. Incluyendo a los modelos incluidos en **OM-(Identificación de Recursos) / OM-(Resource identification)**, **OM-(Distribución) / OM-(Distribution)**, **OM-(Entrega de Producto) / OM-(Product Release)** y **OM-(Evaluación) / OM-(Assessment)**.

En la tabla 3-20 se presenta el *PS1Set-PS2Set)* **OM-(Identificación de Recursdos)** en el que se identifican los recursos humanos y materiales, de la organización, que potencialmente intervendrán en la operación del componente. Se identifican: Personas; Ubicación; Edificios; Cuartos; Computadoras; Equipo de Comunicaciones; Canales de Comunicación y Equipo Periférico.

Tabla 3-20 *PS2Set* Identificación de Recursos de la *PS1Set* OM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
17 OM- (Identificación de Recursos) / OM-(Resource identification)	Personnel List (Lista de Personal)	Forma Web		Resource: Resources
	Campus Identification (Identificar Campus)	Forma Web, Mapa de Campus	x	Resource: Resources
	Building Identification (Identificar Edificios)	Forma Web, Plano de Edificio	x	Resource: Resources
	Site / Room Identification (Identifica Sitios)	Forma Web, Plano de Cuarto	x	Resource: Resources
	Site/Room Assignation to person	Forma Web, Plano de Edificio (M)	x	Resource: Resources

	(Asignación de Sitio a Persona)			
	Communication Device Identification (Identificar Dispositivos de Comunicación)	Forma Web, Plano de Edificio (M)	x	Resource: Resources
	Channel Identification (Identificar Canales)	Forma Web, Plano de Red	x	Resource: Resources
	Processor Identification (Identificar Procesadores)	Forma Web, Plano de Edificio (M), Plano de Red (M), Plano de Cuarto (M)	x	Resource: Resources
	Peripheral Identification (Identificar Periféricos)	Forma Web, Plano de Edificio (M), Plano de Red (M), Plano de Cuarto (M)	x	Resource: Resources
	Other Resources Identification (Identificar Otros Recursos)	Forma Web, Plano de Edificio (M), Plano de Cuarto (M)	x	Resource: Resources

En las tablas 3-21 se presenta el PS1Set-(PS2Set) de **OM-(Distribución)** en el que se identifican los recursos humanos y materiales que intervienen en la operación del componente, en su contexto de operación real. Se define la distribución, en la red de computadoras, de los elementos computacionales: resultados, entradas, funciones, y datos.

Tabla 3-21 *PS2Set* Distribución del *PS1set* OM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
18 OM-(Distribución) / OM- (Distribution)	Result Used by Person (Resultados Usados por Persona)	Línea de producción (M), Plano de Edificio (M), Forma Web	x	Resource: CompNetworks
	Result Produced At Processor And I/O Devices (Resultados Producidos en Procesador y Dispositivo de	Plano de Red (M), Componentes (M), Distribución , Forma Web	x	Resource: CompNetworks

	E/S)			
	Input Introduced At (Entradas Capturadas en)	Plano de Red (M), Componentes (M), Distribución (M), Forma Web	x	Resource: CompNetworks
	Functional Options Processed At (Opciones Funcionales Producidas en)	Plano de Red (M), Componentes (M), Distribución (M), Forma Web	x	Resource: CompNetworks
	Data Stored At (Datos Almacenados en)	Plano de Red (M), Componentes (M), Distribución (M), Forma Web	x	Resource: CompNetworks
	Equipment Allocation (Ubicación de Equipo)	Forma Web, Plano de Edificio (M), Plano de Red (M), Plano de Cuarto (M)	x	Resource: CompNetworks
	Other Resources Used At Activities (Otros Recursos Usados en Actividades)	Forma Web, Plano de Edificio (M), Plano de Red (M), Plano de Cuarto (M)		Resource: CompNetworks

En la tabla 3-22 se presenta el PS1Set-(PS2Set) de **OM-(Entrega del Producto)** en el que se efectúa la *PS3Set* Prueba de Aceptación **/ Acceptance Test**, desarrollando las modificaciones y agregados necesarios para que el usuario se muestre complacido cuando se obtiene la funcionalidad que requirió.

Tabla 3-22 *PS2Set* Entrega del Producto del *PS1Set* OM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
19 OM-(Entrega del Producto) / OM-(Product Release)	Acceptance Test (Prueba de Aceptación)	Archs. Fuente (M), Archs. Objeto (M), Archs ejecutables (M), Componentes, (M) Resultados Requeri-	x	Logic: Component

		dos del Sistema (M), Resultados Requeridos de Componentes (M), Documentación del Usuario, Forma Web		

La tabla 3-23 presenta el PS1set-(PS2Set) de **OM-(Evaluación)** en el cual se efectúa una reunión de evaluación de la forma en que se desarrolló el componente y la comparación con datos históricos del desarrollo de componentes similares. Un reporte identifica oportunidades de mejora del proceso y los problemas presentados durante el proceso, proponiendo cursos de acción que mejoren la calidad o la eficiencia del proceso de desarrollo.

Tabla 3-23 *PS2Set* Evaluación del *PS1Set* OM

PS1Set – (PS2Set)	PS3Set	ProductoDelTrabajo	DGM	Dimensión: Nivel
20 OM-(Evaluación) / OM-(Assessment)	Assessment (Evaluación)	Forma Web, Gráficas Contables, Reporte de Evaluación		Logic: Component

En esta sección se presentaron todos los PS1Sets **CIM, PIM, PSM, IM** y **OM**, que incluyen los modelos o Productos-DelTrabajo / *work-products* para realizar el componente de software.

Los resultados presentados en este documento de tesis se enfocan principalmente en los PS1Sets **CIM, PIM** y **PSM**, incluyendo un ejemplo en el prototipo construido. Sin embargo consideramos muy importante incluir todos los conjuntos de modelos del PS en este anexo. Esto permite que en el futuro se retome este método a partir de la Arquitectura de Software obtenida en el PS1Set **PSM** desarrollándose a mayor detalle la construcción y la transición de acuerdo a la plataforma y el lenguaje escogido para el componente de software.

3.4Modelos Complementarios

El PS2Set **Negocio Existente / Existing Business,** de la tabla 3-2, es uno de los conjuntos que contienen *PS3Sets* donde se propone complementar los diagramas UML. Allí se incluyen tres versiones diferentes de diagramas de entidades: 1) Diagramas de Entidades a Nivel Usuario (figura 3-10), 2) Diagramas de Entidades Combinado de Usuarios y 3) el Diagrama de Entidades de la Aplicación (figura 3-11). Todos estos diagramas nos permiten mostrar una representación abstracta y estática, de la forma en que trabaja actualmente la organización. Es decir, un modelo conceptual.

WPs agregados a los 13 diagramas existentes en el lenguaje UML 2.0, nos permitirán comenzar la utilización de los elementos del lenguaje UML a partir de una representación estática simple de la forma en que se trabaja o se pretende trabajar. Esto permite q desarrolladores sin experiencia en el dominio de aplicación, mejora la práctica actual que pide que los primeros diagramas los desarrollen especialistas en el dominio de la aplicación. La inclusión de la representación estática complementaria tiene la meta de reducir la dependencia (relativa a expertos) de los resultados obtenidos.

La intervención de los expertos en el dominio de la application es más eficiente cuando se inicia con elementos proporcionados por la representación estática del Diagrama de Entidades a Nivel Aplicación; esta situación contrasta con el punto de partida sugerido en el Proceso Unificado de Rational (RUP) que requiere el desarrollo de casos de uso mediante la intervención de desarrolladores experimentados.

El Diagrama de Entidades a Nivel Aplicación, se obtiene mediante la elaboración de un conjunto de Diagramas a Nivel Usuario, como el mostrado en la figura 3-10.

El diagrama de la figura 3-10 es uno de los Diagramas de Entidades a Nivel Usuario que se presentan en el caso de estudio del "Programador de Reuniones" incluido en el prototipo del CD del anexo C. Mediante estos diagramas se representa la solución propuesta para el problema-modelo enunciado por Rob Allen y complementado por Axel Van Lamsweerde [LDM92].

El problema-modelo del Programador de Reuniones se enuncia como sigue:

> *Mantener itinerarios coherentes de reuniones para varios individuos. Se registra al menos la hora, duración y los participantes en cada reunión. Algunas de las reuniones pueden incluir personas cuyos programas de trabajo no los mantiene el programador. Puede agregar o eliminar una reunión en cualquier momento (mientras ésta no haya ocurrido) y también pueden agregarse o eliminarse participantes. Una reunión puede programarse a cualquier hora conveniente para todos (o un número suficiente de) los participantes. Algunas reuniones requieren ocurrir en un orden específico. El desarrollador puede mantener información acerca de las preferencias de programación para cada usuario.*

Como se observa en la figura 3-10 el diagrama de Entidades a Nivel Usuario ubica en el centro a una elipse con el nombre del usuario de quien se recopila información para entender su participación. En seguida se dibujan las entidades, con las que el usuario tiene alguna interacción, como elipses en la periferia del área disponible. Luego se dibujan flechas etiquetadas que representan interacciones o transacciones entre las entidades. Por último, se revisa el diagrama y se vuelve a dibujar si es necesario, siguiendo el procedimiento sugerido por Ken Orr [O81].

Cuando se termina de dibujar todos los Diagramas de Entidad a Nivel Usuario, de los actores involucrados, tenemos representaciones individuales de tarea que la aplicación. Si se requiere ahora tener una idea completa, integramos todos los diagramas en uno: el Diagrama de Entidades Combinado a Nivel Usuario.

En el diagrama de entidades combinado se tiene mucha información, incluye todas las interacciones entre las entidades relacionadas con el sistema. Para tener sólo las interacciones críticas se construye el Diagrama de Entidades a Nivel Aplicación, que se logra dibujando una frontera alrededor de las entidades internas. Se representa la aplicación cuando se consolidan todas las entidades internas en una sóla elipse que representa a la aplicación (figura 3-11).

Una primera versión de diagramas de casos de uso se produce a partir del Diagrama de Entidades a Nivel Aplicación y de los resultados obtenidos directamente a partir de dicho diagrama, como son la lista de objetivos y los Diagramas de Línea de Producción (figura 3-13). Los casos de uso resultantes se validan con los usuarios de la aplicación o pueden utilizarse para tener un medio de comunicación mejor y más eficiente con expertos en desarrollo de software del dominio específico.

Los Diagramas de Línea de Producción (figura 3-12) muestran las salidas/entradas (nombre del elemento en el área reservada al nombre del símbolo que representa una clase, ej. R-PORE-C) y procesos principales (nombre del procedimiento, en el área reservada para la operación del símbolo de clase, ej. Envía (U001)), conectados en un flujo (mediante flechas). Las flechas representan las transformaciones entre elementos de la línea de producción. Este tipo de diagramas se obtiene a partir del Diagrama de Entidades a Nivel Aplicación y de la lista de objetivos obtenida, siguiendo un procedimiento simple:

- En el diagrama de entidades a nivel aplicación, numerar las transacciones (mediante etiquetas asociadas a las flechas), considerando condiciones ideales (ninguna de las entidades responsables de la producción falla), iniciando con la transformación que se realiza primero en el proceso y siguiéndolo paso a paso.
- Diagramar una línea de ensamble para las salidas principales del sistema Colocando a la izquierda la transacción numerada al último y trabajando hacia atrás (la siguiente transacción con el número menor inmediato, colocándola en el dibujo hacia la derecha) hasta que se incluya la transacción inicial.
- Conforme se incluyan las transacciones, en el diagrama de la línea de ensamble, se marcan los objetivos correspondientes de la lista.
- Al terminar una rama de la línea de ensamble, para cualquier transacción que falte incluir, se inicia una línea de ensamble separada y se trabaja hacia atrás

hasta que ocurra una transacción, que se incluyó previamente en otra rama del proceso.

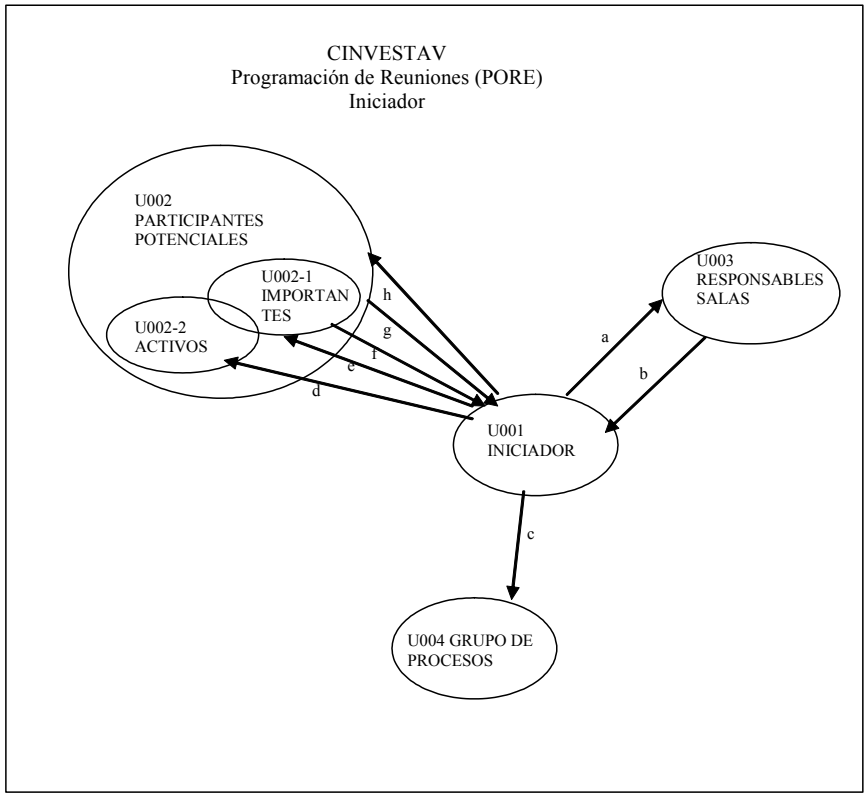


Figura 3-10 Diagrama de Entidades a Nivel Usuario, correspondiente al Iniciador de la reunión.

El diagrama de línea de producción mostrado en la figura 3-12 es una porción del diagrama incluido en el prototipo incluido en el CD del anexo C. El diagrama completo se puede acceder siguiendo la cadena de opciones siguiente: **Configura >Manejo de Productos del Trabajo > Despliegue /Impresión > PS1 > Product Set > CIM: Componente Identification > SCOPE**.

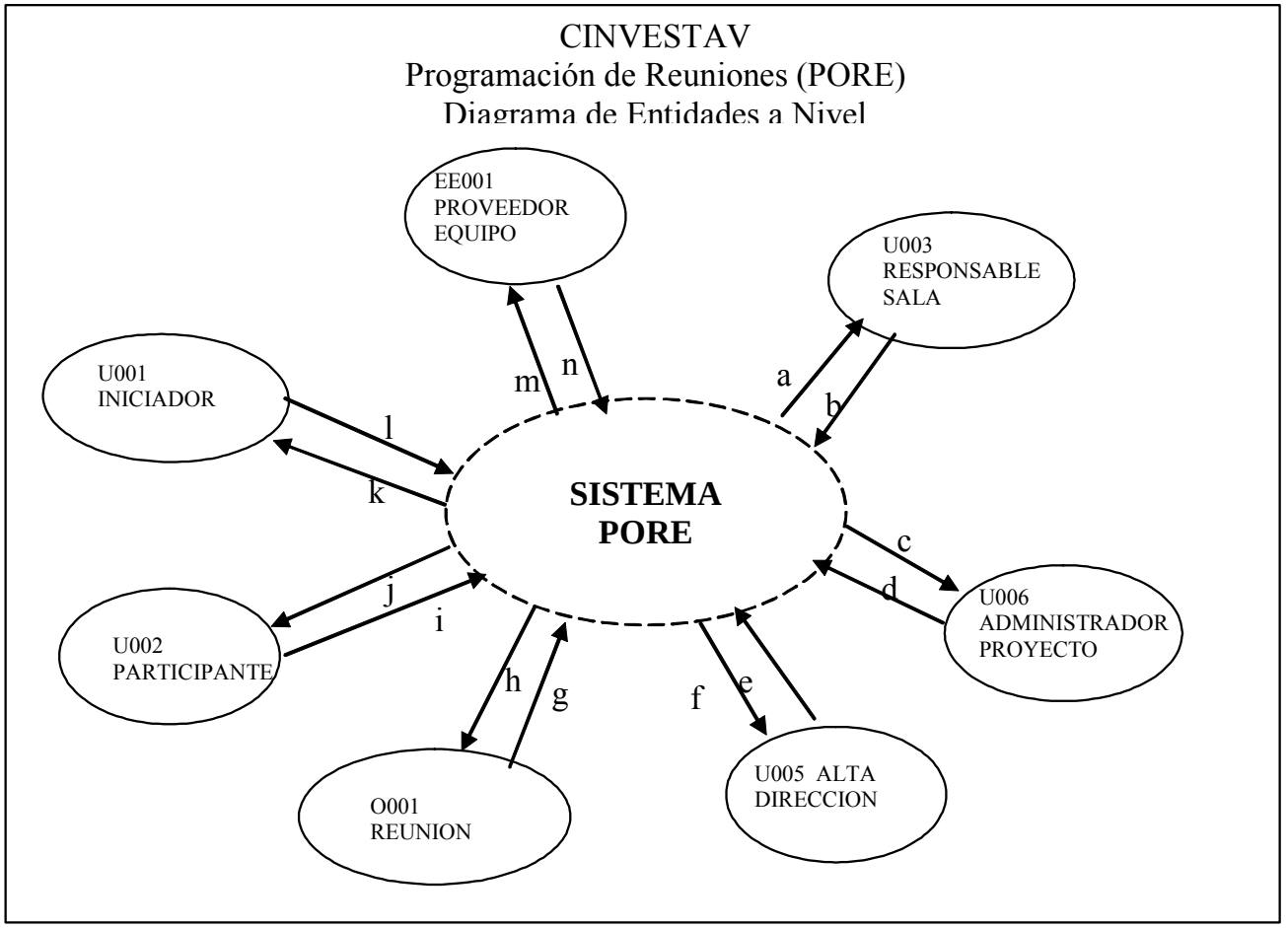


Figura 3-11 Diagrama de Entidades a Nivel Aplicación, del sistema PORE.

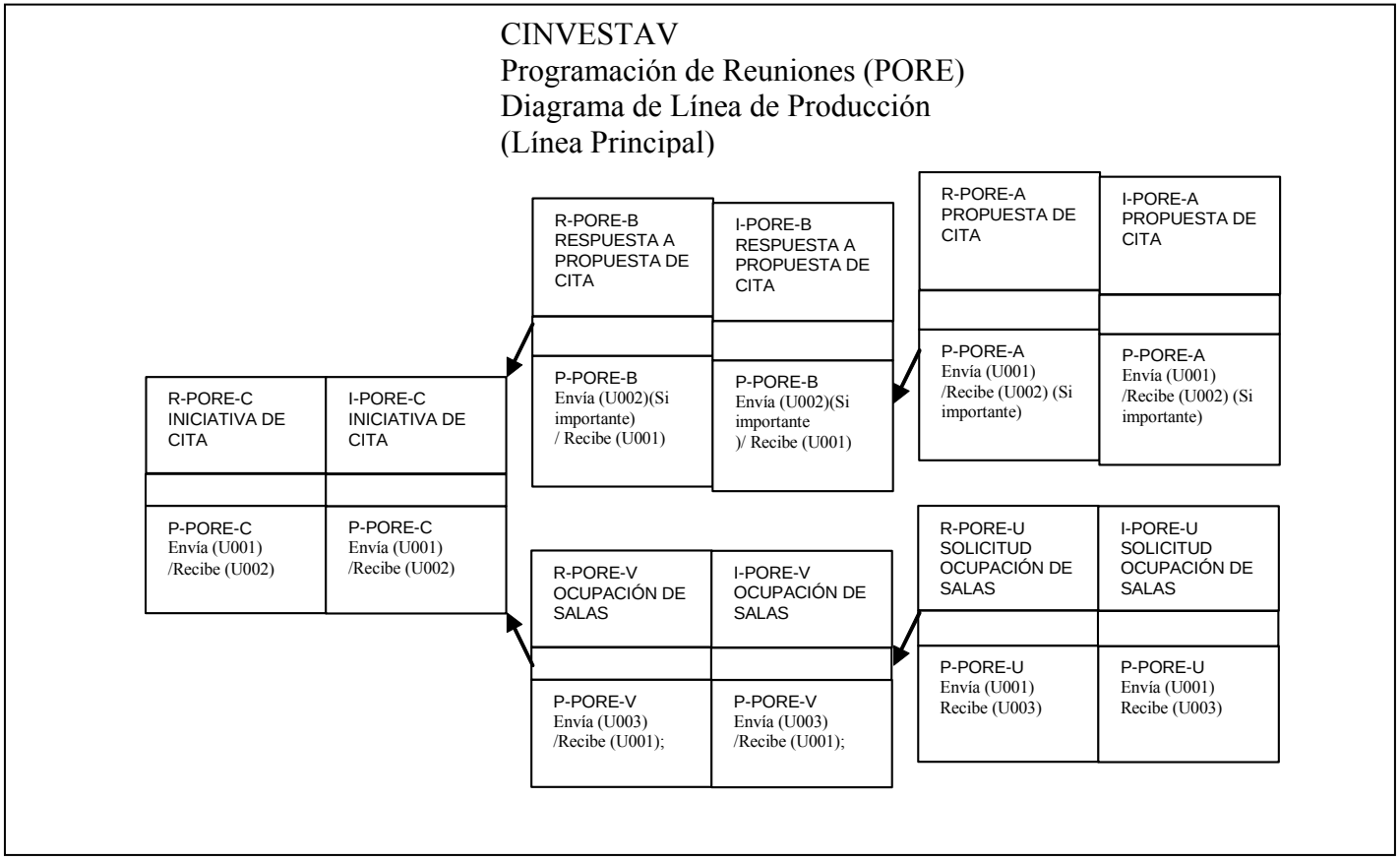


Figura 3-12 Diagrama de Línea de Producción.

El nivel de abstracción es mayor en los diagramas de entidades que en los diagramas de casos de uso (del UML). Considerando que se tiene mayor nivel de abstracción, cuando es mayor el número de detalles no considerados intencionalmente. El nivel de abstracción mayor es la razón por la que, realizando la secuencia de diagramas del PS1Set-(PS2Set) **CIM-**(**Negocio Existente**), de la figura 3-11 y el PS2Set **CIM** (**Identificación de Componentes**), mostrada en la tabla 3-7, el desarrollador obtiene una imagen más simple del proceso real.

Examinando la secuencia sugerida en la tabla 3-2, observamos que para sintetizar fácilmente los diagramas del UML: casos de uso y diagramas de clases, se cuenta con la información identificada durante la realización de CIM-(Identificación de Componentes), del renglón cuatro. Esto se infiere ya que los diagramas de casos de uso y de clases, se tienen que realizar en PIM-(Ingeniería Inversa) y/o en PIM-(Ingeniería Directa)**,** colocados en los renglones numerados como siete y ocho de la tabla 3-2.

La información incluida hasta aquí completa la descripción del Conjunto de Productos (PS) / Product Set (PS).

3.5Conclusión

Se otorga la importancia principal al Conjunto de Productos (PS) que se sugiere desarrollar, supeditando el proceso de desarrollo al PS que el equipo se compromete a construir, según el contexto en que se usará el software y la forma en que se desarrollará.

Los productos requeridos en la categoría Ingeniería se detallan, especificando una secuencia explícita de realización de los diagramas utilizados, en sus cuatro niveles jerárquicos.

El conjunto de productos PS identifica los modelos UML a incluir en los conjuntos CIM, PIM, PSM, IM y OM, ayudando a elegir el nivel de abstracción utilizado en cada caso. La colocación de los modelos en sus conjuntos contenedores específicos permite dimensionar el proceso generico MDP requerido para generar cada producto-del-trabajo.

Algunos diagramas se añaden a los modelos incluidos en UML, mejorando la representación estática del dominio de aplicación, antes de comenzar el desarrollo de los diagramas de casos de uso. Los tipos de diagramas adicionales disminuyen la necesidad de que el proyecto cuente con expertos en el dominio de la aplicación, debido a que permiten representar la realización en curso de las operaciones en el ambiente problema.

Colocamos al producto de software y su arquitectura, en el papel principal del desarrollo. Aportando un Conjunto de Productos / Product Set (PS), que sugiere para la categoría de Ingeniería, de áreas de proceso: a) un orden de construcción, b) los modelos a utilizar y c) modelos apropiados para el modelado del negocio.

4 MDP: Proceso Dirigido por Modelos

4.1Introducción

El Proceso Dirigido por Modelos / Model Driven Process (MDP), satisface los siguientes requerimientos:

Nombre	Descripción
Desarrollo en equipo	El desarrollo de software requiere trabajo en equipo disciplinado y coordinado, asignando responsabilidades y tareas a los participantes.
Simplificación	La complejidad del proceso se reduce evitando denominaciones semejantes a niveles diferentes (Ej. nombres de fases Vs. disciplinas), mejorando la interpretación y realización incremental del proceso.
Flexibilidad	Los procesos no deben ser secuenciales ni rígidos, de tal manera que el proceso pueda modificarse, incluso aún cuando se realicen transformaciones automáticas entre modelos.
Agilidad	El proceso debe permitir la adecuación dinámica y realización en forma ágil: incremental, cooperativo, directo y adaptable, es decir adecuar productos, sub-procesos y formas.

El proceso de AGD cede el lugar principal al conjunto de productos. Además, la complejidad del proceso disminuye, eliminando los conceptos asociado al nivel grueso del proceso (ej. fases).

Con la finalidad de estructurar el proceso se identifica un conjunto de formas comunes de trabajo, generalizando cuatro modos de trabajo-en-grupo:

- Colaborativo / Collaborative
- Cooperativo / Cooperative
- Control y / Control
- Equipo / Team.

Del análisis del desarrollo de software derivamos la hipótesis de que estos modos de trabajo incluyen la mayoría de los protocolos realizados durante la elaboración del software. Considerando, durante la descripción de los modos de trabajo, la inclusión de los mecanismos necesarios para permitir su realización flexible y ágil (ej. orden de realización de sub-procesos dependiendo de las necesidades).

Cada modo de trabajo se desarrolla por un grupo cuyos miembros interpretan varios *roles* diferentes, con responsabilidades específicas. El modo de Trabajo en Equipo / Team mode, da contexto a los otros tres modos y consta de siete sub-procesos que se introducen en la § 4.2 (ej. al terminar el sub-proceso Evaluación, del modo Equipo, se inicia una Sesión de Seguimiento, del modo Control).

La ejecución concurrente de los modos de trabajo-en-grupo, controlan el esfuerzo realizado ajustándose al conjunto de productos / Product Set (PS) del método para Desarrollo Arquitectónico en Grupo / Architectural Group Development (AGD). El proceso controla el esfuerzo requerido variando el tamaño del producto a construir (ej. no se programa la elaboración del PIM - Requerimientos de Software, sino cada uno de los modelos que la integran).

La variación necesaria del producto, se logra agrupando los modelos o separándolos, dependiendo del tamaño del componente o sistema a construir. Se considera que el trabajo requerido, para realizar el producto, se mantenga simple de administrar.

A continuación se explica la estructura del MDP, se describen los modos de trabajo en grupo, se incluye el patrón del modo de proceso Equipo, se detallan los aspectos dinámicos del proceso.

4.2Estructura del MDP

El proceso de software es: "una colección de actividades relacionadas, vistas como un proceso coherente sujeto a razonamiento e involucrado en la producción de un sistema de software" [WGDGKR99]. Las actividades realizadas por un grupo de trabajo abstraen varios modos de trabajo-en-grupo:

1. *Equipo / **Team***, las responsabilidades del desarrollo de software, con calidad, se distribuyen entre los miembros del equipo cuyos roles son diferentes y se realiza mediante un grupo de sub-procesos que se ejecutan sistemáticamente;
2. *Colaborativo / **Collaborative***, la responsabilidad la comparten varios actores, representando alternativamente el rol de conductor, quien toma las decisiones;
3. *Cooperativo / **Cooperative***, evalúa un producto o documento, mediante reglas que evitan los conflictos cuando un grupo de individuos toma un acuerdo;
4. *Control / **Control***, revisión y definición de posibles cursos de acción para mejorar el proceso.

Es posible, también, trabajar en forma individual; aunque se trata como un caso especial del modo Colaborativo / Collaborative). En el trabajo individual, hay únicamente un conductor / **driver**, sin incluir ningún otro rol: ni un co-conductor / **co-driver** ni tampoco un *asistentes* / **assistant**.

MDP está dirigido por modelos y siempre se seguirán los mismos pasos (sub-procesos) [M94], para obtener cualquier modelo (o conjunto de modelos) requerido en la resolución de un problema.

Este principio permite realizar el mismo proceso sistemáticamente. Sin importa el conjunto de productos a realizar, siempre se utilizan los mismos sub-procesos; ajustando los parámetros al caso específico –ej. las métricas utilizadas.

Un ejemplo del conjunto de productos, a construir, es la categoría "Ingeniería": *con sus niveles de abstracción: PS1Set*, *PS2Set*, PS3Set y WP; aunque también pueden ser los productos definidos para cualquiera otra áreas de proceso del CMMI (ej. "Administración de Configuración", de la categoría "Soporte").

Los modos de trabajo-en-grupo, están organizados mediante sub-proceso realizados por actores con roles definidos mediante responsabilidades sociales y administrativas. El modo Equipo, constituyen un núcleo de proceso sistemático, constituido por los sub-procesos:

1. *SP-LAUNCH & STRATEGY* (Lanzamiento & estrategia): Define el problema y organiza al grupo de trabajo. Incluye la definición del sistema de valores para evaluar los resultados y la forma de trabajo,
2. *SP-PLANNING* (Planificación): Planificación de cada miembro del equipo e integración del plan para el grupo,
3. *SP-REQUIREMENTS* (Requerimientos): Identifica los requerimientos del cliente/usuario y se analizan para determinar los ensambles/partes del producto a realizar y los participantes,
4. *SP-ANÁLISIS & DESIGN* (Análisis y Síntesis): Sintetiza soluciones y se analizan éstas para proponer la mejor,
5. *SP-IMPLEMENTATION* (Implementación): Construye los elementos de la solución dejándolos en operación y sin errores,
6. *SP-TEST* (Prueba): Integra, prueba y adecua el funcionamiento en conjunto de los elementos construidos. Construcción de los documentos necesarios para que el cliente/usuario utilice el producto integrado.
7. *SP-POSTMORTEM* (Pos-término): Evalúa la forma en que se trabajó, usando el sistema de valores definido en el sub-proceso SP-Launh & Strategy.

4.3Modos de Trabajo-en-Grupo

Los resultados son el producto de las actividades realizadas por los participantes del proyecto. La actividad tiene artefactos de entrada y de salida. Así mismo, los artefactos son productos del proceso, e insumos para realizar otras actividades y producir otros artefactos.

Se define el comportamiento y responsabilidad de un grupo de individuos, para producir en equipo. Todo participante tienen el rol de Miembro del Equipo / Team Member y potencialmente puede realizar otros roles concurrentemente (al menos un rol por cada modo de trabajo-en-grupo).

Las habilidades administrativas y sociales definen a los roles. Esta forma de caracterizar los roles difiere de cómo se realiza la definición en RUP. En RUP se considera habilidades técnica (ej. arquitecto, diseñador de casos de uso).

Cada modo de trabajo-en-grupo define un conjunto específico de roles:

- ***Modo en Equipo***: 1)Miembro del equipo / *Team Member*, 2) Líder del Equipo / *Team Leader*, 3) Administrador de Planeación / *Planning Manager*, 4) Administrador de Soporte / *Support Manager*, 5) Administrador de Desarrollo / *Development Manager*, 6) Administrador de Calidad/Proceso / *Quality/Process Manager*, 7) Cliente / *Client* (externo) y 8) Administrador del Proyecto / *Project Manager* (externo)
- ***Modo Colaborativo***: 1) Conductor / *Driver*, 2) Co-conductor / *Co-driver* y 3) Asistente / *Assistant*
- ***Modo Cooperativo***: 1) Facilitador / *Facilitator,* 2) Apuntador / *Writer*, 3) Revisor / *Reviewer*, 4) Observador / *Observer*, y 5) Productor / *Producer*

- ***Modo de Control***: 1) Líder / *Leader*, 2) Apuntador / *Recorder* y 3) Participante / *Participant*

Un ejemplo de la asignación de roles para un particiante, en tres momentos diferentes en el tiempo es:

- en el momento x, es:
  - en el modo Equipo: miembro del equipo y Administrador de Planificación
  - para el modo Colaborativo: Conductor.
- en el momento y, es:
  - en el modo Equipo: miembro del equipo y Administrador de Planificación
  - para el modo Cooperativo: Revisor.
- en el momento z, es:
  - en el modo Equipo: miembro del equipo y Administrador de Planificación
  - para el modo Control: Apuntador.

Una representación estática de la forma en que se realiza el trabajo para seguir la estructura de procesos del ambiente MDP, se muestra en el Diagrama de Entidades Combinado a Nivel Usuario de la figura 4-1. Los modos de trabajo en grupo: Equipo, Colaborativo, Cooperativo y Control, se representan por una elipse mayo englobando otras elipses menores que representan a los actores y sus roles. Las relaciones entre roles se representan por flechas.

Las relaciones entre los roles, de la figura 4-1, tienen asociadas unas etiquetas (ej. a), enlistadas en el anexo de este capítulo. Para cada relación se detalla la etiqueta, incluyendo los nombres de los *productos-del-trabajo* (modelos), formas y documentos intercambiados.

En el anexo A de esta tesis se incluye la explicación del patrón (MOPROSOFT) utilizado para describir cada uno de los modos de trabajo en grupo, así como las instancias del patrón que describen a los modos de trabajo-en-grupo: Colaborativo, Cooperativo y Control.

La estructura del MDP, desde la perspectiva de un participante, se persive como la del modo Equipo del trabajo-en-grupo. Este modo tiene un sólo nivel de proceso, sin conceptos que traslapen su significado. Se evita así el problema que tienen los procesos con dos niveles, como el Proceso Unificado de Rational / Rational Unified Process (RUP) [RUP03a] y la Especificación del Metamodelo para Ingeniería del Proceso de Software / Software Process Engineering Metamodel Specification SPEMS [OMG-SPEMS02], que tienen definiciones similares de fases (en el 1er nivel) y disciplinas (en el 2º nivel). Dado que MDP tiene un nivel de proceso, evita tener varios conceptos con definiciones semejantes y resulta un proceso claro y sencillo.

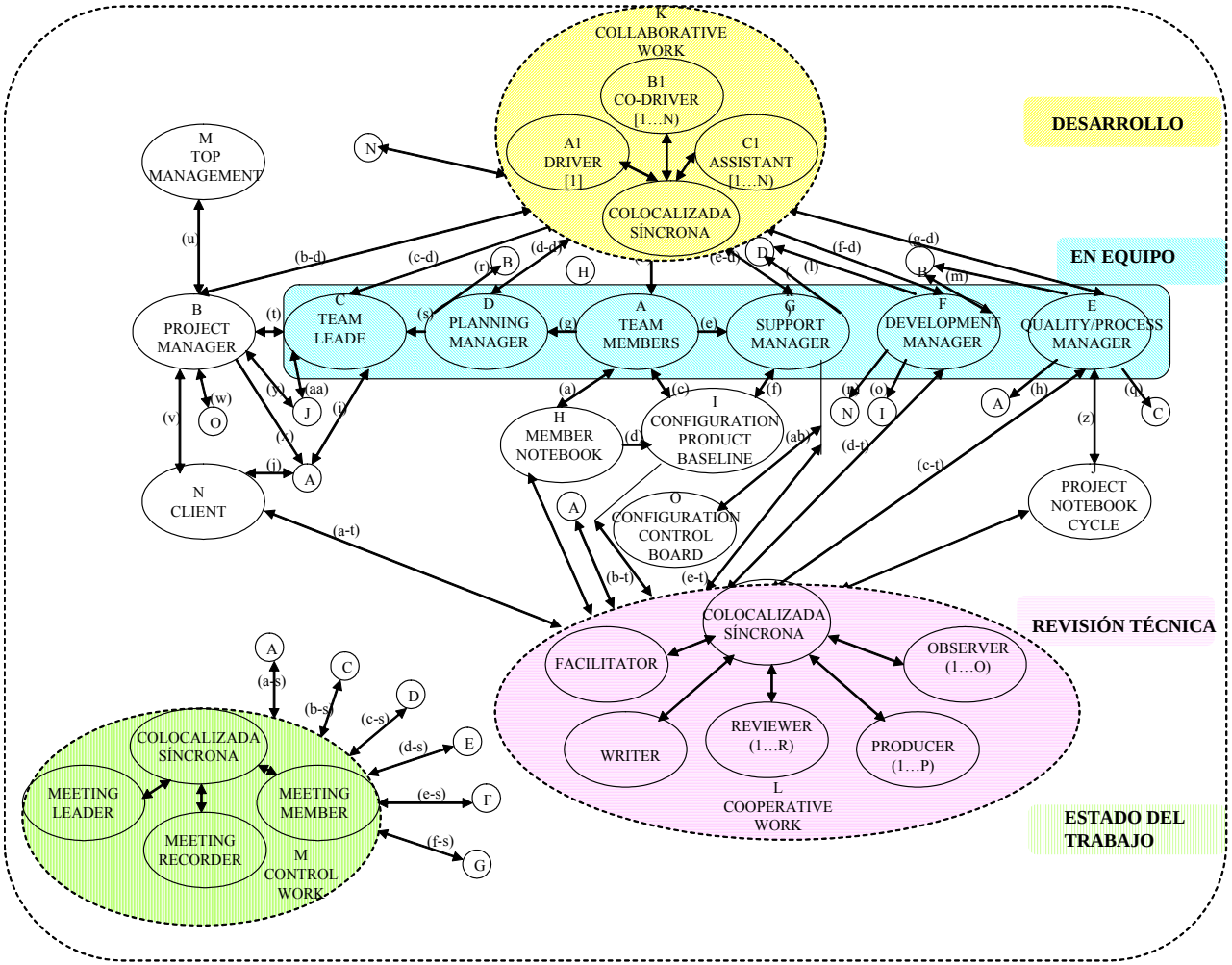


Figura 4-1 Estructura de Procesos del Proceso Dirigido por Modelos (MDP).

A continuación incluimos el patrón del modo "Equipo" de trabajo-en-grupo, pues este modo sirve de referencia para iniciar y realizar los otros modos de trabajo en grupo del MDP.

4.4Patrón del Modo de Proceso en Equipo (MP-TE)

4.4.1 Definición general del mode de proceso

MODO DE PROCESO	Mp-TE: modo de proceso en equipo
Categoría/Modo	Modos de Trabajo en Grupo
Propósito	Desarrollo de cualquier producto-del-trabajo requerido en las áreas de proceso necesarias para obtener el producto de software deseado. El modo de trabajo en Equipo da contexto a los otros tres modos (Colaborativo, Cooperativo y de Control (ver figura 4-1) y consta de siete sub-procesos que se repiten sistemáticamente. Se maneja la tesis de que una vez definidos los resultados a obtener (el modelo o conjunto de modelos requerido de cualquiera de las áreas de proceso) siempre se tendrán que seguir los mismos pasos (sub-procesos) [M94].
Descripción	No importa el conjunto de productos (Ej. Los conjuntos de la categoría Ingeniería de áreas de proceso: *PS1Set*, *PS2Set* o PS3Set (del método AGD), o los productos definidos para otra cualquiera de las áreas de proceso del CMMI), que se realice actualmente, siempre se utilizan los mismos pasos (sub-procesos) como se muestra en las figuras 4-2 y 4-5 (cambiando sólo algunos parámetros para cada caso (ej. las métricas utilizadas). Para controlar el proceso se utilizan formas de captura de datos (instancias adecuadas de las formas utilizadas en TSP-PSP).
Objetivos	Desarrollo de un producto-del-trabajo específico, proporcionando el contexto necesario para asegurar O1 Lograr las metas específicas (obtener el producto-del-trabajo requerido). O2 Institucionalizar un proceso administrado. O3 Institucionalizar un proceso definido. O4 Institucionalizar un proceso administrado cuantitativamente. O5 Institucionalizar un proceso optimizante.
Indicadores	Para las áreas de proceso de la categoría de Ingeniería, se consideran

los indicadores siguientes [H99]:

Contenido de MDP-SUMQ:

I1 Percent Defect Free

I2 Defect/page

I3 Defects/KLOC

I4 Defect ratios

I5 Development time ratios (%)

I6 A/FR

I7 Personal review rates

I8 Inspection rates

I9 Tasas de inserción de defectos (Defectos insertados por hora)

I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)

I11 Yields de las fases (Defectos eliminados en la fase / Defectos eliminados en la fase + ESCAPES)

I12 Yield del proceso

Para las demás áreas se establecen ad hoc, documentándose los indicadores utilizados.

Metas cuantitativas

Para las áreas de proceso de la categoría de Ingeniería, se consideran los indicadores siguientes [H99]:

I1 Percent Defect Free (PDF)

Compilación	> 10%
Prueba de unidad	> 50%
Prueba de integración	> 70%
Prueba de sistema (aceptación)	>90%

I2 Defect/page

I3 Defects/KLOC

Total de defectos inyectados	76 – 150
Compilación	< 10
Prueba de unidad	< 5
Integración	< 0.5
Prueba de sistema	< 0.2

I4 Defect ratios

Defectos revisión DLD/Defecto en prueba de unidad	> 2.0

Defectos revisión de código/Defectos

Compilación	> 2.0

I5 Development time ratios (%)

Inspección de requerimientos/Tiempo

requerimientos > 0.25
Inspección HLD/Tiempo HLD > 0.5
DLD/Tiempo codificación > 1.0
Revisión DLD/Tiempo DLD > 0.5
Revisión codificación/Tiempo codificación > 0.5
I6 A/FR
I7 Personal review rates
I8 Inspection rates
Páginas de requerimientos/Hora < 2
Páginas HLD/Hora < 5
Líneastexto DLD/Hora < 100
LOCs/Hora < 200
I9 Tasas de inserción de defectos (Defectos insertados por hora)
Defectos requerimientos/Hora 0.25
Defectos HLD/Hora 0.25
Defectos DLD/Hora 2.0
Defectos código/Hora 4.0
Defectos compilación 0.3
Defectos prueba unidad 0.2
I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)
Defectos inspección requerimientos/Hora 0.5
Defectos inspección HLD/Hora 0.5
Defectos inspección DLD/Hora 2.0
Defectos revisión código/Hora 6.0
Defectos inspección código/Hora 1.0
I11 Yields de los sub-procesos
Inspección de requerimientos - 70%
Revisiones e inspecciones de diseño - 70%
Revisiones e inspecciones de código - 70%
Compilación - 50%
Prueba sistema con 5 o menos
defectos/K LOCs - 90%
Prueba sistema a < 1.0 defectos/KL - 80%
I12 Yield del proceso
Antes compilación >75%

Antes prueba unidad	> 85%
Antes de integración	> 97.5%
Antes de prueba sistema	> 99%

Responsabilidad y autoridad

La autoridad para validar los resultados del proceso es el Project Manager. Aunque este rol se considera externo al equipo de trabajo (ver adelante).

En la figura 4-1 se muestra el trabajo en equipo (diagonales hacia la derecha) con sus roles, interrelaciones y sub-procesos. Donde se muestra que el trabajo lo realiza un equipo (grupo) de desarrollo cuyos miembros, además de interpretar el *rol* de ingenieros de desarrollo (**Team Member**), interpretan también las **responsabilidades** de cinco *role*s adicionales (elipses alineadas horizontalmente): 1) **Team Leader** (Líder del Equipo); 2) **Planning Manager** (Administrador de Planificación); 3) **Support Manager** (Administrador de Soporte); 4)**Development Manager** (Administrador de Desarrollo), y 5)**Quality/Process Manager** (Administrador de Calidad/Proceso).

Hay tres *role*s adicionales a los mencionados para el equipo, que también son parte del contexto del trabajo de desarrollo de software: 1) **project manager***;* 2) **client** y *3)* **top management** .

Project manager.- administra los recursos que necesita el equipo: humanos y materiales. Acuerda con el *client* y con *top management.* Tiene la última decisión cuando hay conflictos de cualquier índole.

Client.- utilizará el componente desarrollado o es el actor que solicita el desarrollo. Es el propietario de las necesidades y restricciones, impuestos al desarrollo.

Top management.- ubica al desarrollo de software en la organización. Representa al grupo de actores que determina el contexto del desarrollo incluyendo los rangos permitidos para todos los parámetros de calidad: del proceso de desarrollo y del producto de software.

Subprocesos

Lista de procesos de los cuales se compone el proceso en cuestión.

Estos pasos (sub-procesos) son:

SP-LAUNCH & STRATEGY (Lanzamiento y estrategia): Define el problema y organiza el grupo de trabajo. Define el sistema de valores para evaluar los resultados y la forma de trabajo

P-PLANNING (Planificación)*:* Planificación de cada miembro del equipo y del grupo

SP-REQUIREMENTS (Requerimientos, Análisis)*:* Identifica los requerimientos del cliente/usuario y se analizan para determinar las partes y los participantes

SP-ANÁLISIS & DESIGN (Análisis y Síntesis): Sintetiza soluciones y se analizan éstas para proponer la mejor

	SP-IMPLEMENTATION (Implementación): Construyen los elementos de la solución dejándolos en operación y sin errores **SP-TEST (Prueba)**: Integra los elementos y se prueban optimizando su funcionamiento en conjunto, construyendo los documentos necesarios para su utilización por el cliente/usuario **SP-POSTMORTEM (Pos-término)**: Evalúa la forma en que se trabajó usando el sistema de valores definido previamente
Procesos relacionados	Da contexto a los modos de trabajo en grupo: colaborativo, cooperativo y control. Los cuatro modos de trabajo-en-grupo integrados (ver figura 4-1) se utilizan para obtener los productos-del-trabajo de todas las áreas de proceso.

4.4.2

4.4.3 Entradas

Nombre	Descripción	Fuente
Entrada-1 del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate
Entrada-2 del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate
Entrada-n del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate

4.4.4 Salidas

Nombre	Descripción	Destino
Producto-1 del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate

Producto-2 del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate
Producto-n del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate

4.4.5 Productos internos

Nombre	Descripción
MDP-LOGT Time Recording Log	Tiempo ocupado en cada tarea del proyecto
MDP-LOG Defect Recording Log	Datos acerca de los defectos encontrados y corregidos
MDP-LOGTEST Test Log	Sumario de las pruebas realizadas y los resultados obtenidos
MDP-INFO Member Information Sheet	Intereses y experiencia de un participante
MDP-STRAT Strategy Form	Decisiones estratégicas
MDP-PEER Team and Peer Evaluation	Evaluación del equipo y de los participantes
MDP-PIP Process Improvement Proposal	Problemas del proceso e ideas de mejora
MDP-SUMDI Defect Injected Summary	Acumula datos acerca de defectos insertados en las partes de un ensamble
MDP-SUMDR Defects Removed Summary	Acumula los datos acerca de defectos eliminados de las partes de un ensamble
MDP-SUMP Plan Summary	Mantiene datos planeados y reales para ensambles del producto
MDP-SUMQ Quality Plan	Mantiene datos de calidad planeados y reales para partes y ensambles
MDP-SUMS Size Sumary	Acumula datos acerca del tamaño del producto
MDP-SUMT Development Time Summary	Acumula datos acerca del tiempo de desarrollo ocupado en partes de los ensambles
MDP-SUMTASK Task Summary	Mantiene datos para facilitar la planeación y seguimiento de las tareas

MDP-TASK Task Planning Template	Estimación del tiempo de desarrollo, el valor planeado y la fecha de terminación de cada tarea
MDP-WEEK Weekly Status Report	Reporte del estado, semanal
MDP-SCHEDULE Schedule Planning Template	Horas estimadas y reales semanalmente
MDP-CSR Comfiguration Status Report	Información acerca del estado seminal del sistema para administración de configuración del software (SCM)
MDP-ITL Issue Traking Log	Seguimiento a los riesgos y temas pendientes del proyecto
MDP-INS-DET Inspection Report-Detail	Datos detallados acerca de una inspección/revisión
MDP-INS-SUM Inspection Report-Sum	Sumario del análisis hecho a los datos de una inspección/revisión
MDP-CCR Configuration Change Request	Envío de un elemento al comité de control de cambios (ccb) para su inclusión en la líneabase

4.4.6 Referencias

- H99 — Humphrey; Introduction To Team Software Process; Addison-Wesley 1999, Isbn: 0-201-47719-X.
- SEI-CMMI-C — Capability Maturity Model® Integration (CMMi℠), Version 1.1; CMMI℠ For Systems Engineering, Software Engineering, Integrated Product And Process Development, And Supplier Sourcing, (CMMI-SE/SW/IPPD/SS, V1.1) - Continuous Representation; CMU/SEI-2002-TR-011, ESC-TR-2002-011
- SEI-CMMI-S — Capability Maturity Model® Integration (CMMI℠), Version 1.1; CMMI℠ For Systems Engineering, Software Engineering, Integrated Product And Process Development, And Supplier Sourcing, (CMMI-SE/SW/IPPD/SS, V1.1) - Staged Representation; CMU/SEI-2002-Tr-012 , ESC-TR-2002-012
- GM04 — Moisés González-García, Ana María Martínez-Enríquez; The Architectural And Group Development Method: An Experimentation; Workshop On QUANTITATIVE TECHNIQUES FOR SOFTWARE AGILE PROCESS (QUTE-SWAP), Acm Sigsoft 2004 / FSE-

12 (Foundations On Software Engineering); Oct. 31 – Nov. 6, 2004.

4.4.7 Prácticas

4.4.7.1 Identificación de roles involucrados y capacitación requerida

Rol	Abreviatura	Objetivo
		CARACTERÍSTICAS (Capacitación)
Roles internos del equipo		
Team Member (Ingenieros de desarrollo)	TE-TM	Trabajar como ingeniero de desarrollo, desarrollando en grupo con los demás miembros del equipo, hacer su propio trabajo utilizando sus habilidades lo mejor posible y trabajar de acuerdo al MP-TE: Modo de Proceso en Equipo. Registrar los datos acerca de su proceso Entregar los datos de las tareas completadas al TE-PM Acumular datos de calidad y entregar al TE-QPM Reportar cambios a la configuración controlada al TE-SM Reportar los temas importantes a seguir al TE-SMObservar y analizar las métricas acerca de su trabajo personal para ayudarse a realizar trabajo de calidad
Team Leader (Líder del Equipo)	TE-TL	Guía al equipo, asegurándose de que los ingenieros reportan los datos de su proceso y que terminan su trabajo como se planeó Le gusta ser líder y naturalmente asume el rol de líder Puede identificar los asuntos importantes y tomar decisiones objetivas No le importa, ocasionalmente, tomar acciones impopulares y desea presionar ala gente para que logre tareas difíciles Respeta a sus compañeros, desea oír sus puntos de vista y quiere ayudarlos a desempeñarse al nivel

		máximo de sus habilidades
Planning Manager (Administrador de Planificación	TE-PM	Soporta y guía a los miembros del equipo en la planificación y seguimiento de su trabajo Mente lógica sintiéndose a gusto siguiendo un plan para realizar el trabajo Aunque no siempre pueda producir un plan, tiende a planificar su trabajo cuando tiene oportunidad Interesado acerca de los datos del proceso Desea insistir a las personas acerca de seguir y medir su trabajo
Support Manager (Administrador de Soporte)	TE-SM	Soporta al equipo en la determinación, obtención y manipulación de las herramientas para satisfacer las necesidades tecnológicas y administrativas del equipo Interesado en herramientas y métodos Usuario de computadoras competente y siente que puede ayudar al equipo con sus necesidades de soporte Tiene experiencia con herramientas y sistemas de soporte Familiarizado con las herramientas que se usarán en el proyecto
Development Manager (Administrador de Desarrollo)	TE-DM	Guía al equipo en la definición, diseño, desarrollo y prueba del producto-del-trabajo específico Gusto por construir cosas Desea ser un ingeniero de software y le gustaría la experiencia de liderear y diseñar un proyecto de desarrolloDiseñador competente y siente que puede liderear un equipo de desarrollo Familiarizado con métodos de diseño Desea oír las ideas de diseño de otras personas y puede comparar objetivamente y lógicamente los factores de calidad de sus ideas de diseño con las suyas
Quality/Process Manager (Administrador de Calidad/Proceso)	TE-QPM	Soporta al equipo en definir las necesidades de su proceso, en hacer el plan de calidad, y en el seguimiento de la calidad del proceso y del producto Interesado en la calidad del software

		Interesado en el proceso y en las métricas del proceso Experiencia o interés en métodos para inspección y revisión Deseoso y capaz de constructivamente revisar y comentar acerca del trabajo de otros sin antagonismo
Roles externos al equipo		
project manager (Administrador del Proyecto)	PROMA	Administra los recursos que necesita el equipo: humanos y materiales. Acuerda con el *client* y con *top management.* Tiene la última decisión cuando hay conflictos de cualquier índole.
Client (Cliente)	CLIENT	Utilizará el componente desarrollado o es el actor que solicita el desarrollo. Es el propietario de las necesidades y restricciones, impuestos al desarrollo.
Top management (Administración superior)	TOPMA	Ubica al desarrollo de software en la organización. Representa al grupo de actores que determina el contexto del desarrollo incluyendo los rangos permitidos para todos los parámetros de calidad: del proceso de desarrollo y del

4.4.7.2 Actividades

Se asocian a los objetivos y describen las tareas y roles responsables.

Rol	Descripción	Modo de Trabajo en Grupo Asociado
SP-LAUNCH & STRATEGY (Lanzamiento y estrategia)		
PROMA TE-TM	A1.- Acordar proceso a seguir de acuerdo a las experiencias previas.	MP-COLL
ROMA TE-TM	A2.- Especificar requisitos de los roles y recolección de información sobre miembros del equipo. (MDP-INFO)	MP-COLL
PROMA TE-TM	A3.- Definición de los objetivos del producto a desarrollar y criterios para evaluarlo.	MP-COLL
PROMA TE-TM	A4.- Asignación de roles para el equipo.	MP-COLL
PROMA	A5.- Definición de metas para el equipo y los roles,	MP-COLL

TE-TM	discusión acerca del desempeño y su mejora.	
ROMA TE-TM	A6.- Acordar procedimiento para las reuniones para control del estado del desarrollo.	MP-COLL
TE-TL TE-TM	A7.- Primera reunión del equipo para reportar y controlar el estado del desarrollo.	MP-CONT
TE-PM TE-TM	A8.- Establecer los datos requeridos del equipo y su utilización.	MP-COLL
TE-TM	A9.- Inicio del proyecto.	MP-COLL
PROMA TE-TM	A10.- Establecer procedimiento para obtener la estrategia a seguir.	MP-COLL
TE-DM TE-QPM TE-TM	A11.- Establecer criterios para definir la estrategia.	MP-COLL
TE-DM TE-TM	A12.- Producir el diseño conceptual del producto a desarrollar	MP-COLL
DEVM TE-TM	A13.- Determinar la estrategia para desarrollo	MP-COLL
TE-PM TE-TM	A14.- Producir y actualizar estimados preliminares de tamaño y tiempo requeridos	MP-COLL
	A15.- Identificar y valorar riesgos del proyecto	MP-COLL
TE-QPM	A16.- Documentación y actualización de estrategia seleccionada	MP-COLL
TE-SM TE-TM	A17.- Producción y actualización del plan de administración de la configuración	MP-COLL
SP-PLANNING (Planificación)		
PROMAT E-TM	A1.- Acordar el proceso de planificación a seguir.	MP-COLL
TE-PM TE-TM	A2.- Identificar y actualizar el tamaño de todos los productos. (MDP-SUMS)	MP-COLL
TE-PM TE-TM	A3.- Producir y actualizar lista de tareas planeadas con tiempos estimados. (MDP-TASK)	MP-COLL
TE-PM TE-TM	A4.- Producir y actualizar el itinerario de tareas incluyendo los esfuerzos requeridos por cada tarea. (MDP-SCHEDULE)	MP-COLL

TE-QPM TE-TM	A5.- Producir y actualizar el plan de calidad. (MDP-SUMP, MDP-SUMQ)	MP-COLL
TE-PM TE-TM	A6.- Producir planes individuales. (MDP-TASK, MDP-SCHEDULE)	MP-COLL
TE-PM	A7.- Balancear cargas de trabajo. (MDP-TASK, MDP-SCHEDULULE, MDP-SUMP)	MP-COLL
SP-REQUIREMENTS (Requerimientos, Análisis)		
PROMA	A1.- Acordar proceso para obtener y corregir requerimientos y sus productos.	MP-COLL
TE-DM PROMA	A2.- Revisar el documento que contiene las necesidades, determinando | modificando las funciones a realizar y su utilización.	MP-COLL
TE-DM CLIENT PROMA	A3.- Presentación de preguntas consolidadas a los responsables de áreas involucradas y registro de respuestas.	MP-COLL
TE-DM	A4.- Bosquejo de la Carpeta de Requerimientos y del trabajo requerido para producirla. Identificando y ubicando los cambios a los requerimientos y funciones.	MP-COLL
TE-TL	A5.- Asignación de tareas a los miembros del equipo, estableciendo compromisos de terminación de las tareas.	MP-COLL
TE-TM TE-DM	A6.- Producción y examen (por el mismo productor) de cada porción asignada de la carpeta de requerimientos (MDP-LOGT, MDP-LOGD). Integración produciendo anteproyecto de Carpeta de Requerimientos.	MP-COLL
TE-DM TE-TM	A7.- Producción | actualización y revisión del plan de prueba de aceptación (sistema).	MP-COLL
TE-QPM TE-TM	A8.- Inspección de la carpeta de requerimientos y del plan de prueba de aceptación. (MDP-INS-DET, MDP-INS-SUM, MDP-LOGD)	MP-COOP
TE-DM	A9.- Actualización de cada porción asignada de la carpeta de requerimientos y producción de versión final.	MP-COLL
TE-DM CLIENT PROMA	A10.- Entrega y aprobación de carpeta de requerimientos, corrigiendo problemas detectados. (MDP-SUMP, MDP-SUMQ)	MP-COLL
TE-SM	A11.- Se registra y archiva la carpeta de requerimientos en la líneabase (de la Adm. De Configuración CM)	P-COLL
SP-ANÁLISIS & DESIGN (Análisis y Síntesis)		

PROMA	A1.- Acordar proceso de diseño y sus productos.	MP-COLL
TE-DM	A2.- Diseño de alto nivel, definiendo la estructura del producto, sus componentes y relaciones entre ellos, asociando los casos de uso a dichos componentes. Determinando las tareas de diseño que se realizarán.	MP-COLL
TE-QPM TE-TM	A3.- Producción del glosario de nombres y las normas para diseñar.	MP-COLL
	A4.- Bosquejo de la Carpeta de Diseño y del trabajo requerido para producirla.	P-COLL
TE-DM TE-TM	A5.- Asignación de tareas a los miembros del equipo, estableciendo compromisos de terminación de las tareas.	MP-COLL
TE-TM TE-DM	A6.- Producción y examen (por el mismo productor) de cada porción asignada de la carpeta de diseño (MDP-LOGT, MDP-LOGD). Integración produciendo anteproyecto de Carpeta de Diseño.	P-COLL
TE-DM TE-TM	A7.- Producción | actualización y revisión del plan de prueba de integración.	MP-COLL
TE-QPM TE-TM	A8.- Inspección de la carpeta de diseño y del plan de prueba de integración. (MDP-INS-DET, MDP-INS-SUM, MDP-LOGD)	MP-COOP
TE-DM	A9.- Actualización de cada porción asignada de la carpeta de diseño y producción de versión final. Verificando que sea seguible hacia la carpeta de requerimientos. (MDP-SUMP, MDP-SUMQ)	MP-COLL
TE-SM	A10.- Se registra y archiva la carpeta de diseño en la líneabase (de la Adm. De Configuración CM)	MP-COLL
	SP-IMPLEMENTATION (Implementación)	
PROMA	A1.- Acordar proceso de implementación y sus productos, así como su mejora. Enfatizando la calidad, seguir normas existentes y la estrategia para manejar componentes de baja calidad.	MP-COLL
TE-DM TE-TM	A2.- Determinar la división del trabajo (BDS) y planificar las tareas de implementación (MDP-SUMP, MDP-SUMQ)	MP-COLL
TE-DM TE-TM	A3.- Asignación de tareas a los miembros del equipo, estableciendo compromisos de terminación de las	MP-COLL

	unidades (de trabajo) resultantes.	
TE-TM	A4.- Realizar diseño detallado (MDP-LOGT, MDP-LOGD).	MP-COLL
TE-TM	A5.- Producción del plan de prueba de unidad.	MP-COLL
TE-TM	A6.- Desarrollo de casos, procedimientos y datos de prueba para la unidad.	MP-COLL
TE-QPM TE-TM	A7.- Inspección de la carpeta de diseño detallado y plan de prueba de la unidad. (MDP-INS-DET, MDP-INS-SUM, MDP-LOGD)	MP-COOP
TE-TM	A8.- Desarrolla el producto (si la unidad es un ensamble de software: codifica) (MDP-LOGT, MDP-LOGD)	MP-COLL
TE-QPM TE-TM	A9.- Inspección del producto (de cada unidad). (MDP-INS-DET, MDP-INS-SUM, MDP-LOGD)	MP-COOP
TE-TM	A10.- Efectuar prueba de unidad (MDP-LOGD, MDP-LOGT)	MP-COLL
TE-QPM	A11.- Revisión de cada unidad (ensambles o partes que contiene) para determinar si su calidad cumple con los criterios del equipo. (MDP-INS-DET, MDP-INS-SUM, MDP-LOGD)	MP-COOP
TE-TM TE-SM	A12.- Entrega de unidades (los ensambles o partes) para su registro en el sistema de CM (administración de configuración).	MP-COLL
SP-TEST (Prueba)		
PROMA	A1.- Acordar proceso de prueba de integración y de aceptación así como sus productos, incluyendo la forma de mejorarlos. Enfatizando la necesidad de contar con: componentes de calidad antes de las pruebas, normas para probar y una estrategia para manejo de componentes de baja calidad.	MP-COLL
TE-DM	A2.- Asignación de tareas de prueba a los miembros del equipo, estableciendo compromisos de terminación de las tareas.	MP-COLL
TE-TM	A3.- Realizar tareas preparativas para las pruebas. Definir y revisar los procesos y recursos requeridos para: construcción, integración y aceptación.	MP-COLL
TE-DM TE-TM	A4.- Construcción del producto verificando que esté completo. (MDP-LOGD)	MP-COLL
TE-DM TE-TM	A5.- Realizar tareas para la prueba de integración. (MDP-LOGTEST, MDP-LOGD)	MP-COLL

TE-QPM TE-TM	A6.- Inspección de resultados de prueba de integración. (MDP-INS-DET, MDP-INS-SUM, MDP-LOGD)	MP-COOP
TE-DM TE-TM	A7.- Realizar tareas para la prueba de aceptación. Probando el producto en condiciones normales y de carga fuerte, incluyendo la instalación, conversión y la recuperación. (MDP-LOGTEST, MDP-LOGD)	MP-COLL
TE-DM TE-TM	A8.- Determinar las tareas requeridas para obtener la documentación de usuario. Realización, revisión y entrega de la documentación de usuario.	MP-COLL
TE-QPM TE-TM	A9.- Inspección de resultados de prueba de aceptación y documentación de usuario. (MDP-INS-DET, MDP-INS-SUM, MDP-LOGD)	MP-COOP
SP-POSTMORTEM (Pos-término)		
PROMA	A1.- Acordar proceso de postmortem y sus productos, así como su mejora. Enfatizando la necesidad de datos de calidad completos, el contenido del reporte del ciclo a realizar y el proceso de evaluación de los compañeros.	MP-COLL
TE-QPM	A2.- Análisis de los datos del proyecto identificando áreas problema y posibilidades de mejora. Evaluando eficacia de sugerencias de MDP-PIP previas y las acciones realizadas. (MDP-PIP)	MP-COLL
TE-TL TE-TM	A3.- Evaluación de la eficiencia en el desempeño de los roles de los miembros del equipo y los recursos utilizados.	MP-COLL
TE-TL TE-TM	A4.- Bosquejo del reporte del ciclo. Asignación de tareas obteniendo compromisos de terminación. Ensamble, revisión y corrección del reporte.	MP-COLL
TE-TM	A5.- Evaluación del equipo y de cada miembro del equipo. (MDP-PEER)	MP-COLL

4.4.8 Diagramas

En la figura 4-2 se observa también que en el extremo derecho del prisma, que representa visualmente al conjunto de los siete pasos (una iteración o ciclo), hay tres flechas que representan los caminos posibles cuando se termina una iteración. La flecha horizontal nos lleva a realizar otra iteración en el mismo *PS1Set: PS2Set* para completar o converger a los resultados deseados. Mientras que la flecha que apunta hacia abajo nos lleva a comenzar la primera iteración de algún *PS1Set: PS2Set* posterior. La flecha que apunta hacia arriba nos indica la decisión de regresarse a trabajar en un producto de algún conjunto previo.

En la parte superior de la figura 4-2 se muestran los cinco *PS1Sets* en la categoría de Ingeniería del método AGD, indicando el número de *PS2Sets* que tiene cada una de ellas: **CIM** con 6 *PS2Sets*; **PIM** con 3 *PS2Set*s; **PSM** con 4 *PS2Set*s; **IM** con 3 *PS2Sets*, y **OM** con 4 *PS2Sets*. Cada uno de los *PS1Sets* comienza con la iteración uno de su primer *PS2Set* y del lado derecho se muestra con (...) la posibilidad de tener n iteraciones del mismo *PS2Set*.

La figura 4-2 relaciona a dos elementos de la estructura del método AGD: el Product Set (Fuente Bradley y líneas llenas) y la Estructura de Procesos (Fuente Times New Roman y líneas punteadas). Los *PS1Sets* agrupan a los modelos o *productos* especificados en la categoría de Ingeniería por el método AGD.

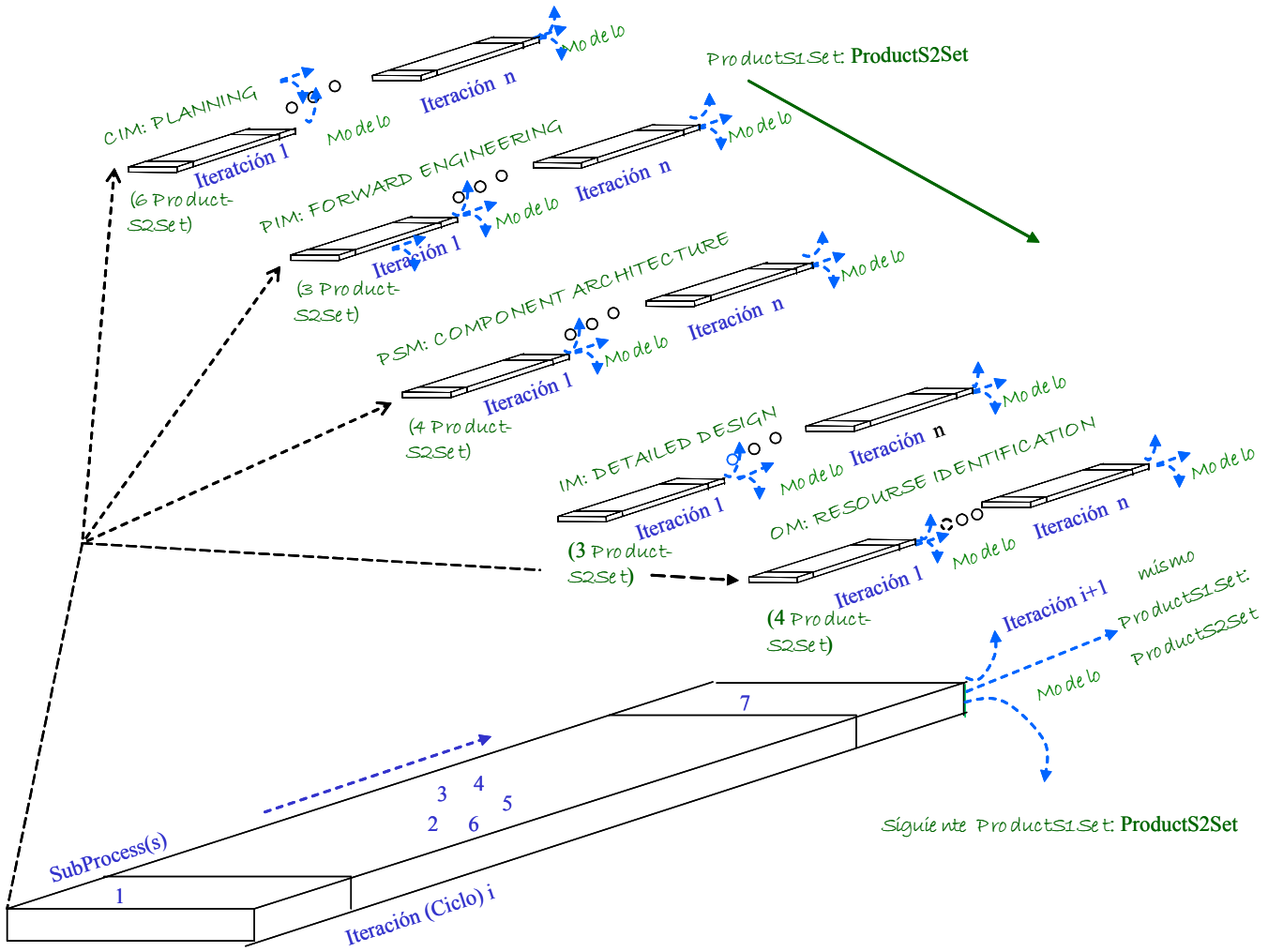


Figura 4-2 SubProcesos (*SubProcess*) para realizar cualquier *PS1Set: PS2Set*

Los sub-procesos (pasos) son parte de la estructura de procesos del MDP:

1. SP-LAUNCH & STRATEGY (Lanzamiento y estrategia)
2. SP-PLANNING (Planificación)
3. SP-REQUIREMENTS (Requerimientos, Análisis)
4. SP-ANÁLISIS & DESIGN (Análisis y Síntesis)
5. SP-IMPLEMENTATION (Implementación)
6. SP-TEST (Prueba)
7. SP-POSTMORTEM (Pos-término)

Estos sub-procesos se muestran en la estructura de una iteración (prisma grande en la parte baja de la figura), donde se muestra que sólo los sub-procesos marcados con el número 1 y 7 tienen un orden fijo de realización. Los sub-procesos marcados con los números 2 al 6 se realizan en el orden que se requiera en cada caso.

Una ventaja de la representación mostrada en la figura 4-2 es que ayuda a decidir (ver figura 4-3) que tipo de modelo de ciclo de vida para el desarrollo se quiere seguir [M94]. Además de acuerdo con las características deseadas, para el componente de software a desarrollar, podemos decidir el número de iteraciones o ciclos: utilizando alguno de los modelos de ciclos de vida para desarrollo de software Incrementales No-monolíticos [G89] siguientes: 1) Construcción y Prueba Incremental; 2) Entrega Evolutiva; 3) Incremental con Marco; 4) Desarrollo por Fases; 5) Prototipo que se Tira, y 6) Prototipo que se Incorpora.

En la parte izquierda de la figura 4-3 mostramos, como ejemplo, la representación diagramática del ciclo de vida Incremental con Marco. Se representa un marco (mostrado como una repisa) que tiene el espacio adecuado para contener a todos los Incrementos (partes: módulos o sub-componentes), es decir que se realizó el diseño general hasta obtener la arquitectura del componente completo (incluyendo todos los requerimientos). Con la arquitectura definida (marco) se procedió a construir y transferir cada incremento, iniciando el incremento siguiente hasta que se transfirió y aceptó el incremento previo.

En la parte derecha de la figura 4-3 mostramos, otro ejemplo, la representación diagramática del ciclo de vida Entrega Evolutiva. En este caso no se obtiene la arquitectura del componente completo, sino que habiendo determinado parcialmente los requerimientos se prosigue con la construcción y transición del primer incremento. En seguida se retoma el levantamiento de otra porción de los requerimientos, evolucionando hacia la recopilación completa. Además de construir y transferir el segundo incremento, probablemente se tenga que modificar el incremento previo debido a la recopilación de requerimientos que no se habían considerado.

En la figura 4-4 todos los miembros del equipo son actores realizando el *rol* común: **team member** (TE-TM) en el que tienen que comportarse como ingeniero de desarrollo eficiente y esforzarse por facilitar el trabajo en grupo.

Cada *rol* del trabajo en equipo toma la responsabilidad de la realización o dirección de algunas actividades, dentro de los *sub-procesos*; y la producción de algunos documentos de control.

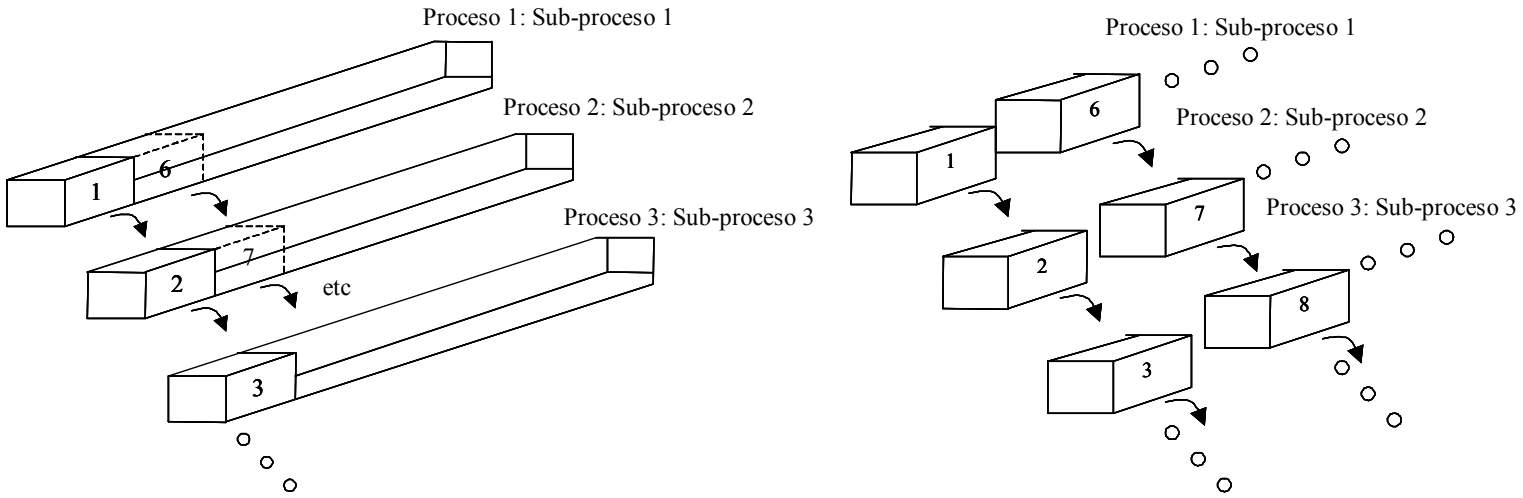


Figura 4-3 Algunos modelos del ciclo de vida Incrementales No-monolíticos

Cuando se trate de desarrollar un programa o módulo pequeño un actor puede realizar todos los *roles*, representando el extremo mínimo del desarrollo. En el extremo máximo tendríamos a varios equipos de diez actores y en todos los grupos cada *rol* (de los cinco posibles) lo desempeñaría una pareja trabajando en modo colaborativo.

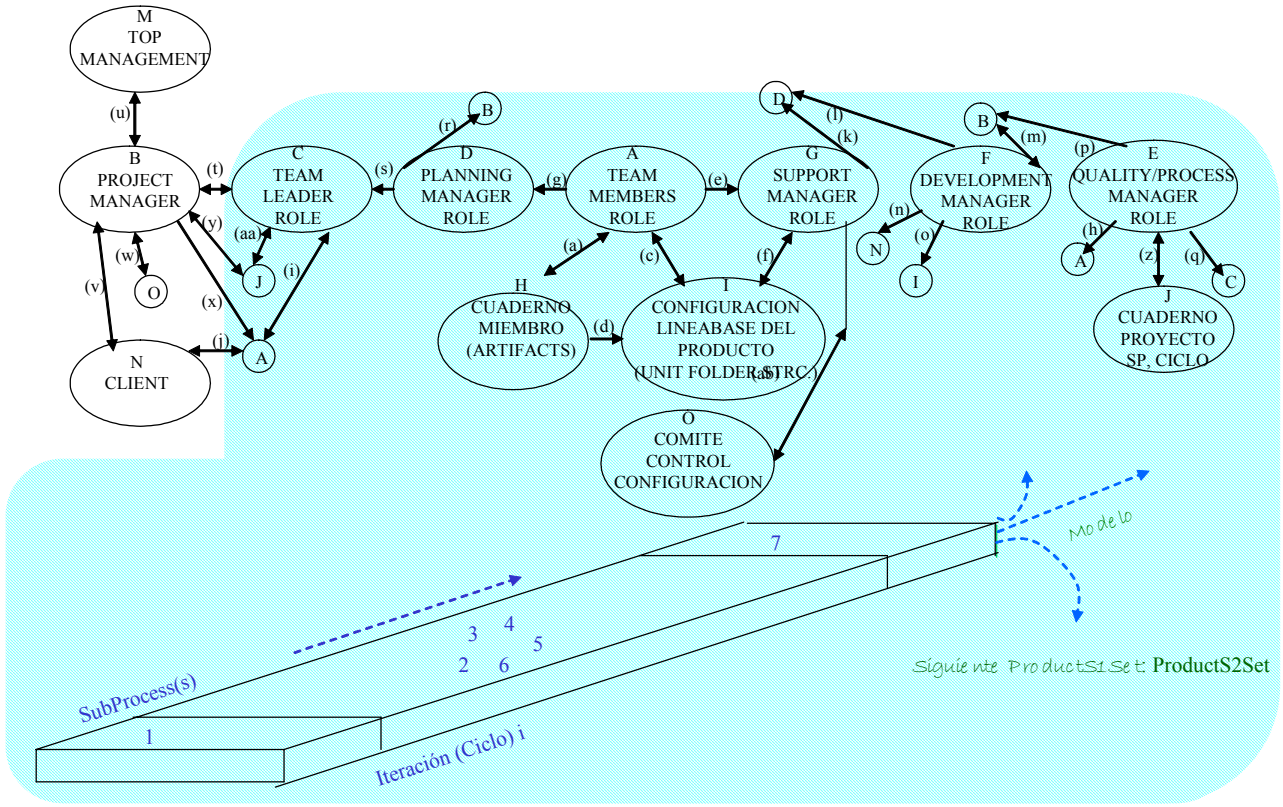


Figura 4-4 Sub-procesos y Roles del Trabajo en Equipo (estática).

En la figura 4-5 observamos los siete pasos o sub-procesos que se realizan para construir un modelo y en general para resolver el problema de obtener un resultado (cualquiera que este sea). Así, dependiendo del grado de avance del desarrollo del componente, se elige en el método AGD el modelo que conviene realizar. Ya con la determinación del modelo a realizar, una pareja de **Team Members** (TE-TM) trabajando en modo colaborativo (estado DESARROLLO) realizará todos los sub-procesos del estado TEAM WORK, hasta terminar la construcción del modelo.

El escenario descrito en el párrafo anterior presupone que la pareja de **Team Member**s (TE-TM) pertenece a un equipo de desarrollo que funciona con los seis roles mostrados en la parte inferior del diagrama de estado 5-5. Si es el caso de un

proyecto que requiera cinco o más personas (actores) para su desarrollo seguramente se habrán repartido los roles y habrá otras parejas o grupos trabajando en otras secciones del mismo modelo o en otros modelos requeridos.

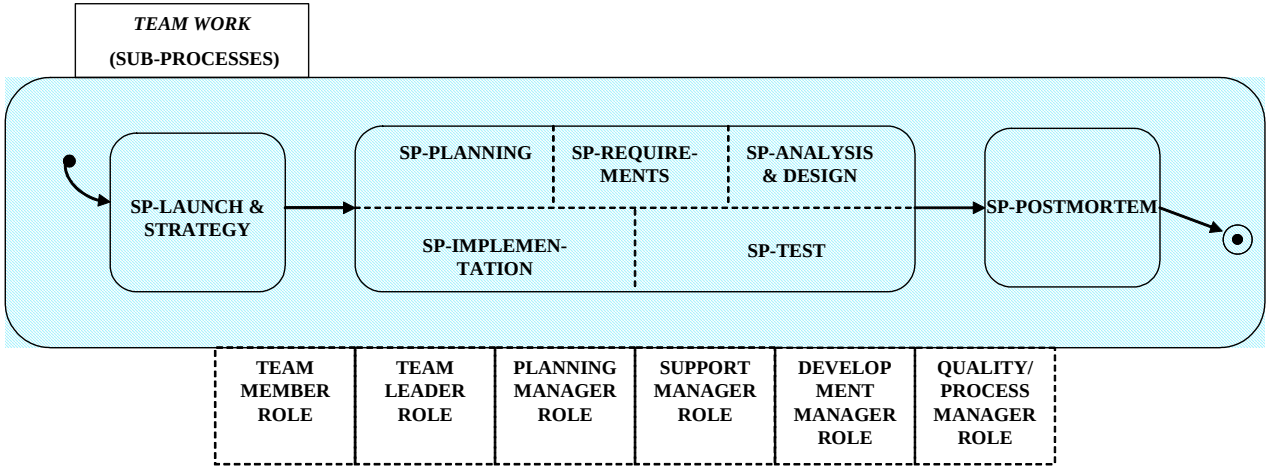


Figura 4-5 Sub-procesos del Trabajo en Equipo (dinámica)

4.4.9 Tabla de Petri para modo de proceso MP-TEAM

NOM-BRE	LUGAR	MARCA POSI-BLE	MARCA HABILITA	TRANSICION	MAR-CA SALI-DA	ACCIO-NES	LUGAR SIGUI-ENTE	ΔT
Inicio del Ci-	Se re-quiere	* C-L	* & C-L &	SP-LAUNCH & STRATEGY	* SP-S		Estrate-gia De-	

clo	producto	C-PS C-WP R-TEAM	C-PS & C-WP & R- TEAM				fini-da	
	Estrate-gia de-finida	* SP-S C-PL	* & (SP-S = SP-PLA-NNING) & C-PL	SP-PLANNING	* SP-S SP-R		Estrate-gia de-finida	
	Estrate-gia de-finida	* SP-S C-RT	* & (SP-S = SP-REQUIRE-MENTS) & C-RT	SP-REQUIRE-MENTS	* SP-S SP-R		Estrate-gia de-finida	
	Estrate-gia de-finida	* SP-S C-DT	* & (SP-S = SP-ANÁLISIS & DESIGN) & C-DT	SP-ANÁLISIS & DESIGN	* SP-S SP-R		Estrate-gia de-finida	
	Estrategia definida	* SP-S C-IM	* & (SP-S = SP-IMPLEME NTATION) & C-IM	SP-IMPLEMEN-TATION	* SP-S SP-R		Estrate-gia de-finida	
	Estrate-gia de-finida	* SP-S C-TT	* & (SP-S = SP- TEST) & C-TT	SP-TEST	* SP-S SP-R		Estrate-gia de-finida	
	Estrate-gia de-finida	* *SP-S* *SP-R* *-PM*	** & (SP-S = SP-POSTMOR TEM) & SP-*	*SP-POSTMOR-TEM*	*		*Produc-to terminado*	

			R = (SP-PLANNING & SP-REQUIRE-MENTS & SP-ANÁLISIS & DESIGN & SP-IMPLEMEN-TATION & SP-TEST) & C-PM					

Tabla de Marcas

Tipo	Etiqueta	*Significado*
	*	*Intención de realizarla*
	C-L	*Conocimiento del Launch.- Se conocen los conceptos y la lógica el trabajo en equipo, se entiende el sub-proceso de Launch (lanzamiento).*
	C-PS	*Conocimiento del proceso de software.- Se conocen los conceptos del proceso personal de software.*
	-WP	*Conocimiento del producto-del-trabajo a desarrollar.- Se conoce su estructura, funcionalidad, etc.*
	C-ST	*Conocimiento para desarrollar estrategia.- Estrategia, diseño conceptual, manejo de riesgos.*
	R-TEAM	*Recurso equipo de desarrollo disponible*

	SP-S	*Sub-proceso (s) siguiente(s): (SP-PLANNING y/o SP-REQUIREMENTS y/o SP-ANALISYS & DESIGN y/o SP-IMPLEMENTATION y/o SP-TEST).*
	SP-R	*Sub-proceso(s) realizado(s):): (SP-PLANNING y/o SP-REQUIREMENTS y/o SP-ANALISYS & DESIGN y/o SP-IMPLEMENTATION y/o SP-TEST).*
	C-PL	*Conocimiento acerca de cómo planificar.- Conocer el proceso de planificar, la forma de seguir los logros del trabajo, la planificación de la calidad.*
	C-RT	*Conocimiento acerca de definir requerimientos y planear pruebas*
	C-DT	*Conocimiento acerca de diseñar en equipo.*
	C-IM	*Conocimiento acerca de implementar en equipo.*
	C-TT	*Conocimiento acerca de proceso para pruebas de integración y de aceptación.*
	C-PM	*Conocimiento acerca del proceso para Postmortem y del trabajo en equipo.*

4.4.10 Verificaciones y Validaciones

Verificación o validación	Activi-dad	Producto	Rol	Descripción
SP-REQUIREMENTS (Requerimientos, Análisis)	A8	carpeta de requerimientos y del plan de prueba de aceptación	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (Validación asociada, con el cliente)
SP-ANÁLISIS & DESIGN (Análisis y Síntesis)	A8	carpeta de diseño y del plan de prueba de integración	TE-DM TE-QPM CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD

SP-IMPLEMENTATION (Implementación)	A7	diseño detallado y plan de prueba de la unidad	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD
	A9	fólder de unidad	TE-DM -QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (verificación)
	A11	fólder de unidad	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (validación)
SP-TEST (Prueba)	A6	resultados de prueba de integración	TE-DM TE-QPM TE-TL	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (verificación)
	A9	Resultados de prueba de aceptación y documentación de usuario	TE-DM TE-QPM TE-TL	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (validación)

4.4.11 Incorporación a la base de conocimiento

Producto	Forma de Aprobación
Carpeta de requerimientos y del plan de prueba de aceptación	SP-REQUIREMENTS (Requerimientos, Análisis) - A8
Carpeta de diseño y del plan de prueba de integración	SP-ANÁLISIS & DESIGN (Análisis y Síntesis) - A8
Diseño detallado y plan de prueba de la unidad	SP-IMPLEMENTATION (Implementación) - A7
Fólder de unidad	SP-IMPLEMENTATION - A9
Resultados de prueba de integración	SP-TEST (Prueba) - A6
Resultados de prueba de aceptación y documentación de usuario	SP-TEST (Prueba) - A9

4.5Dinámica de los Modos de Trabajo-en-Grupo

El proceso MDP es incremental, cuando los desarrolladores elaboran su estrategia y planificación para obtener conjuntos de productos, agregando detalles en cada ciclo. Esta estrategia puede fluctuar entre dos casos extremos: a) producir un producto pequeño en una sola iteración (todos los modelos del Conjunto de Productos / Product Set (PS) se realizan en un intervalo de tiempo pequeño y b) producir un producto grande realizable en varias iteraciones (ciclos), cada una realizada en un tiempo corto.

Las reglas básicas para realizar las sesiones de trabajo, seguidas en los modos de trabajo Colaborativo, Cooperativo y de Control son las siguientes:

- La sesión de trabajo debe tener duración con límite de tiempo (se recomienda dos horas).
- Se asigna un horario de bloque fijo, para realizar sesiones, para cada modo de trabajo y sin traslapes. Se tendrá un lugar preestablecido para la sesión de cada modo.
- Las sesiones de los modos control / **control** y cooperativo / **cooperative** deben tener la prioridad más alta en los planes de trabajo. Sin excusa deben empezar a tiempo y los participantes permanecer hasta terminar.
- Durante la sesión se evitan interrupciones.
- Todos los miembros del equipo son profesionales honestos y bien intencionados.
- Cualquier miembro del grupo de trabajo puede señalar violaciones a las reglas básicas.

Presentamos aquí el comportamiento en el tiempo del MDP (ej. cuando se termina un modelo, se solicita una sesión de revisión —hecha en modo cooperativo—). Utilizamos un diagrama de estado / State Chart, mostrado en la figura 4-6; en el que se muestra la realización concurrente (durante el mismo lapso de tiempo, en forma coordinada) de los modos de trabajo-en-grupo: en Equipo, Colaborativo, Cooperativo y de Control. La concurrencia de los modos de

trabajos se representa, en este tipo de diagramas, dividiendo el estado en sub-estados (separados por líneas punteadas).

En la figura notamos la existencia de un proyecto y dentro del mismo la coexistencia de dos sub-estados concurrentes: la Administración de Proyecto y el Desarrollo en Equipo. El Desarrollo en Equipo utiliza los cuatro modos de trabajo-en-grupo dentro del contexto común de un grupo de desarrollo.

La separación de la administración del proyecto y del equipo de desarrollo, sugiere informar al administrador el estado y la repercusión de los trabajos realizados en modo colaborativo, cooperativo y de control (sin incluir los detalles técnicos).

Se informa, a la administración de proyecto, una selección de los datos detallados acerca de la productividad y número de defectos, situándolos en contexto. Evitando de esta manera que los informes técnicos se utilicen para evaluación del personal. Pues si se utilizan para realizar ajustes administrativos (recompensar o corregir comportamiento), seguramente habrá renuencia para reflejar deficiencias y la credibilidad será nula.

En la figura 4-6 observamos que el rol de Administrador del Proyecto / **Project Manager** está continuamente realizando tres sub-procesos concurrentes: 1) administrando los recursos materiales y humanos, 2) evaluando el desempeño del equipo de desarrollo y 3) tomando decisiones que minimicen el uso de recursos y los riesgos; asegurando que el proceso MDP obtenga los resultados deseados.

Al término del proyecto, se organizan e incluyen los documentos que reportan las métricas[29] de desempeño y los pormenores técnicos del desarrollo, en la base de resultados de los proyectos de la organización. De tal manera que se pueda usar esta información para estimaciones en proyectos futuros.

[29] Métrica.- Medida cuantitativa del grado en que un sistema, componente o proceso, posee un atributo dado [IEEE 61012].

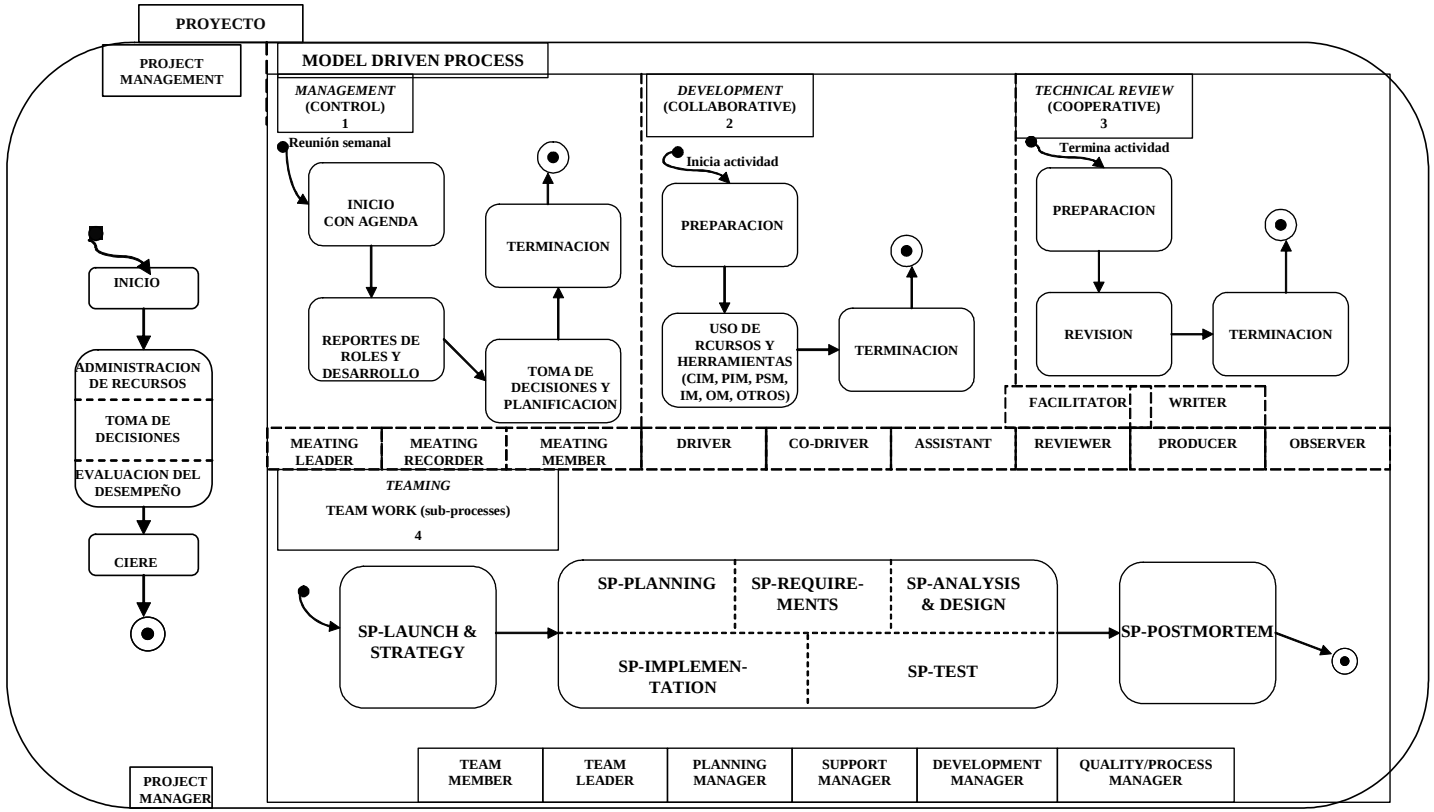


Figura 4-6 Carta de estados de un proyecto que utiliza el MDP.

La participación del administrador del proyecto es muy importante en algunas de las reuniones de control del avance del proceso, sobre todo al inicio del mismo cuando se define el problema a resolver, la estructura del equipo, así como los lineamientos y forma de trabajo requeridos.

El detalle de cada sub-estado del estado "PROYECTO" (figura 4-6), consultarlo en los patrones de proceso de cada uno de los modos de trabajo en grupo, que incluimos en la sección 5.4 y el anexo A.

4.6Conclusión

El proceso se dirige por modelos de las secuencias de productos del PS con la iteración sistemática del Proceso Dirigido por Modelos / Model Driven Process (MDP). Cada ciclo del MDP se planifica para obtener un conjunto de productos específico. De esta forma se integra incrementalmente el producto de software, objetivo del proceso.

MDP busca la sinergia de cuatro modos de trabajo-en-grupo y la realización de cualquier producto mediante iteraciones de los subprocesos del modo en Equipo.

El grupo de desarrollo sigue el Procesos Dirigido por Modelos, trabajando en cuatro modos posibles de trabajo-en-grupo:

- **Colaborativo / Collaborative**.- protocolo para desarrollar, en grupo de dos o más participantes, los productos del PS o los requeridos mientras se realizan sub-procesos de cualquier otro modo de trabajo-en-grupo, compartiendo la responsabilidad del resultado.
- **Cooperativo / Cooperative**.- protocolo para las revisiones técnicas de los modelos o productos, que maneja conflictos en la toma de decisiones de aceptación o de rechazo del producto.
- **Control / Control**.- protocolo para reportar el desarrollo técnico efectuado y los resultados obtenidos en el ejercicio de las funciones administrativas asignadas (roles), proponer mejoras del proceso, administrar riesgos, así como planificar las tareas siguientes.
- **En Equipo / Team Work**.- protocolo para establecer la dinámica de un equipo de desarrollo.

Todo modelo o producto-del-trabajo, se desarrolla en equipo, mediante ciclos de siete subprocesos, organizados mediante cinco *roles,* además del rol de desarrollador o miembro del equipo.

Las vistas estática y dinámica del proceso MDP describen un proceso sistemático útil para desarrollar cualquiera de los productos del PS. MDP tiene un sólo nivel de proceso, lo que se refleja en la capacitación de los participantes en solamente cuatro modos de trabajo-en-grupo. Otros modelos, como SPEMS y RUP manejan dos niveles de proceso.

Los modelos para evaluación de la capacidad de desarrollo de software, como CMMI, manejan más de veinte áreas de proceso. En el caso del AGD, los productos de dichas áreas se ubicaron en el PS y en las formas de captura de información acerca del desempeño del equipo. Realizándose una instancia del proceso MDP para obtener dichos productos.

La integración de los modos de trabajo-en-grupo, toma como base al modo de trabajo en Equipo, con sus siete subprocesos. Los sub-procesos se realizan en el orden que definen los participantes durante el primero: SP-Launch&Strategy. De esta forma se integra un conjunto mínimo de sub-procesos, flexible y dinámico, que se realiza concurrentemente (simultáneos con eventos coincidentes), con los otros tres modos de trabajo.

*4.7*Relaciones entre roles, en el MDP (Anexo para figura 4-1)

(a) -TEAM MEMBER <-> MEMBER NOTEBOOK
Technical information, artifacts, filled forms, received forms, time management

(b) -TEAM MEMBER (i) -> COLLABORATIVE WORK
Decisión tomada producto
Decisión tomada meta

SP-LAUNCH&STRATEGY
MDP-INFO
SP-REQUIREMENTS
Produce y revisa su parte del documento SRS
SP-ANALYSIS&DESIGN
Produce y revisa su parte del documento SDS
SP-INPLEMENTATION
Revisión de diseño
MDP-LOGD, MDP-LOGT
Plan de prueba de unidad
Casos de prueba, procedimientos de prueba, datos de prueba y resultados de prueba
Código fuente del componente
Revisión de código con lista de verificación
Compilar y componer el código hasta libre errores
Formas MDP-LOGD, MDP-LOGT
Formas MDP-LOGD, MDP-LOGT de prueba unitaria
SP-TEST
Procesos y procedimientos de BUILD
Procesos y recursos para desarrollo de prueba de integración
Procesos y recursos para desarrollo de prueba de sistema
Mide tamaño y tiempo de ejecución par cada prueba
Revisa material de prueba y corrige errores
Pruducto BUILD: todas las partes necesarias están a la mano
Pruducto BUILD: construir producto, proveerlo a prueba integración
Pruducto BUILD: propietario de producto registra los defectos en bitácora MDP-LOGD

SP-POSTMORTEM
Forma CWPEER evaluando a cada rol

Dificultad y contribución de cada rol
Porcentajes sumando 100%
Eficacia de cada rol en escala 1(inadecuada) a 5 (superior)
-COLLABORATIVE WORK : TEAM MEMBER (i)
Desacuerdo con opinión sobre producto
Desacuerdo con opinión sobre meta

SP-LAUNCH&STRATEGY
Rol

(c) -TEAM MEMBER -> CONFIGURATION PRODUCT BASELINE (UNIT FOLDER STRC.)
Artefacto, MDP-INS, MDP-LOGD, MDP-CCR
Navega la estructura de UDF
-CONFIGURATION PRODUCT BASELINE (UNIT FOLDER STRC.) -> TEAM MEMBER
Elementos de UDF

(d) -MEMBER NOTEBOOL (ARTIFACTS) -> CONFIGURATION PRODUCT BASELINE (UNIT FOLDER STRC.)
Inspected, corrected artifact

(e) -TEAM MEMBER -> SUPPORT MANAGER
Notifica artefacto se inspeccionó, corrigió

(f) -SUPPORT MANAGER <-> CONFIGURATION PRODUCT BASELINE (UNIT FOLDER STRC.)
incluye artefacto en línea base
Excluye artefacto de línea base
Solicita inconsistencias
Navega la estructura de UDF
- CONFIGURATION PRODUCT BASELINE (UNIT FOLDER STRC.) -> SUPPORT MANAGER
Inclusión artefacto en línea base
Exclusión artefacto de línea base
Inconsistencias
Estructura de UDF

(g) -TEAM MEMBER -> PLANNING MANAGER
MDP-TASK, MDP-SCHEDULE, MDP-WEEK, MDP-PEER

(h) -QUALITY/PROCESS LEADER -> TEAM MEMBER

criterios de estrategia

(i) -TEAM MEMBER -> TEAM LEADER
Temas para reunión semanal
Problema desempeño
Sección reporte de ciclo
-TEAM LEADER -> TEAM MEMBER
Pide temas para reunión semanal
Agenda reunión semanal
Reporte reunión semanal
Sugerencia a problema desempeño
Reporte de ciclo

(j) -TEAM MEMBER -> CLIENT
Preguntas de detalle
-CLIENT -> TEAM MEMBER
Respuestas de detalle

(k) – SUPORT MANAGER -> PLANNING MANAGER
Datos semanales de CM, cambios, riesgos, comentarios

(l) –DEVELOPMENT MANAGER -> PLANNING MANAGER
Datos semanales producción

(m) –DEVELOPMENT MANAGER -> PROJECT MANAGER
SRS
–PROJECT MANAGER -> DEVELOPMENT MANAGER
SRS con Vo.Bo.

(n) - DEVELOPMENT MANAGER -> CLIENT
SRS
-CLIENT -> DEVELOPMENT MANAGER
SRS con Vo.Bo.
(o) –DEVELOPMENT MANAGER -> CONFIGURATION PRODUCT BASELINE
Artefacto, MDP-INS, MDP-LOGD, MDP-CCR
Navega la estructura de UDF
- CONFIGURATION PRODUCT BASELINE -> DEVELOPMENT MANAGER
Artefactos en estructura de UDF

(p) -QUALITY/PROCESS LEADER -> PROJECT MANAGER
Criterios de estrategia
Problemas de calidad

(q) - QUALITY/PROCESS MANAGER -> TEAM LEADER
Asuntos de reunión semanal

Problemas de calidad

(r) –PLANNING MANAGER -> PROJECT MANAGER
Solicita ayuda entreguen a tiempo morosos
Reporta estado personal y consolidado

(s) –PLANNING MANAGER -> TEAM LEADER
Solicita ayuda entreguen a tiempo morosos

(t) – TEAM LEADER -> PROJECT MANAGER
+Reporte Desempeño Del Equipo
Reporte Reunión Equipo Semanal
Actualizados MDP-TASK, MDP-SCHEDULE, MDP-WEEK
del equipo e ingeniero, y MDP-CSR
Actualizada bitácora MDP-ITL
Formas MDP-INS, MDP-LOGD de inspecciones
Indicaciones sobre morosos
-PROJECT MANAGER -> TEAM LEADER
Vo.Bo. y modificaciones a:
+Reporte Desempeño Del Equipo
Indicaciones para el equipo
Directivas, restricciones
-PROJECT MANAGER -> TOP MANAGEMENT
Reporte de ciclo

(v) -PROJECT MANAGER -> CLIENT
Pide objetivos iniciales del producto
Reporte De Ciclo
-CLIENT -> PROJECT MANAGER
Objetivos iniciales del producto

(w) - CONFIGURATION CONTROL BOARD -> PROJECT MANAGER
CCR: Configuration Change Request
Dictaminadas
–PROJECT MANAGER -> CONFIGURATION CONTROL BOARD
VoBo de CCR dictaminadas
y modificaciones

(x) –PROJECT MANAGER -> TEAM MEMBER
Ayuda y motivación a morosos

(y) -PROJECT MANAGER -> PROJECT NOTEBOOK

SP-TEAM LAUNCH (R)
Directrices para lanzamiento
Lo que se espera de cada team member
Forma de evaluación
Proceso a seguir
Forma MDP-INFO, cada team member
Objetivos iniciales críticos del producto
Objetivos iniciales opcionales del producto
Criterios iniciales para evaluar producto terminado
Asignaciones de equipo y de rol
Restricciones, programación de reuniones
Forma de reporte semanal
Requerimientos de datos semanales

Problemas en proceso y sus causas
Pasos para minimizar problemas

(z) –QUALITY/PROCESS MANAGER -> PROJECT NOTEBOOK
Resultados de reuniones de equipo (SEMANAL)
Reporte Desempeño Del Equipo
Reporte reunión semanal
Actualiza MDP-TASK, MDP-SCHEDULE, MDP-WEEK del equipo e ingeniero, y MDP-CSR
Actualiza bitácora MDP-ITL
Formas MDP-INS, MDP-LOGD de inspecciones
SP-LAUNCH&STRATEGY
MDP-SUMP, MDP-SUMQ, MDP-SUMS, MDP-LOGD, MDP-LOGT
SP-PLAN
MDP-SUMP, MDP-SUMQ, MDP-SUMS, MDP-LOGD, MDP-LOGT
SP-REQUIREMENTS
MDP-SUMP, MDP-SUMQ, MDP-SUMS, MDP-LOGD, MDP-LOGT
SP-ANALYSIS&DESIGN
MDP-SUMP, MDP-SUMQ, MDP-SUMS, MDP-LOGD, MDP-LOGT
SP-IMPLEMENTATION
MDP-SUMP, MDP-SUMQ, MDP-SUMS, MDP-LOGD, MDP-LOGT

(aa) –TEAM LEADER <-> PROJECT NOTEBOOK
Asegura:
Resultados de reuniones de equipo
Reporte Desempeño Del Equipo
Reporte reunión semanal
Formas actualizadas MDP-TASK, MDP-SCHEDULE, MDP-WEEK del equipo e ingeniero, y MDP-CSR
Bitácora MDP-ITL actualizada

Formas MDP-INS, MDP-LOGD de inspecciones

(ab) –SUPPORT MANAGER <-> CONFIGURATION CONTROL BOARD
CCR, artefacto terminado, anexos
CCR proponiendo cambios
–CONFIGURATION CONTROL BOARD <-> SUPPORT MANAGER
CCR: Configuration Change Request dictaminadas

(a-d) - COLLABORATIVE WORK -> CLIENT SP-REQUIREMENTS
Funciones a realizar por diferentes versiones del producto
Como se usan las funciones
Preguntas consolidadas
-CLIENT -> COLLABORATIVE WORK
SP-REQUIREMENTS
Respuesta a preguntas consolidadas

(b-d) -PROJECT MANAGER-> COLLABORATIVE WORK
SP- LAUNCH&STRATEGY
Directrices para lanzamiento
Lo que se espera de cada team member
Forma de evaluación
Pide definir proceso a seguir
Pide forma MDP-INFO, cada team member
Objetivos iniciales críticos del producto
Objetivos iniciales opcionales del producto
Criterios iniciales para evaluar producto terminado
Asignaciones de equipo y de rol
Restricciones, programación de reuniones
Forma de reporte semanal
Requerimientos de datos semanales
Problemas en proceso y sus causas
Sugiere pasos para minimizar problemas

SP-LAUNCH&STRATEGY
Estrategia como se produce y utiliza
Criterios para estrategia efectiva
Formas de producir estimaciones de tamaño y tiempo
Cuando son apropiados los cambios y cuando evitarlos
Síntomas técnicos y del negocio de problemas en estrategia
Como determinar el tamaño del siguiente ciclo

SP-PLAN

Describe planes de tarea y secuencias y su producción
Describe el plan de calidad y su producción
Problemas con planes anteriores y como mejorarlos

SP-REQUIREMENTS
Describe proceso de requerimientos
Como realizar inspección de requerimientos

SP-ANALYSIS&DESIGN
Describe el proceso de diseño con una SDS ejemplo
Describe la inspección de diseño y su reporte
Normas y convenciones de diseño

SP-IMPLEMENTATION
Importancia de implementar con calidad
Necesidad y contenido de norma de codificación
Estrategia para manejar componentes de calidad pobre

SP-TEST
Describe necesidad de componentes de calidad antes de prueba
Necesidad y contenido de normas de prueba
Estrategia para componentes de calidad pobre

SP-POSTMORTEM
Describe necesidad de contar con datos de proceso completos y exactos
Describe contenido del reporte de ciclo
Describe las formas y proceso de evaluación de colegas

-COLLABORATIVE WORK -> PROJECT MANAGER
SP-TEAM LAUNCH (R)
Retroalimentación de lanzamiento
Proceso acordado
Forma MDP-INFO cada team member
Razones para cambiar rol
Lecciones aprendidas de roles
Problemas en desempeño de roles y como manejarlos

(c-d) –TEAM LEADER: COLLABORATIVE WORK
SP-LAUNCH&STRATEGY
Dirige primera reunión de equipo
SP-STRAT
SP-PLAN
SP-REQUIREMENTS
Pide compromisos término tareas
SP-ANALYSIS&DESIGN

Pide compromisos término tareas
SP-IMPLEMENTATION
pide compromisos término tareas
SP-TEST
Ayuda ubicar el desarrollo de la prueba y tareas de prueba entre miembros del equipo (equipo de prueba)
SP-POSTMORTEM
Dirige evaluación de efectividad de roles, acciones del administrador del proyecto e instalaciones
Dirige el bosquejo del reporte del ciclo
Compromisos para terminar tareas (reporte ciclo)

-COLLABORATIVE WORK -> TEAM LEADER
SP-REQUIREMENTS
Compromisos para terminar tareas (SP-REQUERIMIENTOS)
SP-ANALYSIS&DESIGN
Compromisos para terminar tareas (SP-analysis&design)
SP-IMPLEMENTATION
Compromisos para terminar tareas (SP-IMPLEMENTATION)
SP-TEST
Ubica y compromete el desarrollo de pruebas y tareas de prueba
Ubica y compromete el desarrollo de documentación
SP-POSTMORTEM
Roles: en que fueron efectivos
Roles: en que se puede mejorar
Ubica trabajo del reporte a miembros
Obtiene compromisos de terminación de secciones del reporte
Ensamble, revisión y corrección reporte completo
Formas MDP-PEER todos los miembros

(d-d) -PLANNING MANAGER -> COLLABORATIVE WORK
SP-LAUNCH&STRATEGY
Obtiene acuerdo de datos a proveer semanalmente
Estimaciones gruesas de los productos de ciclos subsecuentes
SP-PLAN:
Dirige identificación de productos del proyecto y tamaños
Dirige producción de lista de tareas
Dirige estimación de horas tarea
Dirige los planes individuales y de equipo de CW_TASK, MDP-SCHEDULE
Produce plan preliminar del equipo
Ayuda a cada miembro realizar su plan personal
Produce la secuencia de valor planeado y las fechas de terminación de tareas

Dirige el balance de cargas
Produce los planes de equipo e individuales finales
-COLLABORATIVE WORK -> PLANNING MANAGER
SP-PLAN:
Documentos de planeación (R)
Identifica otros productos y sus tamaños
Registra MDP-STRAT y tamaños en MDP-SUMS
Lista de tareas con estimaciones de equipo e ingeniero
Horas estimadas de cada miembro en proyecto/semana
Formas MDP-TASK y MDP-SCHEDULE para cada ingeniero
Tareas y estimaciones en MDP-TASK
Produce secuencia planeada
Horas semanales en CW_SCHEDULE
Produce las formas MDP-TASK y MDP-SCHEDULE del equipo
Plan, cada miembro
Ubicar tareas entre miembros del equipo
Estimación de tiempo para realizar cada tarea
Incluir datos en formas MDP-TASK y MDP-SCHEDULE
Balancea cargas de trabajo del equipo
Identifica des-balances de cargas de trabajo
Reubicar tareas para minimizar la secuencia de tareas
Plan del ingeniero balanceado
Plan consolidado del equipo (TASK, SCHEDULE, SUMP Y SUMQ)

(e-d) -SUPPORT MANAGER -> COLLABORATIVE WORK
SP-LAUNCH&STRATEGY
Dirige identificación y evaluación riesgos
Incluye riesgos en MDP-ITL
Define el proceso de control de configuración
SP-REQUIREMENTS
Incluir en línea base la SRS final
SP-ANALYSIS&DESIGN
Incluir en línea base la SDS final
SP-IMPLEMENTATION
Incluir componentes correctos a linea base

-COLLABORATIVE WORK -> SUPPORT MANAGER
SP-LAUNCH&STRATEGY
Plan de Admon. Configuración
Configuration control board and procedures
Herramientas de soporte necesarias

(f-d) -(DEVELOPMENT MANAGER -> COLLABORATIVE WORK)
SP-LAUNCH&STRATEGY:
Dirige discusión criterios de estrategia

Dirige producción de estrategia de producción
Dirige estimaciones preliminares de tamaño y tiempo
Dirige identificación y evaluación de riesgos del proyecto
SP-PLANNING
SP-REQUIREMENTS:
Preguntas consolidadas (necesidades)
Descripción documento SRS y el trabajo requerido para desarrollarlo
Dirige producción SRS
Problemas en SRS
Dirige producción plan de pruebas de sistema
SP-ANALYSIS&DESIGN
Descripción documento SDS y el trabajo requerido para desarrollarlo
Dirige producción SDS
Problemas en SDS
Dirige producción plan de prueba
SP-IMPLEMENTATION
Dirige planeación del trabajo de implementación
SP-TEST
Dirige desarrollo de material de pruebas
Dirige tareas de construcción del producto
Dirige tareas de prueba de integración
Dirige tareas de prueba de sistema
Dirige tareas de documentación (usuario)
-COLLABORATIVE WORK -> DEVELOPMENT MANAGER
SP-LAUNCH&STRATEGY:
Criterios de estrategia
Diseño conceptual
SP-LAUNCH&STRATEGY: estrategia completa
Estimaciones de tamaño y tiempo todos elementos próximo ciclo
Estimaciones todos elementos ciclos subsecuentes
Procedimiento administración configuración
Riesgos y comentarios en bitácora MDP-ITL
SP-REQUIREMENTS
Respuestas a preguntas consolidadas
Revisión de documento de necesidades
Funciones a realizar por versiones diferentes
Utilización de funciones
Plan de pruebas del sistema
Secciones SRS modificadas después inspección
SRS modificada final
Verificación SRS seguible a documento de necesidades
SP-ANALYSIS&DESIGN

Estructura del producto para el ciclo
Nombrado de componentes
Ubicación de casos de uso a componentes
Identificar las tareas de diseño a realizar
Plan de pruebas de integración
Secciones SDS modificadas después inspección
SDS modificada final
Verificación de SDS seguible a SRS
SP-IMPLEMENTATION
Definición y planeación tareas formas MDP-SUMP Y MDP-SUMQ
Ubicación de tareas
SP-TEST
Integración: comprueba completes, prueba integración el producto
Integración: registra todas actividades en bitácora pruebas (MDP-LOGTEST)
Integración: productor registra todos defectos en bitácora (MDP-LOGD)
Sistema: prueba producto en condiciones normales y de esfuerzo
Sistema: prueba producto en instalación, conversión , recuperación
Sistema: prueba de regresión
Sistema: registra todas actividades en bitácora pruebas (MDP-LOGTEST)
Sistema: productor registra todos defectos en bitácora (MDP-LOGD)
Documentación: bosquejo de documentación y tareas
Documentación: ubicación de éstas áreas a equipo documentación
Documentación: revisar completes en bosquejo de documentación con equipo de pruebas
Documentación: borrador documentación de usuario
Documentación: revisión, corrección y producción de documentación de usuario

(g-d) -(QUALITY/PROCESS LEADER -> COLLABORATIVE WORK)
SP-PLAN
Dirige plan calidad
SP-IMPLEMENTATION
Revisión de datos del componente
SP-POSTMORTEM
Dirige análisis de datos del proyecto e identificación de problemas y áreas de mejora

-COLLABORATIVE WORK -> QUALITY/PROCESS LEADER
SP-LAUNCH&STRATEGY
Criterios de estrategia
Diseño conceptual
MDP-STRAT: estrategia completa
Estimaciones de tamaño y tiempo todos elementos próximo ciclo
Estimaciones todos elementos ciclos subsecuentes
Procedimiento administración configuración
Riesgos y comentarios en bitácora MDP-ITL
SP-PLAN
Objetivos de calidad del equipo

Estimación defectos inyectados y yield defectos eliminados
Planes MDP-SUMP y MDP-SUMQ
SP-ANALYSIS&DESIGN
Glosario de nombres y normas de diseño
SP-IMPLEMENTATION
Resultado revisión datos de componente
SP-POSTMORTEM
Formas MDP-PIP con hallazgos de liderazgo, planeación, proceso, calidad o soporte
Formas MDP-PIP con acciones de equipo sugeridas y responsabilidades
Formas MDP-PIP con áreas de mejora para administración de proyecto e instalaciones

(a-t) –CLIENT -> COOPERATIVE WORK
INS-MILESTONE
Inquietudes y dudas
-COOPERATIVE WORK -> CLIENT
INS-MILESTONE
Respuestas técnicas

(b-t) -(TEAM MEMBER -> COOPERATIVE WORK)
+(WORK FLOWS, PHASE) (COMO PRODUCER)
SP-INPLEMENTATION
Diseño detallado
Plan de prueba de unidad
Casos de prueba, procedimientos de prueba, datos de prueba
Código fuente del componente
Resultados prueba de unidad

SP-TEST
Procesos y procedimientos de BUILD
Procesos y recursos para desarrollo de prueba de integración
Procesos y recursos para desarrollo de prueba de sistema
Resultados prueba integración
Resultados prueba sistema condiciones normales y de esfuerzo
Resultados prueba sistema instalación, conversión y recuperación
Documentación de usuario final

-COOPERATIVE WORK -> TEAM MEMBER
INSSUM, MDP-INS, MDP-ITL, MDP-LOGD MDP-SUMP, MDP-SUMQ, MDP-LOGD, MDP-LOGT

(c-t) -QUALITY/PROCESS LEADER :COOPERATIVE WORK
SP-REQUIREMENTS

Organiza inspección de SRS y plan de pruebas del sistema

SP-ANALYSIS&DESIGN
Organiza inspección de SDS y plan de pruebas de integración
(Se cubre y hace referencia a todo caso de uso, el diseño es completo y correcto el plan de prueba de integración es adecuado)

SP-IMPLEMENTATION
Organiza inspección
Diseño detallado de cada componente
Plan de prueba de unidad
casos de prueba, procedimientos de prueba, datos de prueba
Código de cada componente

SP-TEST
Organiza inspección
Procesos y procedimientos de BUILD
Procesos y recursos para desarrollo de prueba de integración
Procesos y recursos para desarrollo de prueba de sistema
Resultados prueba integración
Resultados prueba sistema condiciones normales y de esfuerzo
Resultados prueba sistema instalación, conversión y recuperación
Documentación de usuario final

SP-POSTMORTEM
Dirige análisis de datos del proyecto e identificación de problemas y áreas de mejora

-COOPERATIVE WORK -> QUALITY/PROCESS LEADER
SP-REQUIREMENTS
Preguntas y problemas
Quien resolverá cada pregunta y problema
Formas: MDP-INSSUM, MDP-INS, MDP-ITL, MDP-LOGD, MDP-SUMP, MDP-SUMQ

SP-ANALYSIS&DESIGN
Comentarios y problemas en MDP-LOGD, MDP-ITL
Quien es responsable resolver cada problema
Formas MDP-INSSUM, MDP-INS, MDP-ITL, MDP-LOGD, MDP-SUMP, MDP-SUMQ

SP-IMPLEMENTATION
De cada productor
Formas MDP-INSSUM, MDP-INS, MDP-ITL, MDP-LOGD
MDP-SUMP, MDP-SUMQ, MDP-LOGD, MDP-LOGT

SP-POSTMORTEM
Del project manager y team leader

Formas MDP-NSSUM, MDP-INS, MDP-ITL, MDP-LOGD. MDP-SUMP, MDP-SUMQ, MDP-LOGD, MDP-LOGT

(d-t) -DEVELOPMENT MANAGER -> COOPERATIVE WORK
Documentos integrados para *phases* siguientes:

SP-REQUIREMENTS (MILESTONE)
Documento SRS
Plan De Pruebas De Sistema

SP-ANALYSIS&DESIGN (MILESTONE)
Documento SDS
Plan De Prueba Integración

SP-TEST
Material de pruebas integración
Material de pruebas sistema
Producto integrado
Resultado prueba de integración
Resultado pruebas de sistema
Redocumentación (usuario)
Resultado pruebas de aceptación

(e-t) -SUPPORT MANAGER -> COOPERATIVE WORK
SP-LAUNCH&STRATEGY
Proceso de control de configuración
Configuration Control Board y procedimientos

-COOPERATIVE WORK -> SUPPORT MANAGER
SP-STRATEGY
Formas VW-INSSUM, MDP-INS, MDP-ITL

(a-s) –TEAM MEMBER -> STATUS MEETING
Reporte de rol, desarrollo y seguimiento
-STATUS MEETING -> TEAM MEETING
Mediciones relativas al rol y evaluación

(b-s) –TEAM LEADER -> STATUS MEETING
Agenda Reunión Semanal
Reporte de Rol
Reporte Desarrollo y Seguimiento

-STATUS MEETING -> TEAM LEADER
Mediciones relativas al rol y evaluación
Tareas comprometidas para ver terminaron
Datos requeridos de todos miembros
Formas llenas sobre el trabajo hecho a la fecha (formas MDP-TASK, MDP-SCHEDULE, MDP-WEEK)
Estado de los riesgos y comentarios (forma MDP-ITL)
Tareas a realizar en semana siguiente y por quien
Contenido del reporte de la status meeting
Reporte de la Status Meeting , y MDP-CSR

(c-s) –PLANNING MANAGER -> STATUS MEETING
Reporte desarrollo y seguimiento
Weekly status report (MDP-WEEK) DEL EQUIPO
MDP-TASK y MDP-SCHEDULE del equipo
Reporte de rol
Horas y valor-ganado del equipo contra planeado
-STATUS MEETING -> PLANNING MANAGER
Mediciones relativas al rol y evaluación

(d-s) -QUALITY/PROCESS LEADER:STATUS MEETING
Reporte desarrollo y seguimiento
Reportes calidad/proceso
Reporte de rol
Defectos en inspecciones, integración y prueba
Porcentaje de ingenieros siguiendo proceso
Problemas en calidad detectados (posibles)

-STATUS MEETING: QUALITY/PROCESS LEADER
Mediciones relativas al rol y evaluación
Resultados reunión de equipo
Reporte reunión Status Meeting
Formas actualizadas MDP-TASK, MDP-SCHEDULE, MDP-WEEK del equipo e ingeniero, y MDP-CSR
Bitácora MDP-ITL actualizada
Formas MDP-INS, MDP-LOGD de inspecciones

(e-s) –DEVELOPMENT MANAGER -> STATUS MEETING
Reporte desarrollo y seguimiento
Reporte de rol
Componentes diseñados, revisados, inspeccionados, implementados y probados
-STATUS MEETING -> DEVELOPMENT MANAGER
Mediciones relativas al rol y evaluación

(f-s) –SUPPORT MANAGER -> STATUS MEETING

Reporte desarrollo y seguimiento
Reporte de rol
(Configuration, Status Report) MDP-CSR, (Issue Tracking Log) MDP-ITL
-STATUS MEETING -> SUPPORT MANAGER
Mediciones relativas al rol y evaluación

5 Experimentación Usando RUP y AGD

5.1Introducción

Como se desglosa al inicio de este documento (§ 1.1), se estableció al Método de Desarrollo Arquitectónico en Grupo (AGD) como la solución aportada, por nuestra investigación, al siguiente hecho:

> *La mayoría de los enfoques [AWSR03] [RUP03a], para desarrollo de software, dan más importancia al proceso que al producto de software.*

Este hecho sustenta la motivación general de nuestro trabajo de investigación.

En este capítulo validamos el método AGD, enfocándonos en las técnicas cuantitativas empleadas para verificar la existencia una evolución del desarrollo de software, tomando como referencia a la norma de facto actual para desarrollar software: el modelo UP (Unified Process) [JBR99] y su versión comercial el sistema para desarrollo de software RUP (Rational Unified Process) [RUP03a] [RUP03b] [K04].

Las técnicas cuantitativas las utilizamos para constatar que haya mejoras, con relación al RUP, en el producto y en el proceso de desarrollo. El producto y el proceso que se comparan contra el producto y el proceso obtenidos mediante RUP, se obtuvieron usando el método alterno: Desarrollo Arquitectónico y en Grupo (AGD) [GM04].

En este contexto, explicamos el experimento[30] realizado para demostrar la hipótesis relacionada con la mejora del producto y los resultados obtenidos de la experimentación[31]. El análisis de los resultados muestra la relación existente entre los componentes del RUP y del AGD. El método AGD resuelve algunos de los problemas que existen en el desarrollo de software, evidenciados en [H01][H03][A02].

Realizamos un experimento controlado [SWM03] para constatar la evolución del ambiente actual de desarrollo de software. Estableciendo como guía para nuestra comprobación la hipótesis principal siguiente:

---

[30] Experimento – operación realizada en condiciones controladas, para descubrir un efecto o ley desconocidos, para probar o establecer una hipótesis, o para ilustrar una ley desconocida [Merriam-Webster Dictionary].

[31] Experimentación – involucra establecer experimentos controlados o casi-controlados en una empresa, para evaluar un proceso [IEEE SWEBOK, pp. 9-7].

Hipótesis: Productos de software de alta calidad (con menos defectos) pueden obtenerse, utilizando un proceso integrado mediante modos de trabajo-en-grupo; cuando el proceso se dirige hacia obtener los productos-del-trabajo requeridos para construir los componentes de la arquitectura de software fundamentada en los modelos MDA y CWSLR [GJM99B].

En la validación de la hipótesis se llevó a cabo una experimentación, resolviendo un caso real de estudio, planteado siguiendo un problema modelo descrito en [SG95A]. El grupo de desarrolladores estuvo compuesto por 12 estudiantes de Postgrado en Ciencias de la Computación. Formando, aleatóriamente, cuatro equipos de tres miembros cada uno. Dos equipos utilizaron el método AGD y otros dos equipos trabajaron con RUP. La meta del experimento fue medir y analizar tanto características del proceso seguido en el desarrollo, como características del software producido.

Las métricas utilizadas en la experimentación se identificaron usando la técnica GQM (Goal Question Metric) [B92], como se explica en § 5.7.

5.2Establecimiento del Experimento

Se realizaron cuatro réplicas del desarrollo de una aplicación (asignación básica). Cada réplica la realizó concurrentemente un equipo de desarrollo de software, independientemente. Los equipos se identifican (ver columnas de la tabla 5-2) iniciando con: AT aquellos que utilizan el método AGD e inician con RT aquellos que atizan el proceso RUP.

El experimento se realizó como parte de dos cursos equivalentes de nivel de Maestría. Se programaron, en cada curso, dos sesiones de dos horas cada una, dedicadas a presentar la información necesaria para realizar el trabajo. En la primera sesión se impartieron conocimientos acerca de como efectuar la experimentación y el diseño del experimento.

En la segunda sesión se explicó la metodología que utilizaría cada equipo. También se contestó un cuestionario diseñado para conocer la experiencia previa en el desarrollo de software.

Para ayudar a los participantes, se construyó una página Web con acceso a las presentaciones de las sesiones mencionadas anteriormente y el material necesario para realizar la experimentación.

Para completar el ambiente de desarrollo, todos los grupos tenían a su disposición el paquete de software RaTional Rose Enterprise Editio 2003, para realizar los modelos requeridos. También, se asignó un profesor para resolver cualquier duda que tuvieran los equipos de trabajo.

Cuando terminó el experimento, los participantes contestaron un cuestionario acerca de la utilidad de los ambientes de desarrollo usados. Durante la sesión final, también se efectuó una dinámica de grupo, con el propósito de evaluar beneficios y desventajas, de cada método.

Tabla 5-1 Formas de captura de datos empleadas en el experimento.

Evento Asociado	Forma de captura (del experimento)	Forma del AGD (identificación)	Propósito
Uso continuo	Bitácora De Tiempos (Time Recording Log)	MDP-LOGT	Registra el tiempo ocupado en cada tarea del proyecto
	Bitácora de Defectos (Defect Recording Log)	MDP-LOGD	Registra datos de defectos encontrados y corregidos
Ciclo Inicio - Fin	Estrategia (Strategy)	MDP-STRAT	Registra decisiones estratégicas
	Entrega del Producto y Retroalimentación (Product Delivery and Feedback)	DP-PDF	Registra tiempo para recibir retroalimentación y comentarios del cliente
Semana Inicio - Fin	Resumen Semana/Ciclo (Weekly/Cycle Summary)	MDP-WEEK	Prepara el reporte de estado semanal
	Itinerario de Tareas (Tasks schedule)	MDP-SCHEDULE	Estima el tiempo de desarrollo y tamaño de las tareas de un participante
	Solicitud de Cambio (Change Request)	MDP-CCR	Envía un producto al comité de control de cambios (CCB) a incluirse en la línea base del producto
	Reporte de Estado de Configuración (Configuration Status Report)	MDP-CSR	Provee información del estado del sistema de administración de configuración
Término del desarrollo de Producto o Ciclo	Inspección Reporte Detallado (Inspection Report-Detail)	MDP-INS-DET	Recopila datos acerca de inspección/revisión: en detalle
	Inpección Resumen	MDP-INS-SUM	Analiza y totaliza datos acerca de la inspec-

	(Inspection Report-Summary)		ción/revisión

Con la finalidad de analizar los resultados de la experimentación, todos los equipos utilizaron las formas pre-establecidas por el método AGD (columna 3, tabla 5-1). Los nombres en inglés para las formas, se deben a que las publicaciones existentes tanto del RUP como del AGD se han redactado en Inglés. Aunque RUP no incluye formas de captura, con la finalidad de analizar los resultados de la experimentación, todos los equipos siguieron las formas pre-establecidas en el método AGD (columna 3, tabla 5-1).

La primera columna de la tabla 5-1 muestra los eventos que determinan el llenado y recolección de los datos, de cada forma de captura. La segunda columna muestra los nombres de las formas de captura, usadas en el experimento. La tercera columna muestra el identificador de las formas de captura, utilizadas por los equipos que usan el método AGD. Por último, la cuarta columna describe la función que cumple cada forma de captura, para recolectar los datos requeridos.

El conjunto de herramientas Rational de IBM (RUP) puede incluir software externo y "plug-ins" para producir algunos reportes de estado, requeridos para administrar un proyecto de software [K04][W01a][W01b]. Una lista parcial del software externo es: ProjectConsole que proporciona acceso a la información para seguimiento del proyecto, RequisitePro soporta el seguimiento de requisitos, ClearQuest ayuda al manejo de defectos y cambios, ClearCase ofrece manejo de cambios en código y SUMMIT Ascendant así como Microsoft Project sirve para estimar y planear el proyecto.

Debido a que el proceso genérico del RUP no establece cómo organizar la documentación (sus resultados), fue necesario definir un procedimiento[32] que utiliza un conjunto de cinco fólderes de resultados. Los fólderes, de la figura 5-1, se utilizaron por todos los equipos con el objetivo de validar y comparar al RUP y al AGD. El conjunto consiste de las carpetas siguientes: 1) Requerimientos, 2) Diseño, 3) Implementación, 4) De Usuario y 5) Métricas.

Como parte del procedimiento para evaluar la experimentación, los participantes de cada equipo, entregaron una versión de las carpetas: 1) semanalmente y 2) concluido cada ciclo.

Para la experimentación se identificaron las métricas en cuyos valores se espera tener diferencias (ej. cantidad de defectos, tamaño del producto, esfuerzo), que caracterizan el desempeño esperado de los grupos de trabajo y los productos obtenidos. Se observaron discrepancias, tanto en el proceso como en el producto resultante, durante la realización del trabajo de los grupos que usaron el método AGD y los que utilizaron el proceso RUP.

5.3Producto de Software a Desarrollar

A cada equipo, se le pidió diseñar y desarrolla un sistema de software acerca del problema modelo de una biblioteca. La versión utilizada, en la experimentación, es la reportada en [SG95]. Incluimos el enunciado del problema en seguida:

Una biblioteca requiere de un sistema de información que proporcione servicios exclusivos a los usuarios y soporte funciones propias del personal de la biblioteca. Las operaciones en-línea, disponibles para los usuarios, son las siguientes:

1. Registrar la salida (o entrega) de una copia del libro.
2. Obtener una lista de libros por un autor específico, y los libros versados en un tema particular.
3. Encontrar, los libros que tiene en préstamo un usuario (un usuario sólo puede ver la información propia).
4. Encontrar, el último usuario que solicitó una copia de un libro en particular (exclusivo del bibliotecario).
5. Registrar la adición (o eliminación) de una copia de un libro a (de) la biblioteca (exclusivo del bibliotecario).

El sistema requerido debe ser capaz de buscar y actualizar el catálogo del acervo bibliográfico rápidamente (evitar esperas largas a los usuarios, y establecer la búsqueda en línea como

[32] Procedimiento – Curso de acción a tomarse para realizar una tarea dada. [IEEE 610.12]

una alternativa viable al catálogo de tarjetas en papel). El bibliotecario manejará fácilmente las actualizaciones y correcciones a un acervo bibliográfico potencialmente grande.

El sistema también debe implantar las restricciones de integridad siguientes:

1. Todas las copias en la biblioteca deben estar disponibles para revisarse o salir.
2. Ninguna copia puede estar disponible y fuera, al mismo tiempo.
3. Los usuarios no pueden tener, más de un número predeterminado de libros en préstamo, en un momento dado.
4. Los usuarios no pueden haber tramitado más de una copia, de un libro determinado, en cualquier momento.

Requisitos de diseño

La arquitectura a elegir para este problema es un sistema orientado a base de datos. El contexto del problema es buscar y actualizar incrementalmente la información de libros individuales, pertenecientes a una colección grande, en su mayor parte estática. En base a esta premisa se pueden considerar algunas variantes interesantes, de diseño:

- ¿Cómo interactúa la aplicación con la base de datos? Mientras algunos resultados deben producirse en tiempo real, otras transacciones podrían agruparse para procesarse en lote.
- ¿Qué tan centralizado debería ser el sistema? Tanto la base de datos como las aplicaciones podrían distribuirse en varias máquinas.
- ¿Influye el tipo de base de datos en la elección de la arquitectura de software? Sistemas que utilizan base de datos orientadas a objetos se destinan a implementaciones diferentes, en relación a aquellos sistemas con bases de datos relacionales.

También, en el enunciado, se pide considerar diseños que no permitan ubicar, en la base de datos, aquella información relativa a la circulación del material bibliográfico.

Utilizando el enunciado del problema planteado, se cuantificó el tamaño del producto requerido, con el propósito de comparar este valor contra los productos resultantes. Para tal efecto, se utilizó la métrica de puntos de función de Albrecht [A79][OR00][A98]. Los puntos de función, posibilitan la evaluación cuantitativa acerca de la porción de funcionalidad requerida que satisfacen los productos obtenidos en el experimento. La fórmula de Albrecht para calcular la cuenta de puntos de función no-ajustado (UFP: Unadjusted Function Points), se expresa como:

Experimento: Fólder de Requerimientos	
AGD: Fólder de Visión/Planificación(FV) CIM-(CWSP Inicialización) CIM-(Planificación/Visión) CIM-(Alcance/Prioridad) CIM-(Resultados de Revisión de FV) *AGD: Fólder de Requerimientos(FR)* CIM-(Negocio existente) CIM-(Identificación de componentes)	CIM-(Integración del prototipo del sistema) CIM-(Requerimientos del sistema) PIM-(Ingeniería inversa) PIM-(Ingeniería directa) PIM-(Integración del prototipo del software) PSM-(restricciones de requerimientos) PSM-(Especificación de prueba de aceptación) PSM-(Requerimientos del software) PSM-(Resultados de Revisión de FR)
Experimento: Fólder de Diseño	**Experimento: Fólder de Usuario**
AGD: Fólder de Diseño (FD) PSM-(Arquitectura del componente) PSM-(Especificación prueba integración) PSM-(Análisis de arquitectura) PSM-(Estructura de división del trabajo y asignación de unidades) PSM-(Resultados de Revisión de FD)	*AGD: Fólder de Operación* IM-(Operación inicial) IM-(Resultados prueba de integración) OM-(Identificación de recursos) OM-(Distribución) OM-(Liberación de producto) OM-(Resultado de prueba de aceptación) OM-(Resultados de evaluación)
Experimento: Fólder de Implementación	**Experimento: Métricas (formas)**
AGD: Fólder de desarrollo de unidad Selección de requerimientos de software (desde FR) Selección de diseño general (desde FD) IM-(Diseño detallado) IM-(Especificación de prueba de unidad) IM-(Resultado de revisdón diseño detallado) IM-(Lista de funciones) IM-(Código fuente) IM-(Resultado de prueba de unidad) Reporte de defectos IM-(IM-(Resultados de Revisión de UDF)	*AGD: Formas* Bitácora de registro de tiempos(MDP-LOGT) Bitácora de registro de defectos(MDP-LOGD) Estrategia (MDP-STRAT) Liberación de producto y retroalimentación (MDP-PDF) Reporte de estado Semana/Ciclo(MDP-WEEK) Programa de tareas(MDP-SCHEDULE) Solicitud de cambio en configuración (MDP-CCR) Reporte de estado de la configuración (MDP-CSR) Reporte de inspección-detalle (MDP-INS-DET) Reporte de inspección-resumen (MDP-INS-SUM)

Figura 5-1 Coincidencia entre Carpetas de Experimentación y Carpetas del AGD.

$$UFP = IN*WI + OUT*WO + INQ*WQ + FILE*WF + EINT*WE \quad (1)$$

Donde: UFP: puntos de función no-ajustados; IN: entradas externas; OUT: salidas externas; INQ: consultas externas; FILE: archivos lógicos internos; EINT: archivos de interfase externos. Así como los pesos de complejidad siguientes: WI: peso para entradas; WO: peso para salidas; WQ: peso para consultas; WF: peso para archivos internos; WE: peso para interfases externas.

Se analizó el postulado del problema para poder aplicar esta fórmula y cuantificar la funcionalidad: identificamos IN = 5 entradas externas, OUT = 9 salidas externas, INQ = 16 consultas externas, FILE = 5 archivos lógicos internos y EINT = 2 archivos de interfase externos.

Sustituyendo los valores mencionados y los pesos (weight) de complejidad promedio siguientes: para entradas WI = 4, para salidas WO = 5, para consultas WQ = 4, para archivos internos WF = 10 y para interfases externas WE = 7, resulta:

$$UFP = 4*4 + 8*5 + 15*4 + 4*10 + 2*7 = 170 \text{ FP} \quad (2)$$

El valor obtenido de puntos de función se ajusta de acuerdo con la plataforma tecnológica requerida por el cliente. Se calcula un factor de ajuste asignando un peso a 14 características específicas de la aplicación. Para la experimentación, el cliente solicitó una aplicación Web, cuyas propiedades rigen la asignación de los pesos.

Para calcular el factor de ajuste AF, los valores de peso para aplicaciones Web (C_i para la i-ecima característica, en la ecuación siguiente), se determinaron de la manera siguiente: a) Un valor 5 (alto) para las propiedades siguientes: comunicación de datos, proceso distribuido, desempeño crítico, entrada de datos en línea, actualización de archivos en línea y facilidad de cambios a los datos & facilidad de uso de la información; y b) Un valor 3 (normal) para las ocho peculiaridades restantes. Cuando se utilizan los valores mencionados (5 = alto y 3 = normal) resulta el factor de ajuste siguiente:

$$AF = 0.65 + 0.01 * \sum_{i=1}^{14} C_i = 0.65 + 0.54 = 1.19 \quad (3)$$

El factor de ajuste AF modifica los puntos de función UFP rersultando la cuenta de Puntos de Función Ajustados (AFP). En el caso del problema tratado, resulta:

$$AFP = UFP*AF = 170*1.19 = 202.3 \text{ FP} \quad (4)$$

El valor AFP, se utiliza en la interpretación del experimento (en las § 6.8 y 6.9), comparándolo con los puntos de función evaluados para los productos finales, y con los puntos de función omitidos por cada equipo.

5.4Procedimiento del Experimento

El sistema se desarrolló en un periodo de 28 días hábiles, suficientes para completar el producto del software. Las actividades efectuadas fueron las siguientes:

Paso 1 – Reiteradamente los equipos desarrollaron su asignación empleando las formas de captura mostradas en la figura 5-1 y los productos-del-trabajo requeridos.

Paso 2 –El producto de software terminado se instaló en la computadora personal del cliente (en la biblioteca).

Paso 3 – Los desarrolladores entregaron las carpetas de documentación del producto a los investigadores que asesoraron el proyecto.

Paso 4 - Las pruebas de aceptación del producto entregado se realizaron usando la plataforma del cliente.

5.5Métricas de Evaluación del Producto y del Proceso de Desarrollo

Seleccionamos un conjunto de métricas, con el propósito de evaluar la calidad del producto de software y el proceso de desarrollo:

M1_A - Defectos identificados en el producto (durante la prueba de aceptación por el cliente).

M2_TC – Líneas de Código (LOC) del producto completo (tamaño del producto).

M2_CN – LOCs de código nuevo o modificado (tamaño del producto).

M3 –A - Omisiones identificadas (durante la prueba de aceptación, en puntos de función).

M5_T - Esfuerzo total (tiempo/hombre).

M7_T - Cuenta de puntos de función en el producto (AFP, tamaño del producto).

La tabla 5-2 muestra los valores obtenidos de las métricas. La experiencia previa, en el desarrollo de software, se calcula a partir de un cuestionario y cuyos valores se encuentran en el rango entre 1 y 5. Las respuestas determinaron un mínimo 1.7 y un máximo de 3.0. El valor mínimo significa que la experiencia en desarrollo de software se obtuvo en cursos y el valor máximo significa que la experiencia se ganó durante la participación en proyectos de cursos. Prácticamente ninguno de los equipos contaba con experiencia laboral en desarrollo de software.

Líneas de Código: M2_xx

La métrica M2_TC, reportada en número de Líneas de Código (LOCs), acumula LOCs generadas automáticamente con LOCs nuevas y modificadas. La cantidad de código generado automáticamente (AC) se calcula restando el valor de M2_CN del valor de M2_TC (Ej. Para AT(1): 1965 – 743 = 1222).

Para este estudio la razón de código generado automáticamente dividido entre el código nuevo o modificado AC / M2_CN está dentro del rango de valores que va desde 1.1 para el equipo AT(2) hasta 1.64 para el equipo AT(1) (Ej. Para AT(1): 1222/743 = 1.64).

Tabla 5-2 Valores de las métricas recolectadas.

Métricas	AT(1)	AT(2)	RT(1)	RT(2)
Plataforma de desarrollo	Web-JSP + Java + MySQL, MySQL -front, JDBC	Visual Basic + Access	Visual Basic + Access + Crystal Reports	Visual Basic + MySQL, MySQL-front, Driver-ODBC de MySQL
Experiencia previa en desarrollo de software	2.3	3.0	1.7	2.1
Tamaño del producto M2_TC – Código total (LOC)	1965 *	5539	4729	3356
M2_CN – Código Nuevo y modificado (LOC)	743 *	2635	1820	1305
M7_T – Funcionalidad del producto AFP (FP, Function Points)	94.00	174.72	145.60	205.92 *
M5_T – Esfuerzo total real (person hour)	167:59	211:27	105:52 *	109:33
M3_A – Omisiones identificadas por el cliente: En la prueba de aceptación (FP)	120.19	119.00	92.82	29.75 *
M1_A – Defectos encontrados en la prueba de aceptación	6 *	10	13	13

* Mejor valor para la métrica (aislada)

Puntos de Función: M7_T y M3_A

Con la finalidad de hacer una evaluación comparativa se considera la funcionalidad requerida del producto, así como la del producto terminado. La funcionalidad del producto, medida en puntos de función (calculada a partir del planteamiento del problema), cuyo valor es 202.3 FP (§ 6.3).
La funcionalidad de los productos de software obtenidos, se midió usando también, puntos de función: esta métrica se nombró M7_T y se puede comparar con los valores obtenidos para M3_A que representa los puntos de función omitidos -de acuerdo a lo identificado por el cliente. Como se desarrolló una aplicación Web, se utilizó un factor de ajuste (AF) de 1.19, para los valores de puntos de función obtenidos y omitidos por el equipo AT(1). En el caso de los otros tres equipos el valor de AF es de 1.04, ya que los productos que desarrollaron no son aplicaciones Web.

Horas Persona: M5_T

El esfuerzo total real, M5_T, corresponde a la cantidad de horas ocupadas por los miembros del equipo en el desarrollo. Recolectado con las formas de captura Bitácora de Registro de Tiempo / Time Recording Log y el Reporte de Estado Semanal/Ciclo / Weekly/Cycle Status Report. Los valores de esta métrica se requieren para calcular las razones de productividad, habilitando un proceso de mejora continua.

Número de Defectos: M1_A

La cantidad de defectos identificados durante la prueba de aceptación, que nombramos M1_A, se obtiene con las formas de captura: Bitácora de Registro de Defectos / Defect Recording Log y Entrega de Producto & Retroalimentación / Product Delivery & Feedback. Esta métrica se obtiene después de entregar el producto de software, y es importante para evaluar la calidad del producto y del proceso seguido en su desarrollo, según lo percibe el cliente.

Necesidad de Combinar Métricas

El mejor valor de la métrica por cada renglón está marcado con un (*). Además, se requiere considerar los valores de otras métricas para evaluar correctamente al producto y al proceso. Por ejemplo, el mejor valor de la métrica M2_CN (Código Nuevo o Modificado) es 743 LOCs, desarrolladas por el equipo AT(1).

Sin embargo, para tener una percepción global objetiva, M2_CN se relaciona con: a) el valor 94.0 FP de M7_T (la funcionalidad del producto), b) el valor 6 para M1_A (defectos encontrados en la prueba de aceptación) y c) el valor 167.59 horas/persona para M5_T (esfuerzo total real). En este ejemplo AT(1) entrega un producto de software con un número menor de LOCs y menos defectos, pero implementa 94 puntos de función (la funcionalidad menor de todas).

Si complementamos nuestro ejemplo considerando ahora al equipo RT(2) y comparamos los valores de sus métricas con los obtenidos por el equipo AT(1), notamos que M2_CN (Código Nuevo o Modificado) tiene un valor de 1305 LOCs, que es aproximadamente el doble de lo logrado por AT(1), con un valor del doble para M7_T = 205.92, y también el doble para el valor de M1_A = 13 defectos. Considerando estas métricas (M2_CN, M7_T y M1_A), los equipos AT(1) y RT(2) se pueden considerar como equivalentes desde la perspectiva del desempeño.

Utilizando el conjunto de métricas explicado en esta sección y tomando en cuenta la necesidad de considerar combinaciones de las mismas, a continuación, se explica el enfoque estadístico utilizado en la experimentación.

5.6Enfoque Estadístico

La metodología utilizada considera un factor de trato experimental (variable independiente) que tiene dos instancias asociadas: el método AGD y el proceso genérico RUP. Además, Incluye un conjunto de observaciones experimentales (variables dependientes) que contiene 6 métricas principales, relacionadas con el proceso y el producto de software. Estas métricas se muestran en la tabla 5-2.

Mediante estas métricas se cuantificó el desempeño de cada equipo de desarrollo. La experimentación consistió técnicamente en una serie de experimentos univariados simultáneos. Es decir, se realizó un experimento independiente para cada uno de los aspectos del desarrollo (métrica) a observar, recopilando todas las métricas a la vez. Todos los experimentos compartieron un diseño único y una muestra de datos.

Para recolectar datos, de dos ambientes diferentes de desarrollo, utilizamos una capa de soporte para el desarrollo del experimento. Esta capa de soporte incluye, como se ve en la figura 5-1, 10 formas para captura de datos (en el fólder Experimento: Métricas), cinco carpetas para ubicar los resultados (Fólder de Requerimientos, Fólder de Diseño, Fólder de Implementación, Fólder de Usuario y Fólder de Experimento) y un procedimiento para la realización del mismo.

Las formas para capturar los datos se usan en los dos ambientes, AGD y RUP. AGD cuenta con formas de captura parecidas, pero RUP no incluye la captura de los datos del proyecto. El conjunto de herramientas de Racional (IBM) puede incluir elementos adicionales, como son software externo y “plug-ins” para producir algunos reportes de estado, requeridos para administrar un proyecto de software [W01a] [W01b]. Incluimos una lista parcial de los elementos adicionales disponibles: ProjectConsole, RequisitePro, ClearQuest, ClearCase, SUMMIT Ascendant, y Microsoft Project.

La hipótesis del experimento (planteada al inicio de la sección 6.1) se interpreta en una forma manejable estadísticamente, resultando pares de hipótesis estadísticas constituidas por: Hipótesis Nula e Hipótesis Alterna. La Hipótesis Nula H_0 toma en cuenta la media de las distribuciones de la métrica en turno son iguales ($\mu_A = \mu_R$). La Hipótesis Alterna H_A significa que las medias de las distribuciones son diferentes y una de ellas es mayor ($\mu_A \neq \mu_R$ and ($\mu_A < \mu_R$ or $\mu_A > \mu_R$)).

Por ejemplo, una de las hipótesis para tratamiento estadístico que mencionamos en secciones posteriores es: Un equipo trabajando con el método AGD entrega productos de software con una densidad de defectos menor, en la prueba de aceptación, que un equipo trabajando con RUP. Esta es la hipótesis alterna H_A para la métrica M1_A.

De esta forma, en una métrica específica, el par de hipótesis para el experimento son las tratables estadísticamente. El resultado de las pruebas estadísticas, para evaluar las hipótesis, se

analiza utilizando los conceptos de Hipótesis Nula e Hipótesis Alterna, y se interpreta para producir una conclusión.

La prueba estadística usada en el estudio es una prueba *t* de dos muestras [B94] [K78], asumiendo dos poblaciones gobernadas por distribuciones de probabilidad normal, y que las desviaciones Standard de las dos poblaciones son iguales $\sigma_1 = \sigma_2$. Otra prueba utilizada es la "Wilconox rank sum", no-paramétrica para muestras pequeñas que no requiere ninguna de las suposiciones mencionadas para la prueba t. La prueba Wilconox se aplica para obtener el umbral o valor-p de α (asociado con el error tipo I). El valor-p es el valor de α que apenas causa el rechazo de la hipótesis H_0.

Para interpretar los resultados obtenidos se utilizan marcos que proveen la base organizacional para abstraer y conceptuar el conjunto de hipótesis estadísticas. La interpretación tiene como finalidad obtener un conjunto de conclusiones.

En este caso del experimento realizado, se definieron dos marcos de estudio: 1) De Mediciones.- constituido por todos los resultados posibles para una métrica dada, y 2) De Criterios.- formado por creencias generales que consideran los efectos de los tratamientos experimentales (AGD y RUP) sobre los resultados de las comparaciones utilizadas.

El primer marco se emplea para obtener las conclusiones estadísticas, mientras que el segundo marco se utiliza en la interpretación de la experimentación, e involucra juicios subjetivos. En la siguiente sección incluimos los resultados de aplicar el paradigma Metas, Preguntas y Métricas / Goal Question Metric (GQM) [B92], mediante el cual establecimos los marcos mencionados, para la investigación.

5.7Creencias, Metas, Preguntas y Métricas

En esta sección incluimos el conjunto completo de creencias y metas que establece el marco de criterios, para interpretar las conclusiones estadísticas. Estos criterios se utilizan para identificar las preguntas y en última instancia las métricas que forman el marco de evaluación.

Para poder utilizar el paradigma GQM tuvimos que establecer las **creencias** o hipótesis que nos motivaban, presentadas a continuación:

C1 - AGD genera productos de software de alta calidad mediante un proceso que se ajusta a modelos de referencia centrados en la arquitectura del producto.

C2 - AGD es un método fácil de aprender y modificar, porque se organiza con dos componentes independientes: 1) El Conjunto de productos (Product Set) y 2) el Proceso Dirigido por Modelos ((MDP).

C3 - AGD es eficiente pues requiere menor tiempo para desarrollar planes y productos, porque esta guiado por componentes bien organizados (Product Set y MDP).

C4 - AGD produce productos documentados, pues utiliza UML, se ajusta a modelos de referencia MDA y CWSLR y usa la estructura de resultados UDF (Unit Development Folder).

Se partió de las creencias anteriores para identificar las **metas** (G), Preguntas (Q) y Métricas (M) que se pretendían probar mediante la experimentación, como se muestran en la tabla 5-3.

Inicialmente se identificaron cinco metas (ej. G1 - Obtener productos de buena calidad, G2- Utilizar un proceso de buena calidad, …), catorce preguntas (ej. Q1 - ¿Eficiencia del produc-

to?, Q2 - ¿Densidad en el producto?, …) y quince métricas (ej. M1 – cantidad de defectos en el producto, M2 – Tamaño del producto, …). También se estableció la relación que existía entre las preguntas y las métricas utilizadas para contestarlas (ej. Q1 → M1, M5, Q2 → M1, M2, …).

La relación Q1 → M1, M5, por ejemplo, indica que la pregunta Q1 acerca de la eficiencia (¿que tan bien funciona?) del producto, se puede responder con las métricas M1 y M5, que cuantifican los defectos encontrados en el producto y el tiempo dedicado a desarrollar el software.

Del conjunto inicial de metas, preguntas y métricas mostradas en la tabla 5-3, se eligieron los elementos que era factible utilizar en el contexto de la experimentación, debido a la inmadurez de capacidad de desarrollo de todos los equipos. Incluimos a continuación las metas, preguntas y métricas usadas:

G1 – Obtener productos de buena calidad (con menos defectos en documentación y código).
G2 – Utilizar un proceso de buena calidad.

A partir de las creencias y metas, se construyeron un conjunto de **preguntas** y de **métricas** necesarias para responderlas. El conjunto completo se incluyó en la tabla 5-3 y en tabla 5-4 incluimos solamente el subconjunto utilizado en la experimentación. Este conjunto de métricas se utilizó para interpretar los resultados obtenidos.

A partir del conjunto de métricas generales, de la tabla 5-4, se definieron métricas más específicasas. Por ejemplo, la métrica "M1 Defectos Identificados en el Producto" se subdividió en las métricas de la tabla 5-5 (ej. M1_R defectos identificados en requerimientos, M1_D defectos identificados en diseño, …).

Como se observa en la tabla 5-4, para contestar cada pregunta y obtener conclusiones se consideran varias métricas relacionadas (ej. La pregunta Q1 se relaciona con las métricas M1 y M5). La combinación de las métricas, mostrada en la columna "Relaciones", generó el conjunto de métricas compuestas que se muestra en la tabla 5-6 (ej. "Densidad de defectos en aceptación, por cada mil líneas de código" se calcula con la fórmula M1_A*1000/M2_CN). Las métricas compuestas se utilizaron también en la interpretación de los resultados del experimento, explicada en las secciones siguientes.

Tabla 5-3 Lista original de Metas, Preguntas y Métricas.

META (G)	PREGUNTAS (Q)	RELACIONES	METRICAS
G1 Obtener productos de buena calidad			
	Q1.- eficiencia del producto	Q1 → M1, M5	**M1** Cantidad de defectos en el producto
	Q2.- densidad de defectos en el producto	Q2 → M1, M2	**M2** Tamaño del producto
	Q3.- omisiones del producto	Q3 → M3, M7	**M3** Omisiones identificadas por el cliente
	Q4.- cambios en el producto	Q4 → M4	**M7** Funcionalidad del producto
			M8 Acumulado de defectos en el producto
G2 Utilizar un proceso de Buena calidad			
	Q5.- productividad con retroalimentación en el producto	Q5 → M10	**M4** Cambios
	Q6.- retroalimentación de un producto	Q6 → M11	**M9** Cambios diferidos
	Q7.- cambios diferidos en un producto	Q7 → M9	**M10** Tiempo de retroalimentación
			M11 Retroalimentación
G3 Facilitar el aprendizaje y modificación del conjunto de productos (PS)			
	Q8.- cantidad de referencias en el producto: uso del CWSLR	Q8 → M12, M13	**M12** Búsquedas
	Q9.- cantidad de referencias en el producto: uso de la MDA	Q9 → M12, M13	**M13** esfuerzo de búsqueda
	Q10.- cantidad de referencias en el producto: uso de ejemplos de producto	Q10 → M12, M13	**M14** Consultas
G4 Facilitar el aprendizaje y modificación del Proceso Dirigido por Modelos (MDP)			
	Q11.- cantidad de referencias al producto, uso de descripciones del proceso	Q11 → M12, M13	**M15** Esfuerzo de experto
G5 Eficientar el Proceso			
	Q12.- productividad en el producto	Q12 → M2, M5	**M5** Esfuerzo real
	Q13.- cantidad de consultas del dominio para un producto	Q13 → M14, M15	**M6** Esfuerzo planificado
	Q14.- desviación de la estimación de tiempo para el producto	Q14 → M5, M6	

Tabla 5-4 Subconjunto de Metas, Preguntas y Métricas utilizadas en la experimentación.

METAS (G)	PREGUNTAS (Q)	RELACIONES	METRICAS
G1 Productos de buena calidad			
	Q1.- ¿eficiencia en un producto?: AGD Vs. RUP	Q1 → M1, M5	M1 Defectos identificados en el producto
	Q2.- ¿densidad de defectos en un producto?: AGD Vs. RUP	Q2 → M1, M2, M7	M2 Tamaño de productos
	Q3).- ¿omisiones en un producto y subconjunto de productos?: AGD Vs. RUP	Q3 → M3, M7	M3 Omisiones identificadas por el cliente
	Q4.- ¿cambios en un producto?: AGD Vs. RUP	Q4 → M2, M4	M4 Cambios
G2 Proceso de buena calidad			
	Q5.- ¿productividad por retroalimentación?: : AGD Vs. RUP	Q5 → M3, M5, M7	M5 Esfuerzo real
	Q6.- ¿productividad en un producto?: AGD Vs. RUP	Q6 → M2, M5	M6 Esfuerzo planeado
	Q7.- ¿desviación en tiempo para un producto?: AGD Vs. RUP	Q7 → M5, M6	M7 Puntos de función

Tabla 5-5 Sub-métricas de M1 (Defectos identificados en el producto).

M1.- Defectos identificados en el producto M1_R.- en requerimientos
M1_D.- en diseño
M1_I.- en implementación
M1_TD.-Total de defectos encontrados en el desarrollo
M1_A.- en prueba de aceptación
M1_C.- defectos acumulados
M1_O.- en operación

Tabla 5-6 Métricas Compuestas.

Nombre de la Métrica
Densidad de Defectos en Aceptación M1_A*1000/M2_CN (Defectos/1K LOC)
Densidad de Defectos en Aceptación – FP M1_A*1000/M7_T (Defectos/1K FP)
PF_satisfechos AFP – M3_A (FP)
PF-desviación M7_T – PF_satisfechos (FP)
Eficacia M2_CN/M7_T (LOCs/FP)
Productividad M2_CN/M5_T (LOCs/Hr)

En la tabla 5-6 se incluye, debajo del nombre dado a cada métrica compuesta, la fórmula con la que se calcula. Por ejemplo, la cantidad de puntos de función satisfechos se calcula por la diferencia de los puntos de función ajustados (AFP), requeridos para el producto, menos las omisiones identificadas por el cliente (M3_A), esto es:

$$PF\_satisfechos = AFP - M3\_A \quad (5)$$

5.8Tratamiento de las Hipótesis

En esta sección explicamos los resultados obtenidos para las hipótesis estadísticas siguientes:

- Los equipos trabajando con el método AGD producen software con menor densidad de defectos (en aceptación), con respecto a los equipos que usaron RUP.
  Diferencia en la variable dependiente "M1_A Defectos identificados en la aceptación" debido a los dos tratamientos independientes: realización mediante AGD o realización siguiendo RUP.
- Los equipos trabajando con el método AGD producen software con menor número de LOCs, por punto de función, en relación a los equipos que usaron RUP.
  Efecto sobre la variable dependiente "Eficacia" ocasionado por los dos tratamientos independientes: realización mediante AGD o realización siguiendo RUP.
- Los equipos trabajando con el método AGD producen software con menor desviación funcional, con relación a los equipos que usaron RUP.
  Relación de la variable dependiente PF_desviación con los dos tratamientos independientes: realización mediante AGD o realización siguiendo RUP.

Las hipótesis reflejan nuestra creencia de que utilizar al método AGD producirá una mejora, con referencia al RUP, en el producto de software obtenido (ej. menor número de defectos detectados en ejecución) y en el proceso de desarrollo involucrado (ej. menores desviaciones en la funcionalidad obtenida).

5.8.1 Producción de software con menor densidad de defectos

El usuario de un producto de software percibe su calidad principalmente por el número de defectos que observa, cuando el producto se instala y usa en el equipo en que correrá al operarse diariamente. En el contexto del experimento realizado, una situación similar ocurrió en la prueba de aceptación, después que los equipos entregaron sus productos al cliente, y prepararon el ambiente de trabajo para que se usara el producto de software. El cliente, junto con un investigador, probaron el producto de software y recopilaron la métrica M1_A (defectos Identificados en la prueba de aceptación). Esta métrica resume el procedimiento de la prueba de aceptación.

Con estos datos efectuamos la prueba de hipótesis para las medias, utilizando la herramienta estadística anexa de [B94]: La prueba proporcionó resultados que rechazan la hipótesis nula ($\mu_A = \mu_R$), con un nivel de confianza del 93.5% y un valor-p de 0.167. Este desenlace implica la existencia de un efecto verdadero, más allá del efecto posible causado por el azar, y que $\mu_A < \mu_R$. El cliente evaluó que el producto desarrollado mediante el método AGD tiene mejor calidad (con $\mu_A = 8$ Defectos).

También se puede validar el resultado anterior usando métricas compuestas (tabla 5-7). Mediante la métrica: Densidad de Defectos en prueba de Aceptación, y cuyo resultado representa los defectos identificados en mil puntos de función. Utilizando la densidad de defectos, la hipótesis nula se rechaza con un nivel de certeza de 81.8% y un valor-p de 0.33. Indicando que hay un efecto significativo verdadero, que $\mu_A < \mu_R$ y la media de la distribución es $\mu_A = 60.53$ Defectos/1kFP.

La validación de nuestra hipótesis para la métrica M1_A: un equipo trabajando con el método AGD entrega productos de software de mejor calidad, la interpretamos de la manera siguiente:

Los grupos de desarrollo que utilizaron AGD consideraron con mayor atención las características de los productos de trabajo durante la definición de la estrategia y planificación de sus actividades. La atención mayor sobre los productos del trabajo ocasionó que, los dos equipos que utilizaron el método AGD, decidieran reducir la cantidad de funcionalidad a entregar. Logrando mayor calidad (M1_A = 6 para TA(1) y M1_A = 10 para TA(2); pero tuvieron también algunos efectos colaterales, como los mencionados a continuación.

Los equipos que usaron AGD obtuvieron los valores extremos de desviación, efecto ocasionado por haberse enfocado en el producto. El equipo AT(1), decidió reducir el alcance a lograr (M5_T = 94.00 PF, debiendo haber entregado 202.3 PF). Desarrollaron solamente la funcionalidad requerida originalmente, para el alcance recortado resultante. De esta forma lograron conseguir la menor desviación (expresada en puntos de función = 11.89 AFP).

En contraste, el equipo AT(2) decidió también reducir el alcance; pero entregaron más resultados que los requeridos originalmente para el alcance recortado (M5_T = 174.72 PF, debiendo haber entregado 202.3 PF). Lo que significó implantar consultas alternas, más despliegues de información con diferentes ordenamientos y opciones adicionales. De esta manera obtuvieron la mayor desviación (expresada en puntos de función = 91.42 AFP).

Tabla 5-7 Métricas Compuestas.

Metrica Compuesta	AT(1)	T(2)	RT(1)	RT(2)
Densidad de defectos en prueba de aceptación M1_A*1000/M2_CN (Defects/1K LOC)	8.1	3.8 *	7.1	10.0
FP- Densidad de defectos en prueba de aceptación (Defects/1K FP) M1_A*1000/M7_T	3.83	57.2 *	89.29	63.13
FP_satisfechos (FP) (AFP – M3_A)	2.11	83.3	82.11	72.55 *
FP_desviación (FP) (M7_T – FP_satisfechos)	1.89 *	91.42	63.49	3.37
Eficacia M2_CN/M7_T (LOCs/FP)	7.28	15.68	13.00	6.59 *
Productividad M2_CN/M5_T (LOCs/Hr)	4.42	12.46	17.19 *	11.91

* Mejor valor para la métrica

Otro comentario, relativo a los equipos que trabajaron con el método AGD: los equipos se organizaron durante todo el proceso con roles orientados a realizar el trabajo en equipo (ej. Líder del equipo, administrador de planificación, etc.). Esta orientación hacia el trabajo en equipo en la forma de asignar roles, mejoró su desempeño.

Los equipos que trabajaron con RUP, asignaron roles de acuerdo al tipo de conocimientos técnicos que requerían las actividades a realizar (ej. Arquitecto, Ingeniero de Casos de Uso, etc.). Este tipo de consideración proporcionó poca ayuda en organizar el trabajo en equipo, pues no se repartieron las responsabilidades administrativas relacionadas con el trabajo en equipo.

En cuanto a los equipos que trabajaron con RUP, se orientaron principalmente al proceso de desarrollo y a la división del trabajo. Con esta estrategia, el equipo RT(2) dividió su esfuerzo de implementación, siguiendo las tres opciones principales (de la pantalla principal) Se asignó una opción a cada uno de tres miembros del equipo. De esta manera realizaron un muy buen trabajo y obtuvieron la mejor funcionalidad (ver tabla 5-2, M7_T = 205.92 AFP); pero las tres interfases de usuario resultaron diferentes, pues no hubo comunicación suficiente entre sus integrantes.

Estilos diferentes de interfases de usuario, para cada opción, implicó que el usuario de la aplicación percibiera mayor complejidad, incrementando el esfuerzo de aprendizaje para utilizarla.

5.8.2 Software con menor número de LOCs, por punto de función

La segunda hipótesis supone que un equipo que trabaja con el método AGD produce menos LOCs por punto de función. Para validar esta hipótesis, utilizamos la razón de eficacia mostrada en el renglón 5 de la tabla 5-7: que relaciona la cantidad de líneas de código con la cuenta de puntos de función. Se requiere codificar la cantidad mínima necesaria de LOCs por punto de función.

Cuando se efectúa la prueba de hipótesis de dos medias de la eficacia, la hipótesis nula ($\mu_A = \mu_R$) no se rechaza. Lográndose un nivel de certeza de 39% y un valor-p de 0.667. Estos valores indican que no hay certeza de que haya diferencia significativa en la eficacia obtenida usando AGD y la obtenida usando RUP. La incertidumbre en la existencia de una diferencia en la eficacia y el número pequeño de equipos participantes sugiere: continuar experimentando.

5.8.3 Desviación Funcional

La cantidad de puntos de función realizados por un equipo puede determinarse a partir de la métrica Funcionalidad del Producto M7_T y de la métrica de Puntos de Función Omitidos M3_A. Para el equipo RT(2) (renglón 3 de la tabla 5-7), la cantidad de puntos de función satisfechos se determina como se muestra a continuación:

$$FP\_satisfechos\ (RT(2)) = AFP - M3\_A(RT(2)) = 202.3 - 29.75 = 172.55 \quad (6)$$

Con este resultado, se puede estimar la desviación estandar del equipo, en relación a la funcionalidad planificada originalmente. La desviación se muestra en el renglón 4 de la tabla 5-7. Calculándose como sigue:

$$FP\_desviación\ (RT(2)) = M7\_T\ (TR(2)) - FP\_satisfechos\ (RT(2)) = 205.92 - 172.55 = 33.37 \quad (7)$$

El esfuerzo requerido para proveer los 33.37 puntos de función (de desviación) se pudiera haber utilizado para desarrollar las omisiones identificadas por el cliente: M3_A = 29.75 puntos de función (ver el renglón 6 de la tabla 5-1). Consideramos este hecho como un indicador de la posibilidad de mejorar el proceso, si se reorienta el esfuerzo para prevenir las desviaciones.

Para la métrica FP_desviación no se rechaza la hipótesis nula ($\mu_A = \mu_R$) obteniendo en la prueba un nivel de confianza de 46% y un valor-p de 0.667. Este resultado indica que no hay certeza en la existencia de una diferencia en FP_desviación cuando se usa AGD o RUP. Dado el resultado obtenido y la cantidad reducida de equipos participantes en el experimento, se requiere realizar más experimentación.

5.9Conclusión

Nuestra hipótesis de "Productos de alta calidad" se tradujo a un conjunto de hipótesis estadísticas, donde cada hipótesis compara las medias de las distribuciones de una métrica. Una de las hipótesis estadísticas del desarrollo de software, cuando se utiliza el método AGD, es:

> *Trabajar con el método AGD permite entrega productos de software con menor número de defectos, al momento de la prueba de aceptación, que un equipo trabajando con RUP.*
>
> *Esta hipótesis es equivalente a la hipótesis alterna* $\boldsymbol{H_A}$ *para la métrica M1_A (defectos encontrados durante la prueba de aceptación).*

La métrica M1_A (renglón 7 de la tabla 5-2) resume el procedimiento de prueba de aceptación, realizado conjuntamente por el cliente y los investigadores. Cuando aplicamos las pruebas a las dos distribuciones de los defectos encontrados durante la aceptación, se rechaza la hipótesis nula ($\mu_A = \mu_R$). El nivel de confianza es 93.5% y el valor-p es 0.167. La interpretación de este hecho es: existe un efecto realmente significativo, diferente al efecto posible debido al azar, y $\mu_A < \mu_R$. En consecuencia, el cliente evalúa de mejor calidad el producto desarrollado usando AGD (μ_A = 8 Defectos).

En relación con las otras cinco métricas recolectadas durante el experimento (ver tabla 5-2), no se rechazó la hipótesis nula (las medias de las distribuciones son iguales ($\mu_A = \mu_R$)), con-

cluyendo que se requiere realizar más repeticiones de la experimentación, para rechazar o aceptar las hipótesis correspondientes, en forma definitiva.

Hasta ahora, con la experimentación realizada se determinó que utilizando el método AGD se obtiene un producto y un proceso de calidad igual o mejor que los obtenidos al utilizar RUP.

Además de la recolección de métricas y de su interpretación estadística, se aplicaron cuestionarios, después de la experimentación, para evaluar la satisfacción de los participantes en cuanto al ambiente utilizado y a la forma en que se llevó el experimento. De esta información dedujimos que la capa común desarrollada para efectuar el experimento proporcionó un buen ambiente de trabajo a los participantes. La capa común del experimento, con sus formas de captura y carpetas de resultados predefinidas, mejoró el desempeño de los equipos que trabajaron con RUP.

De los cuestionarios y evaluación del experimento, se obtuvieron comentarios y útil retroalimentación, orientados a optimizar: tanto el método AGD, como el diseño del experimento

Los resultados evidencían que AGD propicia que los desarrolladores produzcan software con menor densidad de defectos identificados en la prueba de aceptación. Interpretamos este hecho, como un efecto de la importancia dada a los productos-del-trabajo obtenidos y de la sinergia de los modos de trabajo-en-grupo, propuestos para el proceso del método AGD: En Equipo / Team, Colaborativo /Collaborative, Cooperativo /Collaborative y De Control / Control.

En relación con el trabajo en grupo, los equipos que usaron RUP asignaron roles diversos a sus participantes durante las fases de “Inception” y de “Elaboration”. Dichos roles se asignaron considerando el dominio técnico (Ej. Arquitecto, Diseñador, etc.). Esto no ocurrió en las fases “Construction” y “Transition” donde sólo utilizaron los roles Desarrollador y Administrador del Proyecto. Mostrando diferencias importantes en la organización del trabajo, dependiendo de la fase realizada, redundando en mayor dificultad para la utilización del RUP.

Contrastando con los equipos que usaron AGD, pues estos durante todos los ciclos de trabajo, se responsabilizaron de un conjunto fijo de roles orientado al trabajo-en-equipo (Ej. Líder del Equipo, Administrador de Planeación, Administrador de Soporte, Desarrollador, ect.). Siempre hubo participantes atendiendo las áreas de proceso: administrativas y de soporte. Además, el desempeño del trabajo en grupo tiende a la mejora continua, pues los participantes ganan experiencia en todos los roles, intercambiándolos periódicamente.

6 Herramientas para Soportar el Proceso de Software

6.1Introducción

En un entorno para desarrollo dirigido por modelos, como CWSP, el núcleo son las herramientas de transformación de los modelos (productos-del-trabajo). En nuestro caso, transformaciones entre los modelos CIM, PIM, PSM e IM incluidos en el Conjunto de Productos del método AGD.

Aunque las herramientas de transformación son importantes no son las únicas necesarias. Se requiere otra funcionalidad (ej. una herramienta para realizar modelos y cambiarlos). En la figura 6-1 se muestran las principales herramientas incluidas en un entorno de desarrollo dirigido por modelos. A continuación describimos cada elemento:

- *Adecuación de proceso*: Ayuda a ajustar el proceso a las circunstancias reales del desarrollo (ej. alcance, recursos, requerimientos), para entregar un producto de calidad a tiempo y dentro del presupuesto acordado.
- *Registro y monitoreo del proceso*: Facilita la captura y presentación de las características del proceso realizado, y del producto que se obtiene.
- *Editor de modelo*: Soporta la construcción y modificación de modelos.
- *Herramienta de Transformación*: Realiza transformaciones verticales entre modelos (ej. modelo PIM a uno o varios modelos PSM) y transformaciones horizontales entre modelos (ej. un modelo contenido en PIM a otro dentro del mismo PIM), utilizando definiciones de transformación.
- *Validación del Modelo*: Verifica los modelos contra un conjunto de reglas (predefinidas o definidas por el usuario) para asegurar que se puedan usar en una transformación.
- Editor de Código (en un Interactive Development Environment (IDE)**)**: Funciones comunes de edición de código.
- *Generador de Código*: Lee en forma de modelo y genera el texto del archivo de código.
- *Compilación, depuración (en un IDE)*: Funciones comunes de compilar y depurar código.
- *Analizador / Parser*: Lee texto de archivo de código y lo deposita en forma de modelo, utilizable por otras herramientas.

La utilización de las herramientas (figura 6-1) se inicia con una interacción entre el responsable del desarrollo del producto de software y la herramienta que ayuda a adecuar el proceso de desarrollo. Mediante un diálogo interactivo se proporcionan las características del producto y de la organización que lo desarrollará, para adecuar el proceso de acuerdo a reglas heurísticas y los datos de experiencias de desarrollos anteriores.

•

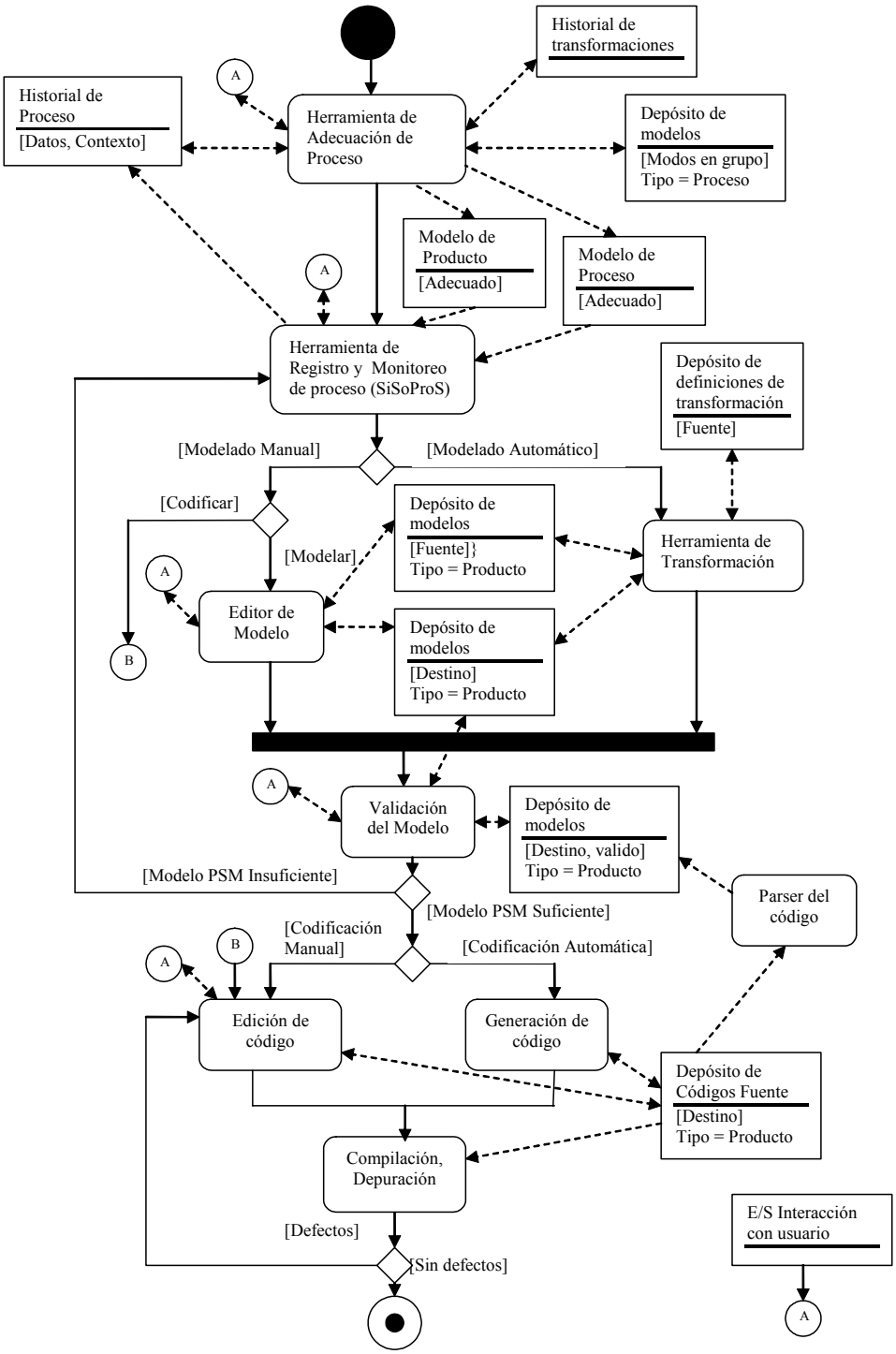

Historial de transformaciones
A
Historial de Proceso
[Datos, Contexto]
Herramienta de Adecuación de Proceso
Depósito de modelos
[Modos en grupo]
Tipo = Proceso
Modelo de Producto
[Adecuado]
Modelo de Proceso
[Adecuado]
A
Herramienta de Registro y Monitoreo de proceso (SiSoProS)
Depósito de definiciones de transformación
[Fuente]
[Modelado Manual]
[Modelado Automático]
[Codificar]
Depósito de modelos
[Fuente]}
Tipo = Producto
Herramienta de Transformación
[Modelar]
A
Editor de Modelo
B
Depósito de modelos
[Destino]
Tipo = Producto
A
Validación del Modelo
Depósito de modelos
[Destino, valido]
Tipo = Producto
[Modelo PSM Insuficiente]
Parser del código
[Modelo PSM Suficiente]
[Codificación Manual]
B
A
[Codificación Automática]
Edición de código
Generación de código
Depósito de Códigos Fuente
[Destino]
Tipo = Producto
Compilación, Depuración
E/S Interacción con usuario
[Defectos]
[Sin defectos]
A

Figura 6-1 Herramientas principales de un entorno dirigido por modelos.

A continuación se configura la herramienta para registro y monitoreo del proceso con los parámetros resultantes de la herramienta de adecuación del proceso. Estableciendo los valores propuestos para las variables que definen la estructura del producto en el conjunto de productos (PS), asi como los cuatro modos de trabajo-en-grupo del proceso MDP.

Se registra y monitorea el proceso de desarrollo que decide si el producto siguiente a elaborar se modelará en forma manual o automática. En caso de contar con la definición de la transformación requerida para obtener el modelo destino, a partir de un modelo fuente existente, será posible realizarla automáticamente utilizando la herramienta de transformación.

Una vez validado el modelo destino se evalúa si el modelo de plataforma específica esta definido hasta el detalle suficiente para poder realizar la codificación. En caso de que el modelo PSM sea suficiente se decidirá si se puede realizar la codificación en forma automática o se prefiere realizar manualmente, registrando los datos del proceso mediante la herramienta de registro y monitoreo.

En cualquiera de los casos se obtiene un código fuente que se compilará y depurará hasta obtener un producto de software libre de defectos, registrando los tiempos utilizados y los defectos encontrados mediante la herramienta de registro y monitoreo.

En las herramientas de la figura 6-1 se incluye un analizador / parser, que permite analizar sintácticamente un código fuente y da como resultado un modelo PSM, habilitando la reingeniería.

En las secciones siguientes se describe la herramienta para registro y monitoreo del proceso: Sistema de Soporte para el Proceso en Equipo (SiSoProS), que validar la utilidad de los conceptos del método AGD.

6.2Antecedentes de la Herramienta Especificada

Además de la necesidad de aplicar el método AGD, se presentó la oportunidad de usarlo en el soporte al desarrollo de software requerido por la industria del software del Estado de Morelos, México.

Los actores de la industria del software en el estado de Morelos están organizados, a partir de agosto del 2002, en la AISAC (Asociación de la Industria del Software). La misión de AISAC se orienta a establecer la estrategia para desarrollar condiciones idóneas en el macroambiente de la entidad, que permitan el crecimiento de la industria del software en el Estado de Morelos.

Los elementos que soportan las técnicas PSP (Personal Software Process) y TSP (Team Software Process) proveen un tratamiento balanceado de: el proceso, el producto y el trabajo de equipo, incorporando la experiencia industrial existente en planeación y administración de proyectos de software. En el estado de Morelos, después de una etapa de concientización, se llegó al consenso de que conviene promover la difusión y uso de estas técnicas.

Uno de los objetivos de AISAC es promover certificaciones internacionales de calidad en desarrollo de software (CMM, ISO), para sus miembros. Logrando capacitar en las técnicas PSP y TSP, con reconocimiento del SEI (Software Engineering Institute) de la CMU (Carnegie Melon University), a personal de siete empresas e instituciones.

El personal capacitado en primera instancia, a su vez, ha capacitado a personal de otras 17 empresas e instituciones afiliadas a la AISAC. En la capacitación, se han utilizado las herramientas disponibles en SEI y las disponibles como software libre, que tienen limitaciones importantes para utilizarlas en el desarrollo industrial de software.

Se usa la herramienta "Software Process Dashboard" [D06], que es una iniciativa de software-libre para crear una herramienta de soporte para PSP / TSP, aunque proporciona soporte limitado. La versión disponible de Dashboard soporta la captura de valores para las métricas usadas por PSP por un desarrollador, pero todavía no incluye la funcionalidad requerida para soportar el desarrollo en equipo y llevar un registro histórico.

Por otro lado el software de soporte usado por SEI, en sus cursos, no proporciona la funcionalidad necesaria para usarlo en el desarrollo de software en un contexto industrial. No está disponible libremente y la versión utilizada en el curso es para los programas de ejercicio, solamente.

En la práctica, los participantes de la industria se han percatado de que no existe una herramienta que soporte adecuadamente el desarrollo de productos de software, en equipo. La necesidad de soporte computacional al proceso de desarrollo esta creciendo pues la comunidad de profesionales entrenados en las tecnicas de PSP y TSP se incrementa rápidamente.

Se identificó la necesidad de mejorar el soporte para la recolección de las métricas (ej. tiempos y defectos) del desarrollo de software en equipo.

6.3Sistema de Soporte para el Proceso de Software en Equipo

El método AGD incluye en su modo de trabajo-en-grupo "Equipo", técnicas semejantes a la técnica Proceso de Software Personal (PSP) y a la técnica Proceso de Software en Equipo (TSP). Por este motivo las herramientas que soportan al AGD se pueden utilizar por personal entrenado en PSP y TSP, obteniendo ventajas adicionales.

Para satisfacer la necesidad mencionada en la sección anterior se desarrolla el Sistema de Soporte para el Proceso de Software en equipo (SiSoProS). SiSoiProS es una herramienta de registro y monitoreo del proceso (figura 6-1) que soporta a equipos de desarrollo de software para la mejora continua del proceso y del producto.

SiSoProS facilita la captura y presentación de características del proceso realizado, y del producto que se obtiene. La interfaz usada en SiSoProS es similar a la del Dashboard, proporciona mayor funcionalidad para el seguimiento del trabajo de equipos de desarrollo y se ajusta al método AGD.

Este sistema consiste en una herramienta para el control del proceso personal y en equipos de desarrollo de software. SiSoProS esta compuesto por los módulos de interfaz, de graficación, de administración de base de datos (BD) y estadístico (ver figura 6-2). El módulo BD funciona como una interfaz lógica entre los demás componentes del sistema y una base de información que contiene el historial de los datos.

MODULO DE INTERFAZ

La interfaz debe permitir la interacción para registrar y desplegar datos en formatos y scripts preestablecidos de manera textual y gráfica, así como la selección de métodos de estimación por parte de los usuarios.

MODULO DE BASE DE DATOS

Se llevará un registro por cada usuario que utilice el SiSoProS, manejando la información para las operaciones que requiere. Habilitará la organización y manejo de la información de múltiples usuarios. Expandiendo las posibilidades como una herramienta que soporta grupos de trabajo.

MÓDULO DE GRAFICACIÓN

Como parte importante dentro de las técnicas para el proceso de software personal y en equipo (similares a PSP y TSP), se encuentra la realización de gráficas, que nos permiten observar en la elaboración de los proyectos el comportamiento individual así como de uno o varios equipos de trabajo. Estas gráficas muestran información relacionada a calidad, eficiencia, tiempo de desarrollo, esfuerzo invertido, etc., del proceso de software.

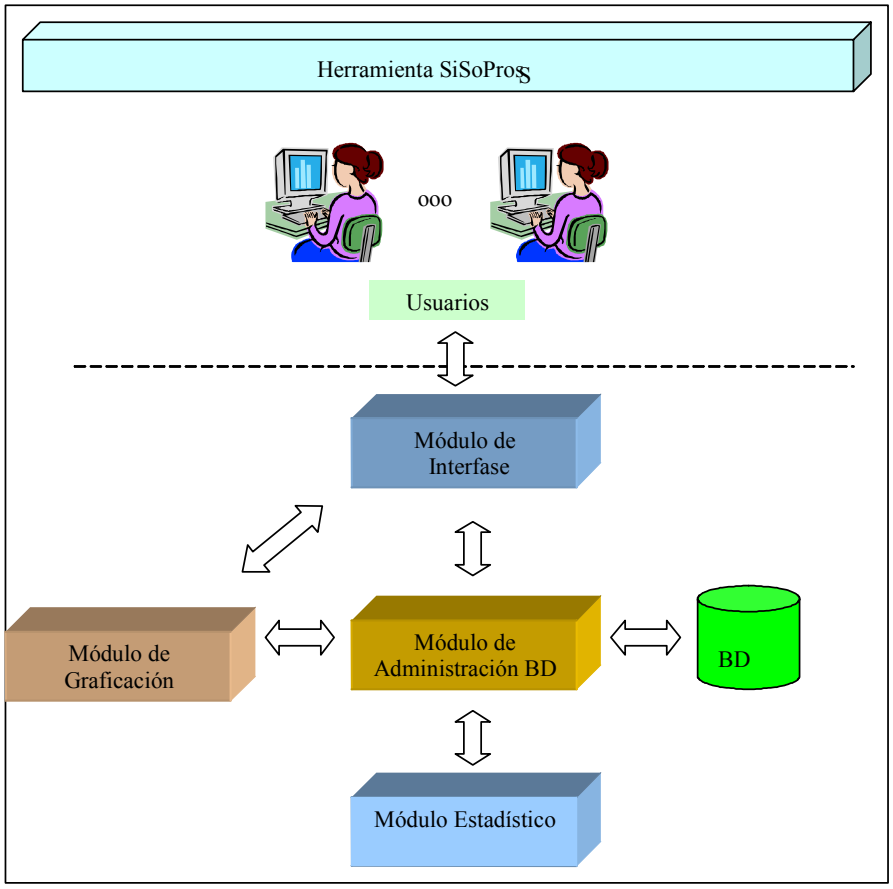


Figura 6-2 Estructura del SiSoPros.

MÓDULO ESTADÍSTICO

A partir de conjuntos de valores de las métricas recolectadas se realiza el tratamiento estadístico.

6.3.1 Interfaz Hombre Maquina

Se requiere el desarrollo de un conjunto de interfaces entre el sistema y el usuario, que le permitan: configurar el producto de software, el proceso a desarrollar; así como utilizar todas las opciones disponibles del SiSoProS.

Ventana Principal (figura 6-3):

Nombre del elemento. CPrincipal.

Descripción del propósito. Esta interfaz muestra todas las opciones de primer nivel, incluye diecisiete menús con alternativas desplegadas cuando se seleccionan, y tres botones (Terminado (producto-del-trabajo), Terminado (SubProceso) y salir).

"-DN", indican: D (Dinámico) – dependiendo de la estructura del trabajo definida previamente, no aparecer si no se especificó un ensamble/parte de este tipo; N (Nombre) - indica que aparecen los nombres de los ensamble/parte elegidos. Ejemplo: en lugar de Sistema-DN puede aparecer "Presupuestos".

Operación de la interfaz. Recibe como entradas eventos de pulsación del botón izquierdo del ratón, sobre sus elementos de menú y botones. Respondiendo con el despliegue de las opciones disponibles o de los resultados solicitados.

Estructura de la interfaz. Esta contiene los siguientes elementos:

- Barra de opciones

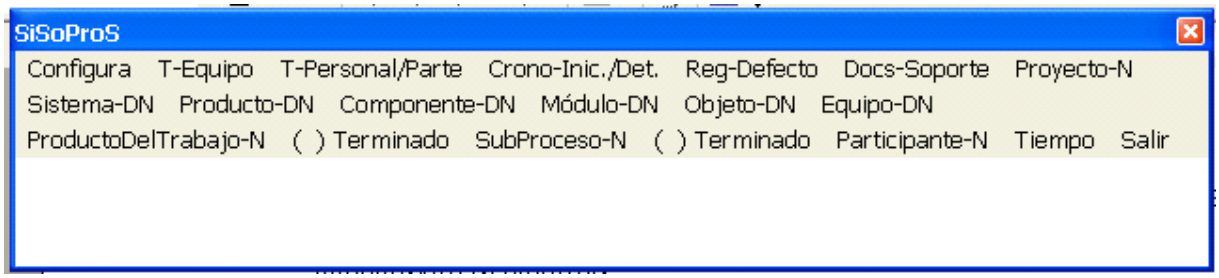


Figura 6-3 *Ventana CPrincipal*.

Configura. Define la estructura del proyecto y los sistemas a desarrollar, registro de los participantes y los equipos para llevar a cabo el desarrollo (figura 6-4).

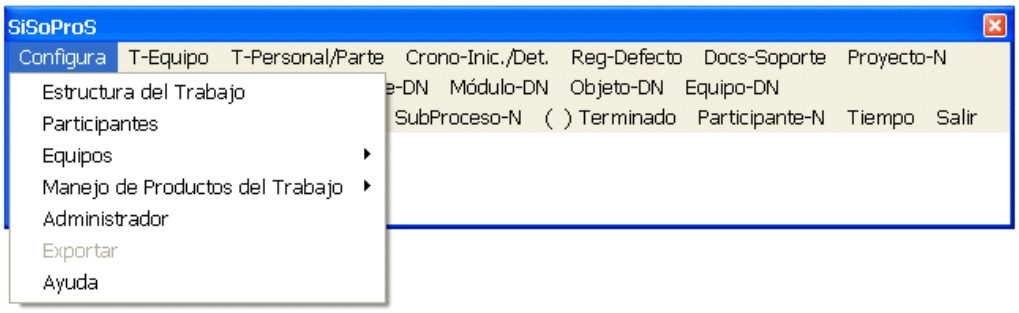


Figura 6-4 *Ventana CPrincipal, mostrando la opción de menú Configura.*

T-Equipo. Asigna los ensamble/parte a los miembros de un equipo y proporciona información acumulada en formatos preestablecidos y gráficamente (figura 6-5).

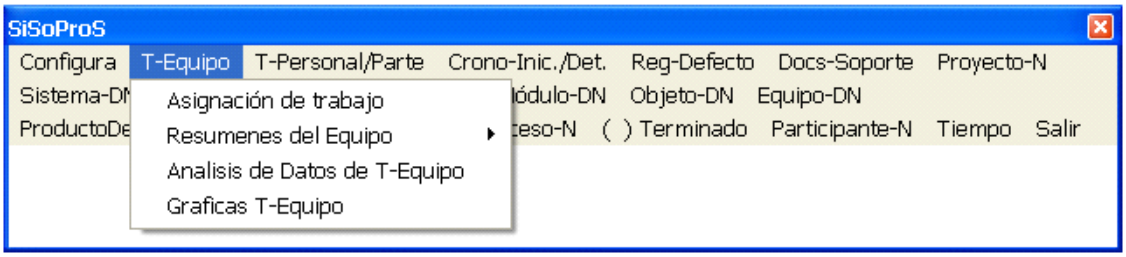


Figura 6-5 Ventana Cprincipal, mostrando el menú T-Equipo las opciones de menús y barras de herramientas.

*T-Personal/Parte***.** Permite trabajar y recibir información agrupando los datos por: participante o por ensamble/parte (figura 6-6).

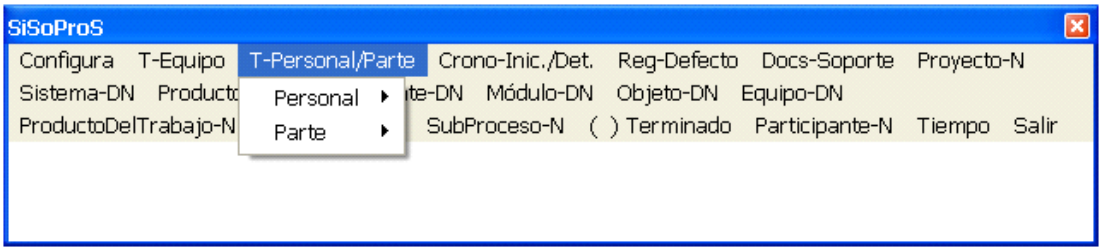


Figura 6-6 Ventana Cprincipal muestra la opción del menú T-Personal/Parte.

– *T-Personal/Parte*: Opción *Personal*. Permite escoger como se agrupan los datos presentados en las opciones de este menú: resultados agrupados por persona (figura 6-7).

–

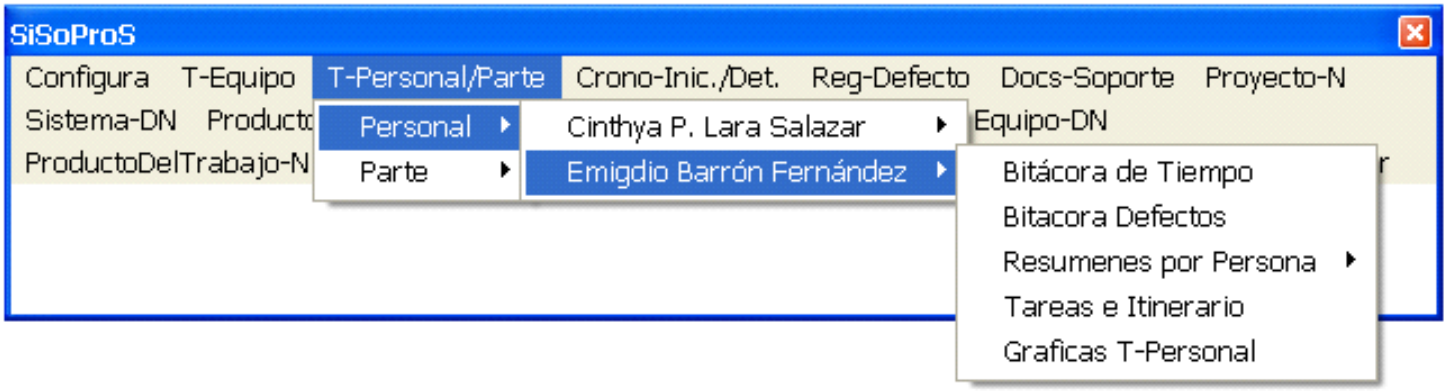


Figura 6-7 Ventana Cprincipal muestra el menú T-Personal/Parte la opción Personal y las diferentes opciones de que está compuesta.

– *T-Personal/Parte*: Opción *Parte*: Escoge como se agrupan los datos presentados en las opciones de este menú: resultados agrupados por ensamble/parte (figura 6-8).

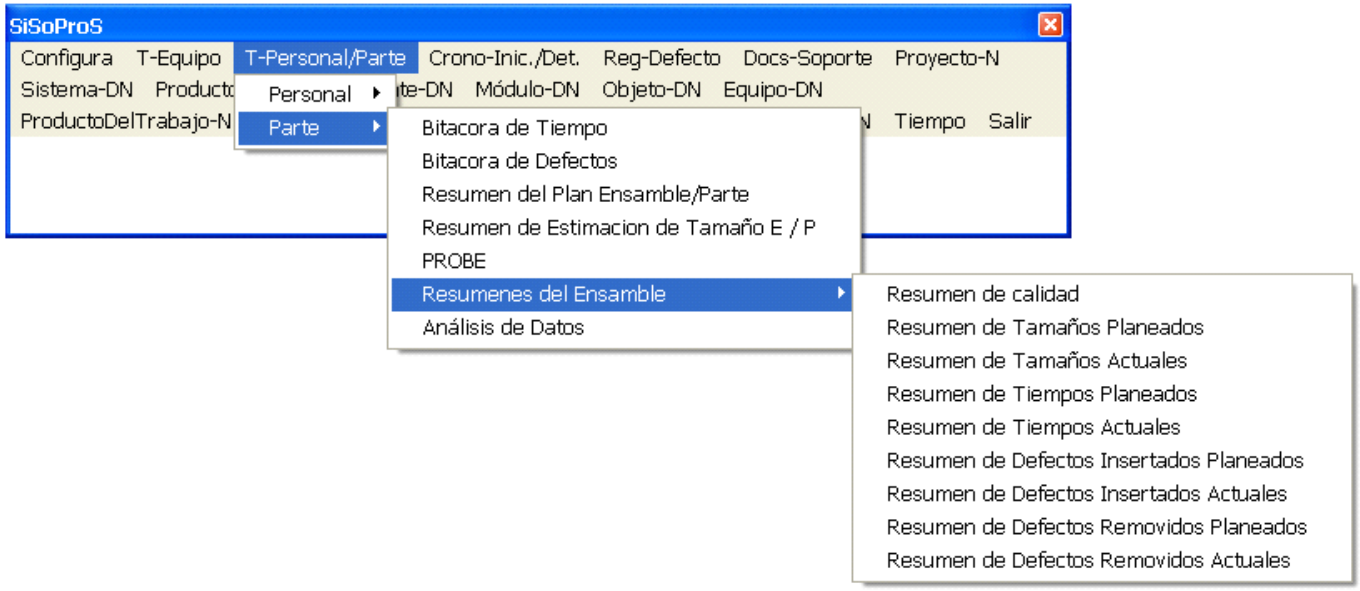


Figura 6-8 Cprincipal muestra el menú T-Personal/Parte con la opción Parte.

Cronómetro Inic. / Det. (Inicio o Paro del Cronómetro)

Para capturar el tiempo dedicado (figura 6-9). Permite iniciar o detener el cronómetro de un participante, o del grupo colaborativo, cuando se requiera. Los nombres que aparecen son de las personas asignadas al ensamble/parte correspondiente.

Figura 6-9 ***CPrincipal, mostrando el menú Crono-Inic/Det.***

*Reg-Defecto***.** Permite capturar información acerca de defectos identificados y removidos por un participante (figura 6-10).

Por ejemplo, un defecto de diseño insertado en cuando se desarrolla el diseño de alto nivel y removido durante las pruebas de integración.

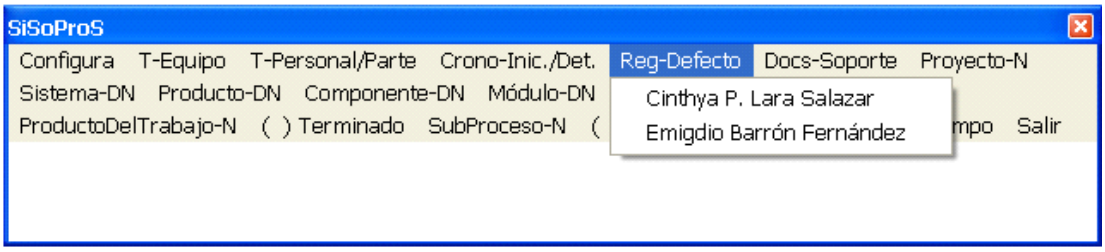


Figura 6-10 ***CPrincipal, mostrando el menú Reg-Defecto.***

La forma desplegada para que el participante específico capture la información acerca de un defecto se muestra a continuación (figura 6-11).

- **Tipo de defecto.** Se especifica el nombre del defecto encontrado.
- **Botón Defecto.** Muestra si el defecto tiene alguna relación con otro tipo de defecto**.**
- **Botón Inicio/Paro.** Inicia el tiempo de corrección del defecto y detiene el tiempo, si es que ya fue corregido el defecto.
- **Botón Terminar.** Indica la terminación de la corrección de defectos.
- **Botón Cancelar**. Cancela el proceso de correcciónde defecto.

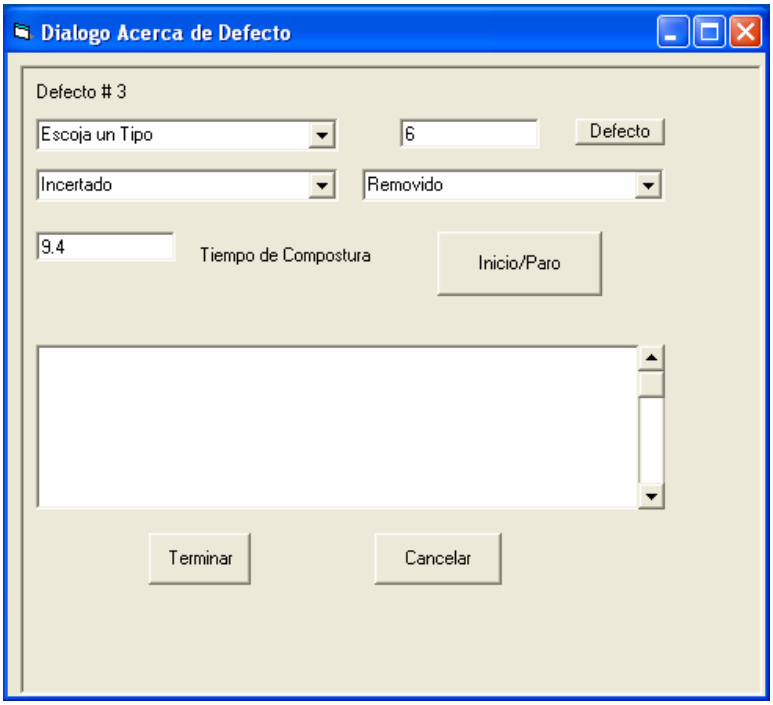


Figura 6-11 Hoja de diálogo de la información capturada acerca de un defecto.

Docs-Soporte. Consulta y utilización de los Scripts, ProductosDelTrabajo, Sub_procesos, Formas y Roles (figura 6-12).

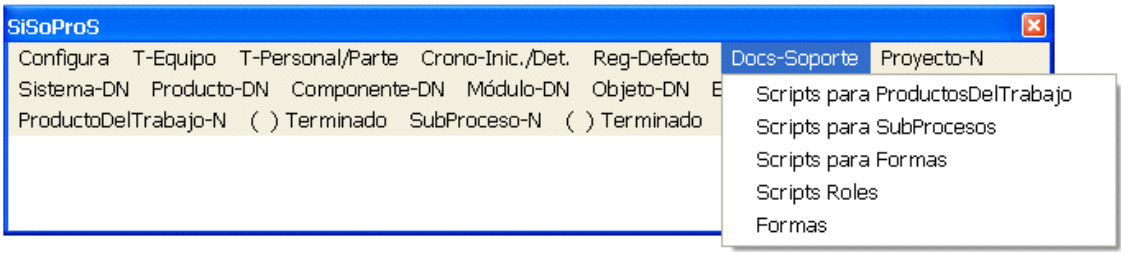


Figura 6-12 *Ventana CPrincipal mostrando el menú Documentos de Soporte.*

*Proyecto-N***.** Muestra el nombre (N) de los proyectos registrados, para seleccionar el que se está trabajando (figura 6-13).

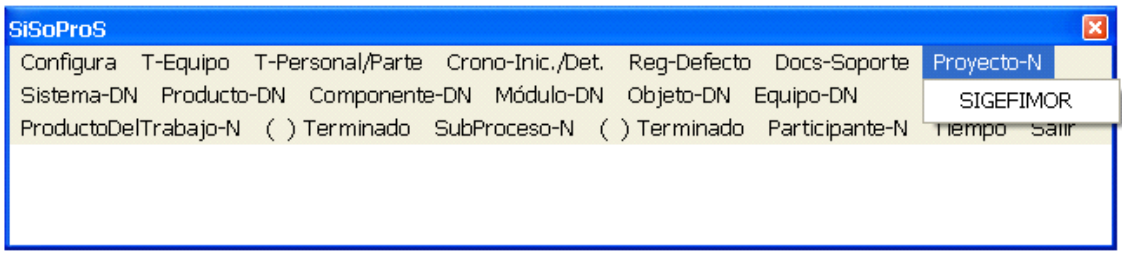


Figura 6-13 Ventana CPrincipal, mostrando ejemplo de un proyecto en el menú Proyecto-N.

Los siguientes menús aparecerán de acuerdo a la "Estructura del Trabajo" que se haya especificado en el menú Configura y la selección que haga el usuario del SiSoProS.

Sistema-DN. Muestra el nombre (N) del sistema seleccionado (del menú sistemas), que es parte del proyecto. Si no se ha configurado algún sistema no aparecerá este menú ni ninguno de los menús "DN" siguientes (figura 6-14).

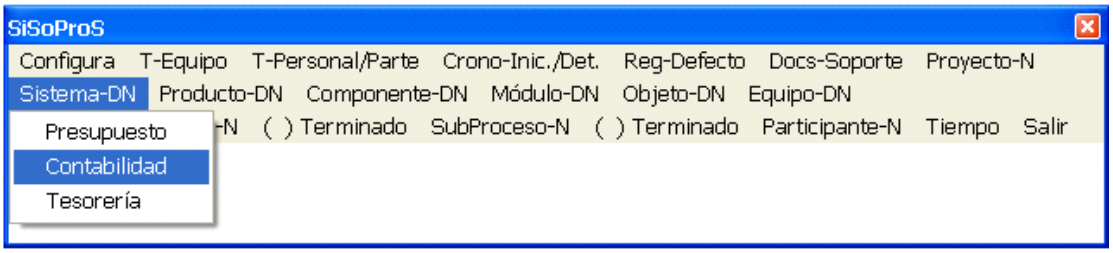


Figura 6-14 *Ventana CPrincipal*, mostrando ejemplos de nombres de sistemas que contiene el menú Sistema-DN.

*Producto-DN***.** Muestra el nombre (N) producto seleccionado (del menú de productos), que es parte del ensamble sistema, mostrado a la izquierda. Si no se ha configurado algún producto no aparecerá este menú ni ninguno de los menús "DN" siguientes (figura 6-15).

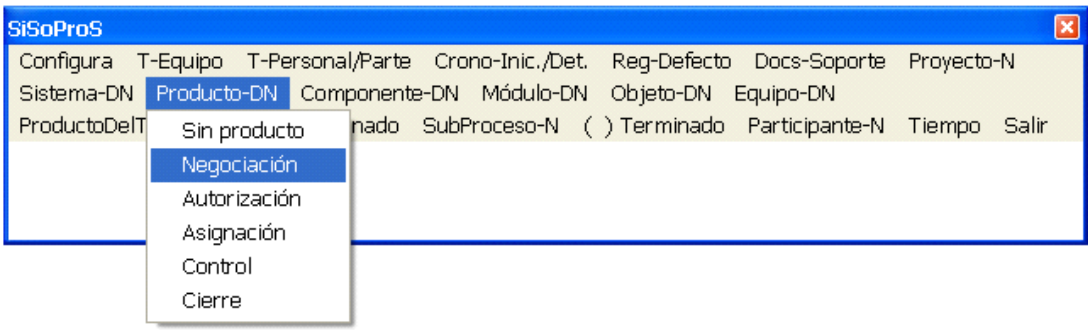


Figura 6-15 Ventana Principal, mostrando ejemplo de un producto en el menú Producto-DN.

*Componente-DN***.** Muestra el nombre (N) de un componente seleccionado (del menú de componentes), que es parte del ensamble producto, mostrado a la izquierda (figura 6-16). Si no se ha configurado algún componente no aparecerá este menú ni ninguno de los menús "DN" siguientes.

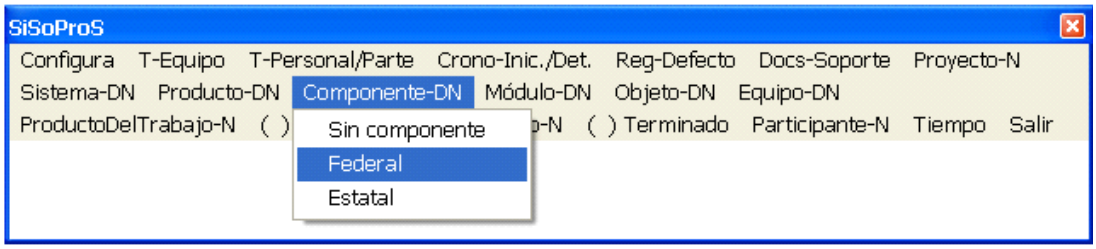


Figura 6-16 Ventana Principal, mostrando ejemplo de un componente en el menú Componente-DN.

*Módulo-DN***.** Muestra el nombre (N) del módulo seleccionado (del menú de módulos), que es parte del ensamble componente, mostrado a la izquierda (figura 6-17). Si no se ha configurado algún módulo no aparecerá este menú ni ninguno de los menús "DN" siguientes.

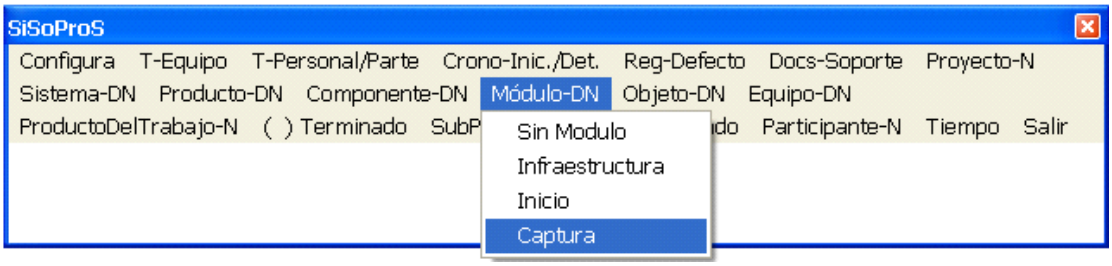


Figura 6-17 Ventana Principal, mostrando ejemplo de un módulo en el menú Módulo-DN.

*Objeto-DN***.** Muestra el nombre (N) del objeto seleccionado (del menú de objetos), que es parte del ensamble módulo, mostrado a la izquierda (figura 6-18). Si no se ha configurado algún objeto no aparecerá este menú.

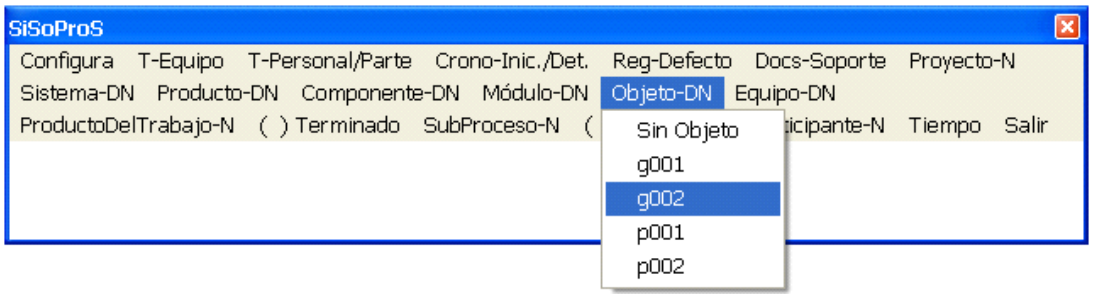


Figura 6-18 Principal, mostrando ejemplo de un objeto en el menú Objeto-DN.

*Equipo-DN***.** Muestra el nombre (N) de los equipos que están almacenados en el sistema y permite elegir el equipo en el cual se desea trabajar (figura 6-19). Si no se ha configurado algún equipo y asignado el ensamble/parte, identificado mediante los menús anteriores, no aparecerá este menú.

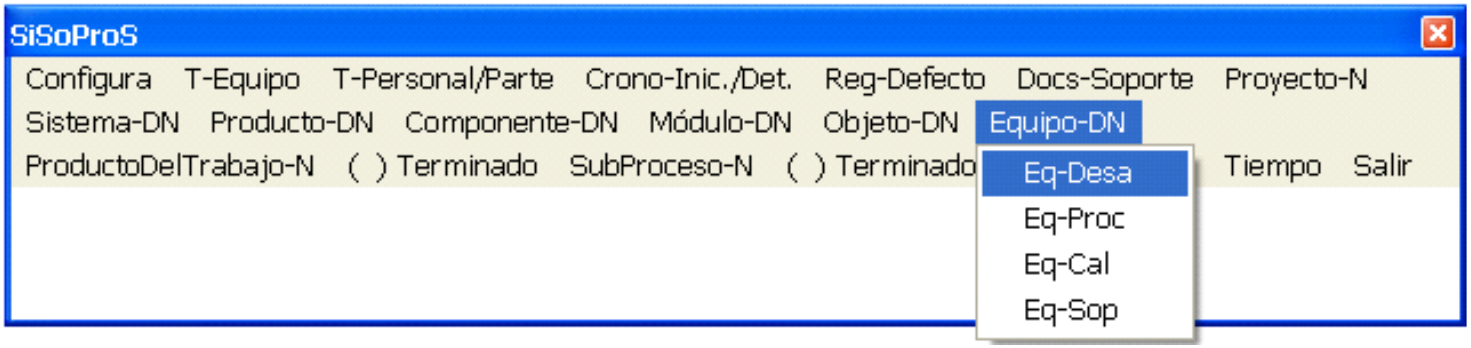


Figura 6-19 Ventana Principal, mostrando ejemplo de un objeto en el menú Objeto-DN.

*ProductoDelTrabajo-N***.** Muestra el nombre (N) del producto del trabajo seleccionado (del menú de ProductosDelTrabajo) (figura 6-20). El menú muestra el conjunto de productos que se deben realizar para la parte más baja de la jerarquía de ensamble/parte, mostrada a la izquierda.

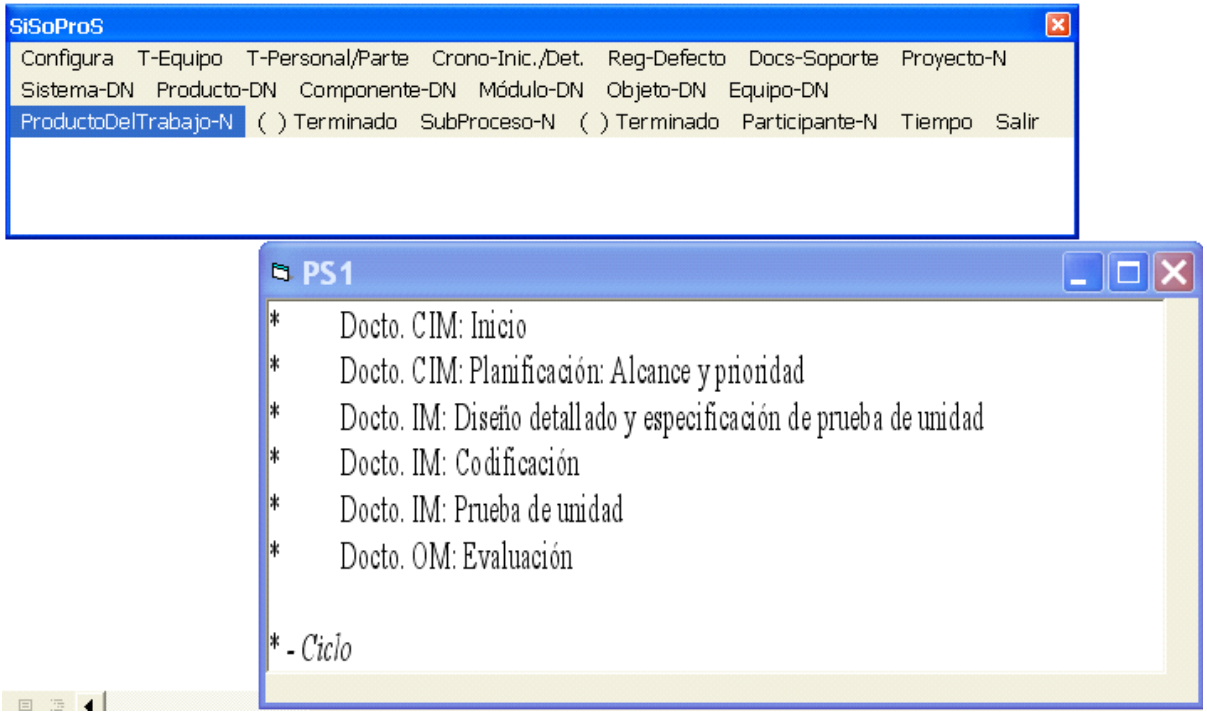


Figura 6-20 Ejemplo de la información contenida en la opción ProductoDelTrabajo-N.

*( ) Terminado***.** Se pone una marca indicando que el producto (a la izquierda) se terminó (figura 6-21).

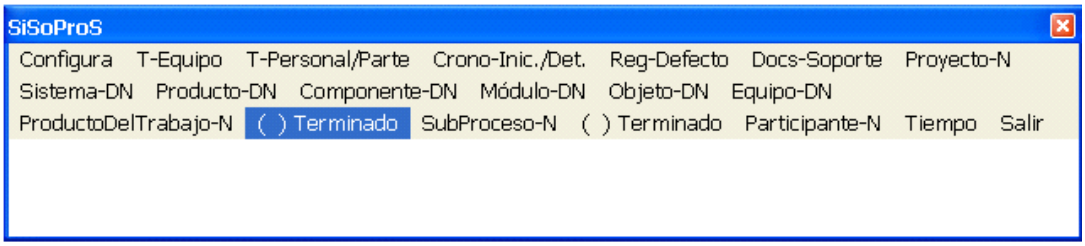


Figura 6-21 ***Ventana Cprincipal muestra la opción ( ) Terminado.***

SubProceso-N. Son subprocesos de un patrón de proceso propuesto (figura 6-22).

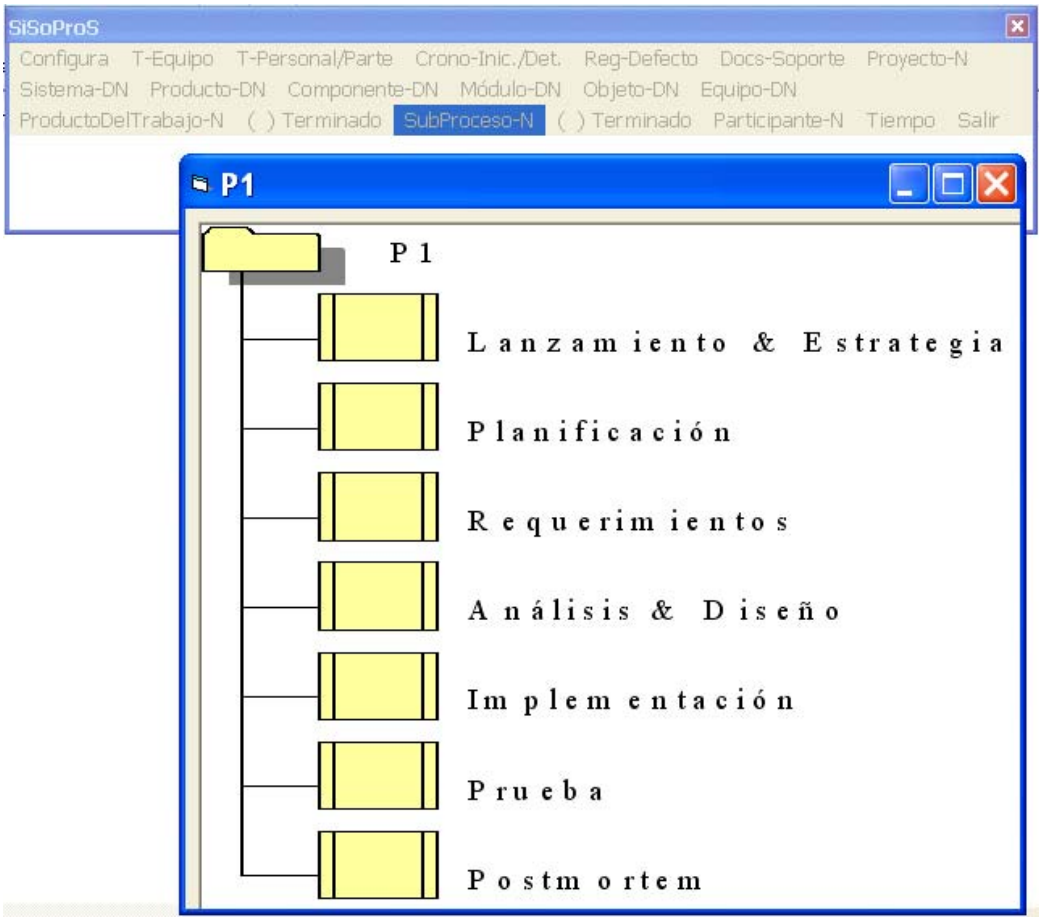


Figura 6-22 I*nformación contenida en la opción SubProceso-N.*

*( ) Terminado***.** Se pone una marca indicando que el sub-proceso (a la izquierda) se terminó.

*Participante-N***.** Muestra el nombre del participante que está asignado al ensamble/parte elegido mediante los menús anteriores (figura 6-23).

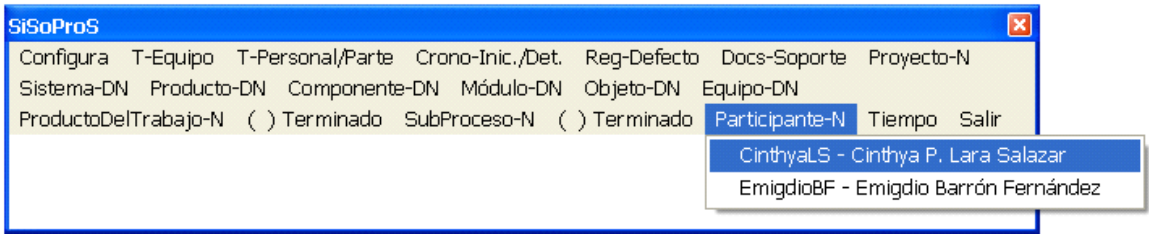


Figura 6-23 ***Cprincipal muestra la opción Participante-N.***

*Tiempo***.** Muestra la unidad de tiempo en que manejará el tiempo de trabajo (figura 6-24).

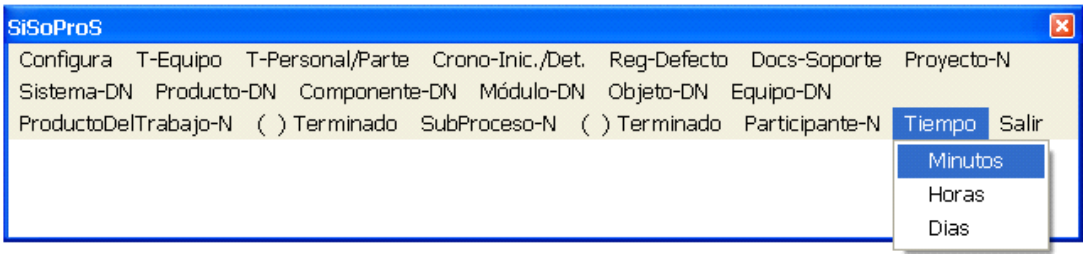


Figura 6-24 Ventana Cprincipal muestra Tiempo en que medida se pude elegir para trabajar.

Salir. (figura 6-25)

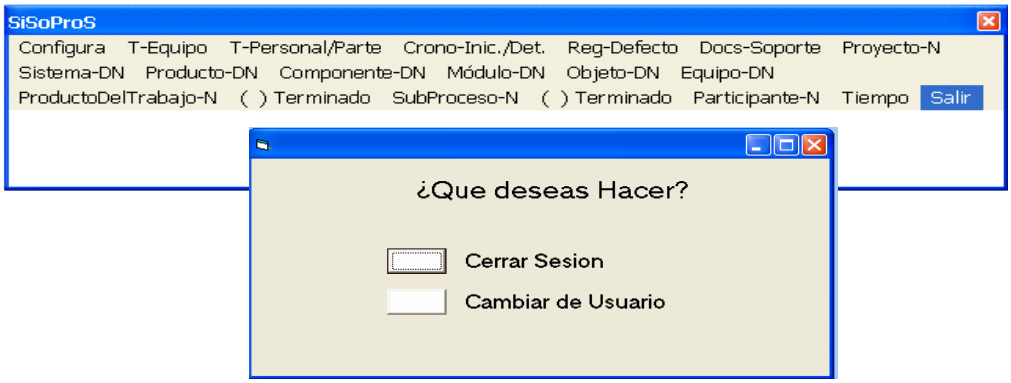


Figura 6-25 Muestra la interfaz de Salida en la que el usuario tiene a elegir dos opciones.

*Cerrar Sesión***.** Salir del sistema.

Cambiar de Usuario. Permite volver a acceder al sistema, para cambiar el usuario que trabaja.

El prototipo de la interfaz, muestra ejemplos de las pantallas que se desplegarán, dicho prototipo se incluye en el CD anexo a esta tesis. Para desarrollar el prototipo se elaboraron los documentos: Especificación de Requerimientos y Especificación de Diseño General, que se utilizarán para el desarrollo del SiSoProS funcional.

6.4Los Productos en SiSoProS

6.4.1 Patrones del Conjunto de Productos

Los desarrolladores de software, capacitados en la técnica Personal Software Proicess (PSP), están acostumbrados a su estrategia con un marco de madurez parecido al del CMMI. Con una progresión de etapas para el PSP, que se muestra en la figura 6-26, donde se incluyen en cada paso las prácticas que se deben agregan para mejorar el proceso de desarrollo. Así, en la etapa PSP0.1 se adicionan a las prácticas de la etapa PSP0, las prácticas mostradas: Uso de norma de codificación, Medición del tamaño y Propuesta de mejora del proceso.

Un marco de madurez similar, se especificó para el SiSoProS. Donde se puede elegir un conjunto de productos específico para cada etapa de evolución, en forma semejante a la progresión de etapas en PSP, como se muestra en la tabla 6-1. De esta manera los desarrolladores, usuarios del PSP, tienen la facilidad de elegir el nivel de evolución deseado para su trabajo de desarrollo.

Se puede observar en la tabla 6-1, los patrones de conjunto de productos y sus pasos equivalentes del PSP. PS1 equivale al PSP1.1, PS2 equivale al PSP2.1 y PS3 equivale al PSP3.

SiSoProS cuenta con cinco patrones de conjuntos de productos del PS1 al PS5, como se muestra en la tabla 6-1. PS4 y PS5 incluyen productos requeridos a nivel gerencial y de la alta dirección, para describir el negocio, y no tienen un nivel correspondiente en PSP.

De acuerdo a la preferencia del usuario-desarrollador, existe la facilidad de definir otros patrones de conjuntos de productos, en función de las condiciones en curso, el contexto de aplicación del software y las experiencias previas. Los patrones adicionales se nombran PSn, con n >5.
PSn, para n > 5: Patrones de usuario
El usuario de SiSoProS puede definir patrones de productos diferentes, de acuerdo a sus necesidades. Se podría tener un patrón para los productos del MOPROSOFT, si se requiere apegarse a esta norma. Otra posibilidad sería crear un patrón de los productos usados en el RUP. Así, la creación de patrones del usuario permitiría trabajar con el conjunto de productos deseado, utilizando el proceso MDP de AGD.
A continuación se describen los patrones de conjuntos de productos, predefinidos en SiSoProS:

PS5: Patrón para conjunto de productos modelando un negocio

Este patrón de productos no requiere de ningún producto previo, para iniciar. Comienza con análisis y modelado a nivel del negocio e incluye los modelos necesarios para la generación del documento de Requerimientos del Sistema, independiente de cómputo. Contiene además los productos del patrón SP4.

PS4: Patrón para conjunto de productos de un software

Los productos del patrón, se realizan a partir de los requerimientos del sistema, independientes de cómputo y considerado fuera del patrón. Los productos incluidos en este patrón son producto de transformaciones a representaciones independientes de plataforma que conformarán el documento de especificación de requerimientos de software. Contiene los productos del patrón PS3.

PS3: Patrón del conjunto de productos realizado incrementalmente

Se utiliza la abstracción, la estrategia consiste en particionar un programa (ensamble) en partes del tamaño del conjunto de productos del patrón PS2. El primer conjunto de productos entregado es un módulo base o núcleo el cual se mejora incrementalmente en iteraciones sucesivas. Cada iteración realiza un conjunto de productos PS2 completo, incluyendo resultados de diseño, codificación, compilación y prueba.

Los productos de PS3 se realizan a partir de los requerimientos del software, independiente de plataforma de desarrollo. Sus productos integran el documento de diseño general, donde se identifican claramente los ensambles y las partes que constituyen al producto de software.

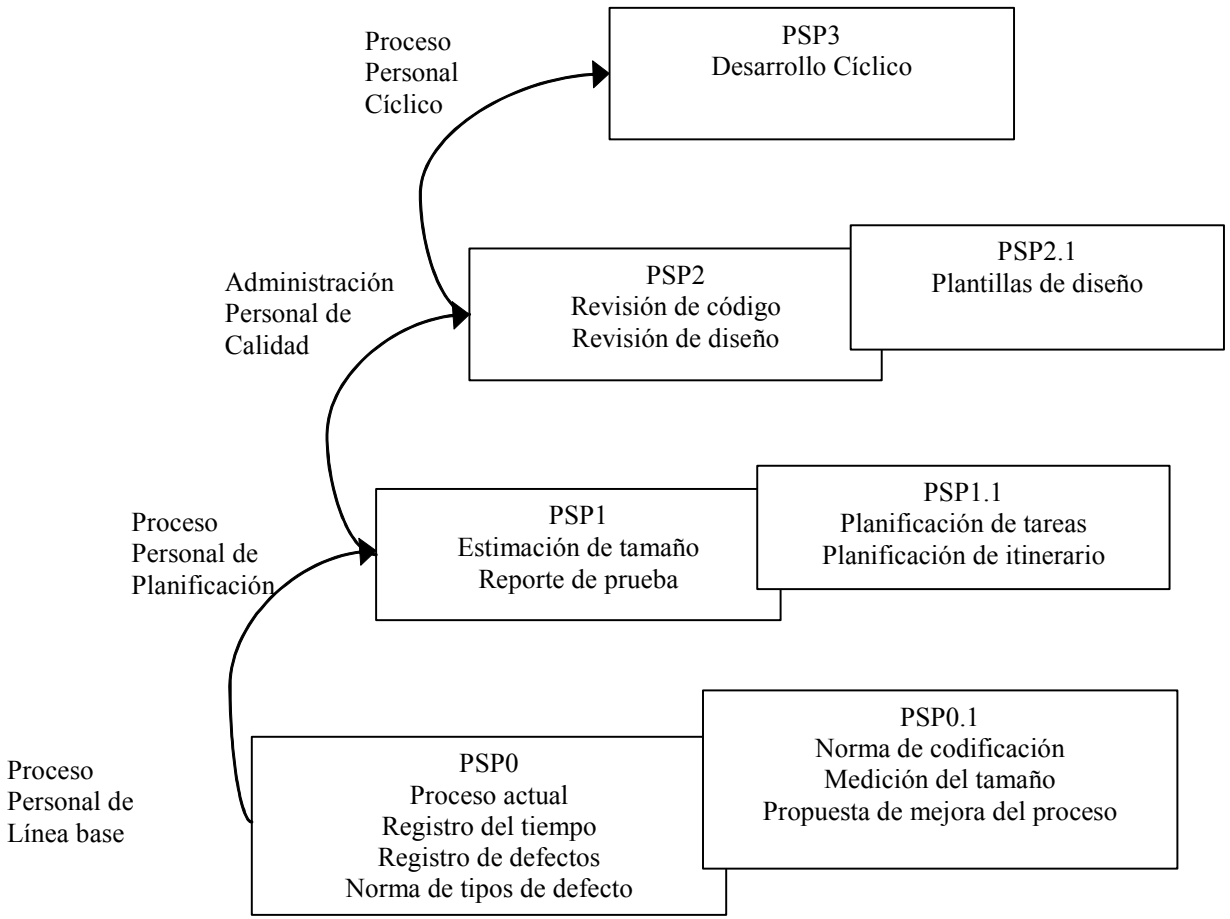


Figura 6-26 Evolución del PSP

Tabla 6-1 Patrones de Productos para SiSoProS

PS1	PS2	PS3	PS4	PS5	*Productos (PS2Set)*
*	*	*	*	*	*CIM: Docto. Inicialización*
*	*	*	*	*	*CIM: Docto. Planificación: Alcance y prioridad*
				*	*CIM: Docto. Negocio existente*
				*	*CIM: Docto. Identificación de componentes*
				*	*CIM: Docto. Prototipo del sistema*
				*	*CIM: Docto. Requerimientos del sistema*
				*	*CIM: Docto. Revisión de requerimientos y prototipo del sistema*
			*	*	*PIM: Docto. Ingeniería inversa*
			*	*	*PIM: Docto. Ingeniería directa*
			*	*	*PIM: Docto. Prototipo del software*
			*	*	*PIM: Docto. Revisión de prototipo*
			*	*	*PIM: Docto. Requerimientos de software y especificación de prueba de aceptación*
			*	*	*PIM: Docto. Revisión de requerimientos del software*
		*	*	*	*PSM: Docto. Restricciones a requerimientos*
		*	*	*	*PSM: Docto. Arquitectura de software (HLD) y especificación de prueba de integración*
		*	*	*	*PSM: Docto. Revisión de arquitectura de software (HLD)*
*	*	**	**	**	*IM: Docto. Diseño detallado y especificación de prueba de uni-*

					dad
	*	**	**	**	*IM: Docto. Revisión de diseño detallado*
*	*	**	**	**	*IM: Docto. Codificación*
	*	**	**	**	*IM: Docto. Revisión de código*
*	*	**	**	**	*IM: Docto. Prueba de unidad*
		*	*	*	*IM: Docto. Integración y prueba de Integración*
				*	*OM: Docto. Identificación de recursos*
				*	*OM: Docto. Distribución*
				*	*OM: Docto. Prueba de aceptación*
*	*	*	*	*	*OM: Docto. Evaluación*

* - Productos para un ensamble, realizables incrementalmente.
** - Productos realizados en iteraciones, para cada parte que compone a un ensamble.

PS2: Patrón para conjunto de productos de calidad

Los productos de este patrón, también se realizan a partir del diseño general para plataforma específica, que no se considera parte del patrón. Se añaden resultados de las revisiones, que ayudan a encontrar defectos en los primeros productos realizados, cuando es más económico repararlos. Se recolectan y analizan los defectos encontrados durante la compilación y las pruebas. Con estos datos se establecen listas de verificación y se realiza un proceso personal de aseguramiento de calidad. Se incluyen criterios de completitud del diseño y se aplican técnicas para verificar y asegurar la consistencia del diseño.

PS1: Patrón para conjunto de productos base y de planificación

Los productos de este patrón, se realizan a partir del diseño general para plataforma específica, que no se considera parte del patrón. PS1 contiene productos necesarios para establecer algunas mediciones básicas y un formato para reportar. Para obtener los productos se sigue el proceso en curso de desarrollo, mejorado mediante la inclusión de mediciones. Utilizando este patrón de productos, se pueden seguir procesos equivalentes a los procesos PSP: PSP0, PSP0.1, PSP1 o PSP1.1.

6.4.2 Patrones para structura del ensamble/parte

SiSoProS cuenta con cinco patrones de estructura del sistema. Un ejemplo en el que utilizamos el patrón con cinco niveles S5 (figura 6-27), muestra un sistema constituido por otros cuatro niveles de elementos: producto, componente, módulo y objeto. La tabla 6-2 muestra los cinco patrones de estructura de ensamble y los elementos que los constituyen.

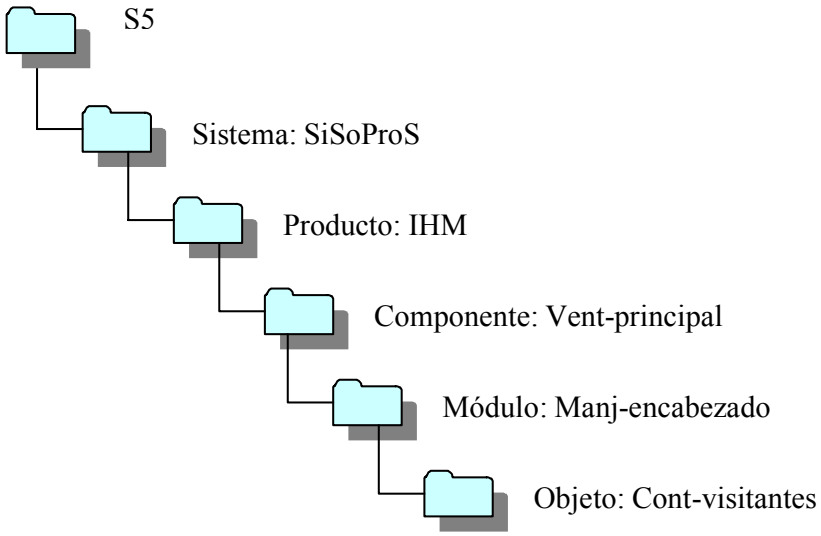


Figura 6-27 Patrón S1 de estructura del sistema

AGD proporciona la facilidad de definir otros patrones, dadas las condiciones del problema, el contexto en que funcionará el software y la experiencia previa de desarrollo. Los nuevos patrones pueden asignar otros nombres a cada ensamble/parte en la jerarquía, así como aumentar o disminuir el número de niveles jerárquicos.

En el patrón S1 un ensamble sistema consiste de varias partes producto; los ensambles producto incluyen varias partes componente; los ensambles componente están formados por varias partes módulo; y los ensamble módulo consisten de varias partes objeto.

Tabla 6-2 Patrones de Productos para SiSoProS

S1	S2	S3	S4	S5	*Ensamble/Parte*
				*	*Sistem*
			*	*	*Producto*
		*	*	*	*Componente*
	*	*	*	*	*Módulo*
*	*	*	*	*	*Objeto*

* - Ensamble/parte contenida en el patrón.

6.5Los Procesos en SiSoProS

6.5.1 Modo Colaborativo para trabajo-en-grupo

Se puede capturar, acumular y reportar el tiempo y los defectos, medidos en el trabajo colaborativo de los participantes, en un ensamble/parte seleccionado. Permitiéndo también capturar, acumular y reportar el tiempo y los defectos, medidos en el trabajo de una persona, para varios ensamble/parte. Esta funcionalidad se utiliza mediante el menú *T Personal/Parte* como se muestra en la figuras 6-6 a 6-8 y los menús *Cron_Inic/Det., Reg-Defect,* mostrados en las figuras 6-9 a 6-11 de § 6.3.1.
Para el trabajo Colaborativo no colocalizado (ej. participantes en diferentes ubicaciones, trabajando en PCs conectadas a una red), se requiere tener la facilidad de trabajar en un espacio de almacenamiento compartido y ocupar alguna herramienta con características “lo Que Tu Ves es lo Que Yo Veo (QTVQYV) / What You See is What I See (WYSWIS)”. SiSoProS tendrá acceso a un almacenamiento compartido (ej. almacenamiento donde reciden archivos usados por varios participantes), mediante la interfaz de la figura 6-28, seleccionando la opción *Explorar Carpeta de Proyecto*.

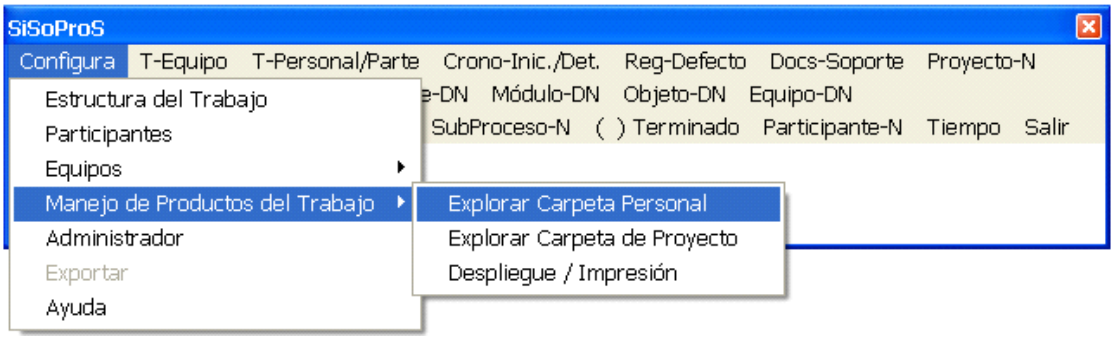


Figura 6-28 Ventana Cprincipal muestrando el menú Configura y la opción Manejo de productos del trabajo.

Para el caso de la realización de trabajo individual (considerando un tipo de trabajo colaborativo en el que sólo interviene el rol conductor / driver), se usa la opción *Explorar Carpeta Personal*. Pudiendo utilizar el área de almacenamiento asignada específicamente al participante.

6.5.2 Modo Cooperativo para trabajo-en-grupo

El trabajo cooperativo en SiSoProS se efectúa durante las revisiones formales realizadas mediante un equipo de revisión. Los productos de este tipo de trabajo están considerados dentro del conjunto de productos PS y se tienen formas predefinidas. Se puede realizar en forma no colocalizada usando el almacenamiento compartido del proyecto, mediante la opción *Explorar Carpeta del Proyecto*, de la interfaz mostrada en la figura 6-28.

6.5.3 Modo de Control para trabajo-en-grupo

Se realizar en una reunión del grupo semanalmente, programada de duración preestablecida, produciendo un reporte con comentarios, acuerdos y decisiones. Para realizar la reunión y producir el reporte se utilizan las formas predefinidas y los reportes acumilados a nivel personal, ensamble/parte y equipo.
Para acceder a instructivos de uso de las formas se selecciona el menú *Docs-Soporte* de la figura 6-12 y para capturar, acumular y reportar las métricas se seleccionan las opciones en los menús: *Crono Inic./Det.* de la figura 6-9, *Reg-Defecto* de las figuras 6-10 y 6-11, *T-Equipo* de la figura 6-5, y *T-Personal/Parte* de las figuras 6-10 y 6-11, en la § 6.3.1.

6.5.4 Modo en Equipo para trabajo-en-grupo

Este modo de trabajo en grupo proporciona la dinámica básica para el desarrollo con calidad de cualquier producto incluido en PS. SiSoPros sugiere un patrón de subprocesos P1 (por defecto), mostrado en la figura 6-29, que se utilizará para obtener cualquier producto asociado a un ensamble/parte. Este patrón de proceso considera fijos únicamente los sub-procesos primero Lanzamiento & Estrategia y último Postmortem. Los cinco intermedios pueden realizarse en el orden que se requiera, efectuando todos o justificando las omisiones.
Existe la facilidad de definir uno o varios patrones alternos, asignados a cualquier ensamble/parte, similar a la definición de patrones adicionales del PS: para estructura del sistema (§ 6.4.2) y patrones para conjunto de productos (§ 6.4.1).
SiSoProS también proporciona ayudas mediante *Scripts para Subprocesos* y *Scripts para ProductosDelTrabjo*, en el menú *Docs-Soporte* de la figura 6-12.

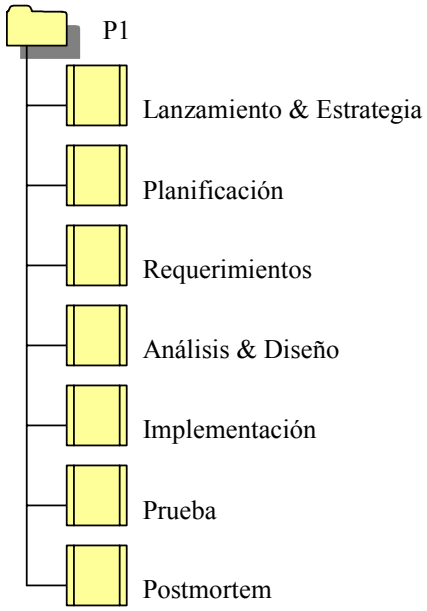


Figura 6-29 Patrón de subprocesos de un proyecto que utiliza MDP.

6.6Conclusión

El sistema SiSoProS abarca la funcionalidad de una herramienta de registro y monitoreo del proceso. Este sistema facilita la captura y presentación de características del proceso realizado, y del producto que se obtiene. SiSoProS se configura con los valores propuestos para las variables que definen la estructura del producto en el conjunto de productos (PS), así como los cuatro modos de trabajo-en-grupo del proceso MDP.

Mediante los conceptos del método AGD se complementa la interfaz hombre máquina de la herramienta "Software Process Dashboard" [D06], agregando componentes visuales que permite proporcionar la funcionalidad requerida para soportar el desarrollo en equipo y llevar un registro histórico.

Mediante el diseño del SiSoProS se valida la utilidad del método AGD, que esta alineado con el estado del arte del desarrollo de software iterativo incremental y ágil. La implementación del SiSoProS producirá una herramienta que soporte adecuadamente el desarrollo en equipo de los productos de software, para la comunidad de profesionales entrenados en las técnicas de PSP y TSP que se incrementa rápidamente en el Estado de Morelos México.

7 Conclusiones

La principal contribución de nuestro trabajo de investigación es haber creado el método de Desarrollo de software Arquitectónico en Grupo (AGD), que asigna la mayor importancia al producto de software y su estructura, asociando un proceso de grupo, a cada uno de los productos-del-trabajo requeridos para construir el resultado. Validamos nuestro enfoque utilizándolo en un ambiente experimental en el que comparamos los resultados de utilizar AGD contra los de usar el Proceso Unificado de Racional (RUP).

Revalidamos los conceptos del método AGD usándolos en el diseño e implementación de una herramienta de registro y monitoreo del desarrollo, que soporta a equipos de productores de software para la mejora continua del proceso y del producto: el Sistema de Soporte para el Proceso de Software en equipo (SiSoProS).

AGD consta de dos elementos principales: un Conjunto de Productos / Product Set (***PS***) y un Proceso Dirigido por Modelos / Model Driven Process (**MDP**); utiliza la arquitectura del producto y el PS para asociar a cada uno de los productos un proceso realizado mediante cuatro modos de trabajo-en-grupo: Equipo, Colaborativo, Cooperativo y Control.

Definición PS para asegurar que el conjunto productos-del-trabajo abarca la mayoría de las necesidades del desarrollo de software pues considera los productos de las cuatro categorías de áreas de proceso que incluye la integración de modelos de madurez de capacidad (CMMI): 1) Administración de Proceso, 2) Administración de Proyecto, 3) Ingeniería y 4) Soporte.

Diseñamos la categoría de Ingeniería, con el enfoque dirigido a modelos de la Arquitectura Dirigida por Modelos (MDA). Utilizamos las representaciones necesarias para desarrollar el producto de software, agrupadas de acuerdo a los tres conjuntos principales de MDA: Modelos Independientes del Cómputo (CIM), Modelos Independientes de la Plataforma (PIM) y Modelos de Plataforma Específica (PSM), complementándolos con los conjuntos: Modelos de Implementación (IM) y Modelos para Operación (OM).

Inicialmente los productos-del-trabajo de la categoría Ingeniería se obtienen mediante el proceso MDP, pero con el enfoque del desarrollo dirigido por modelos se podrá elegir obtenerlos automáticamente o mediante MDP, conforme habilitemos transformaciones.

Especificamos MDP para realiza en equipo sub-procesos: lanzamiento & estrategia, planificación, requerimientos, análisis & diseño, implementación, prueba y pos-término. Iterándolos continuamente de manera flexible, para obtener cada producto-del-trabajo. El software lo obtenemos incrementalmente en forma disciplinada, monitoreándose peculiaridades del producto y características del proceso de manera semejante a como se realiza en el "Proceso de software en equipo" (TSP). El método AGD establece que el desarrollo se lleva acabo en forma: grupal, incremental, cooperativa, adaptable y directa.

La validación la logramos experimentando mediante réplicas del desarrollo de un sistema para manejo del acervo de una biblioteca (Problema modelo del área de arquitecturas de software [SG95A]), utilizando AGD y RUP, comprobando la siguiente hipótesis de trabajo:

> *"Se requiere un método cuyo proceso se dirige a desarrollar los componentes de la arquitectura de software, mediante modos de trabajo-en-grupo, para obtener productos de software de mejor calidad (con menos defectos)."*

Determinamos con la experimentación realizada que utilizando el método AGD, se obtiene un producto y un proceso de calidad igual o mejor que los obtenidos al utilizar RUP. Observando una disminución de defectos en el producto, y un proceso eficaz, debido principalmente a la mayor atención en el producto software y a que las actividades se realizaron mediante los modos de trabajo-en-grupo establecidos.

7.1 Contribuciones

En el desarrollo de esta investigación hemos identificado dos áreas prioritarias: el conjunto de productos (PS) y el proceso dirigido por modelos (MDP). Describimos a continuación los logros obtenidos, asociándolos con los requisitos que satisfacen.

7.1.1 Conjunto de Productos

Incluimos en el modelo de referencia CWSLR, además de los elementos de otros modelos (Ej. RM-ODP [ASF04] [BD01] [ODP98]), las siguientes áreas de interés del dominio del trabajo cooperativo y las aplicaciones computacionales contemporáneas:

- Organización de entradas funciones y salidas integrando procesos;
- Memoria organizacional o de grupo acerca del desempeño y decisiones dentro del proceso;
- Diálogo acerca del proceso actual y memoria organizacional;
- Establecimiento de normas para el desempeño deseado de la organización;
- Reglas y servicios para cooperar en la resolución de conflictos y
- Coordinación de la organización para facilitar el trabajo cooperativo

Esta decisión ocasionó la existencia de varios elementos del conjunto de productos, mismos que el desarrollador deberá decidir explícitamente si se elaborará o no. Los beneficios mencionados los logramos satisfaciendo la siguiente necesidad:

*Conceptos de grupo***:** Se requiere incluir los elementos necesarios (ej. memoria de grupo, cooperación), para soportar el trabajo en grupo que se realiza en las organizaciones.

Ajustamos el método AGD a la arquitectura del software usando como base al modelo de referencia para sistemas de información que soportan la cooperación "Trabajo cooperativo: almacenamiento, lógica y recursos" CWSLR (anexo B) y además de considerar a la "Arquitectura Dirigida a Modelos" MDA [MM03]. Ayudándonos a trabajar áreas y elementos existentes en las modelos mencionados, que de otro modo podrían no haberse considerado en el producto resultante.

Mediante el modelo CWSLR establecimos una guía que influencia a otros niveles del ambiente, principalmente: la estructura de productos (PS: Product Set), la organización de los procesos (MDP: Model Driven Process) y la interfase con el usuario (ej. del SiSoProS: Sistema de Soporte al Proceso de Software en equipo).

Utilizando MDA aportamos a la arquitectura de software un estilo que concuerda con elementos aceptados por OMG: Object Management Group, como son: CORBA[33] [OMG02A] y XMI[34]. Esta infraestructura permitirá sustituir procesos mediante transformaciones de modelo a modelo.

Usar como referencias a CWSLR y a MDA proporcionamos al desarrollador la certeza de que la mayoría de los productos-del-trabajo requeridos ya existen en el PS y que están ordenados de acuerdo a la práctica de un grupo grande de desarrolladores. Aportando una respuesta para satisfacer el requisito:

> *Modelos de Referencia***:** Durante la construcción de un producto de software se requieren modelos de referencia para identificar los productos-del-trabajo a desarrollar.

Utilizamos en PS, como núcleo base, los modelos del lenguaje UML: Unified Modeling Language, para usar en lo posible elementos estandarizados. El orden del PS aporta guías para identificar cuando y como utilizar cada modelo mediante las tablas de ***PS2Set***s y **PS3Set**s del capítulo tres.

Identificamos los modelos UML utilizados en los conjuntos CIM, PIM, PSM, IM y OM, con la intención de ayudar a elegir el nivel de abstracción requerido en cada caso. Con PS cerramos el abanico de posibilidades para el conjunto de modelos, permitiendo su utilización por desarrolladores con poca experiencia, aumentando sus posibilidades de éxito.

Establecimos como uno de los fundamentos a la MDA, para definir estilos arquitectónicos adicionales (anexo B, § B.8). De manera que la arquitectura resultante tenga características conocidas que permitan obtener los atributos de calidad requeridos para el componente.

Incluimos en PS elementos para representar la operatividad de aplicaciones Web. Para diseñar la interfase hombre máquina en Web, utilizamos la técnica OOWS: Object Oriented approach for Web Solution modeling [AFGP02, PAF01], que considera vistas navegacionales (concernientes a la interfaz humano máquina HMI) de las clases definidas para el sistema. Esto complementa la especificación requerida:

> *Especificación***:** En el desarrollo de cada producto-del-trabajo se requiere especificar el elemento del lenguaje UML a utilizar.

Mantuvimos simple al Conjunto de Productos / *Product Set* (PS), proponiendo cadenas de modelos e indicando la forma de obtener cada uno a partir de la información y características

---

[33] CORBA – (Common Objet Request Broker Architecture), arquitectura e infraestructura, independiente-de-proveedor, que las aplicaciones computacionales usan, para trabajar juntas sobre las redes.

[34] XMI – (XML Metadata Intergchange), modelo de intercambio de información abierta, que proporciona a los desarrolladores que trabajan con tecnología de objetos, la habilidad de intercambiar datos de programación sobre el Internet de manera normalizada.

de los modelos desarrollados anteriormente. El orden de los modelos para los conjuntos CIM, PIM, PSM, IM y OM del PS, lo mostramos en el capítulo tres.

Incluimos las entradas necesarias (elementos del modelo previo) para el desarrollo de un modelo objetivo y la manera en que la información de entrada se transforma para obtener los componentes requeridos, al inicio de la página que contiene cada modelo en el caso de estudio usado para el prototipo del SiSoProS. Esto nos permite plantear, para trabajos futuros, el soporte automatizado para desarrollar cada modelo, satisfaciendo la siguiente necesidad:

> *Transformación entre modelos*: Identificar las transformaciones entre modelos con la finalidad de habilitar su realización automática.

Complementamos UML, para modelar conceptualmente el negocio (en forma muy abstracta, incluyendo a los elementos independientes de cómputo), pues además de sus modelos utilizamos técnicas visuales estructuradas, modificadas a Orientación a Objetos (OO), en los subconjuntos de productos del PS1Set "**CIM**". Los principales complementos son los diagramas de colaboración para entidades y los diagramas de línea de producción (§ 3.4). Estas técnicas permiten representar los requerimientos iniciales del sistema a partir de los componentes elementales (salidas, entradas y opciones) y disminuir la necesidad de contar con expertos en el dominio de la aplicación.

Para facilitar el manejo de las tareas de diseño general incluimos diagramas de entidad relación para especificar las relaciones entre los datos (estados) relacionados con los objetos involucrados. Usándose la nomenclatura del ELKA: Entities, Links, Keys and Atributes [R81, GB86]. Los modelos mencionados aportan representaciones que satisfacen el siguiente requerimiento:

> *Modelado conceptual del negocio***:** Para producir un modelo de proceso del negocio es necesario tener elementos explícitos en UML, que sean parte del modelo conceptual del componente en desarrollo.

Utilizamos el proceso MDP (con sus siete sub-procesos) para construir los productos-del-trabajo contenidos en PS. La estructura de PS facilita organizar el desarrollo dirigiéndolo a subproductos que evolucionan de manera asíncrona e independiente (ej. componentes, módulos …).

7.1.2 Proceso Dirigido por Modelos

Organizamos MDP (§ 4.1) integrando cuatro modos de trabajo en grupo, que habitualmente se consideran aisladamente. Esta composición aporta sinergia, que mejora el producto y el proceso de desarrollo. Los modos son: Equipo; Colaborativo; Cooperativo y Control.

Elaboramos, usando MDP, cualquier producto o subconjunto de productos mediante los sub-procesos del modo en "Equipo", realizados sistemáticamente. Asignando a sus participantes roles con responsabilidad relacionada a la administración del trabajo en equipo.

Se requiere tomar en consideración continuamente las interrelaciones entre los cuatro modos, pues cada desarrollador tiene roles diferentes para cada uno de los modos de trabajo y los

realiza de acuerdo a los eventos que suceden. Por ejemplo, cuando se asigna algún producto-del-trabajo, se tiene que decidir si se realizará por una o más personas (en modo colaborativo).

La dinámica que seguimos la dirige el modo de trabajo en equipo, aunque se sabe que tenemos que revisar los productos (modo cooperativo), así como monitorear y corregir el desempeño (en modo control). Ayudando a satisfacer la siguiente necesidad:

> *Desarrollo en equipo***:** El desarrollo de software requiere trabajo en equipo disciplinado y coordinado, asignando responsabilidades y tareas a los participantes.

Otorgamos importancia al análisis del producto deseado y su descomposición en subproductos, en el Proceso Dirigido por Modelos (MDP). Damos mayor importancia al Conjunto de Productos (***PS***) ubicándolo en el nivel principal y asociando una instancia del proceso MDP, a cada producto-del-trabajo necesario para elaborar los componentes de su arquitectura.

Definimos, para el trabajo en equipo, un sólo nivel de sub-procesos y este nivel lo asociamos a un conjunto de productos (uno o varios productos-del-trabajo); aportando una respuesta al siguiente requerimiento:

> *Simplificación***:** La complejidad del proceso se reduce evitando denominaciones semejantes a niveles diferentes (Ej. nombres de fases Vs. disciplinas), mejorando la interpretación y realización incremental del proceso.

Determinamos que la construcción de cualquier modelo o producto-del-trabajo se realiza mediante ciclos de siete sub-procesos, determinando en forma flexibles su orden de ejecución. Con la única restricción de realizar siempre los sub-procesos: primero (lanzamiento &estrategia) y último (pos-término). Permitiendo que los cinco sub-procesos restantes se realicen en cualquier orden o incluso omitirlos si así se justifica.

Permitimos adecuar el proceso, según: a) el contexto en que se usará el software, b) la forma en que se desarrollará y c) la experiencia de desarrollos previos, dando facilidades para elegir: 1) la estructura jerárquica (niveles de ensamble-parte) del producto, 2) el conjunto específico de productos-del-trabajo a desarrollar para cada parte, 3) la composición del equipo de desarrollo, 4) los roles asignados a cada participante, 5) desarrollar colaboratívamente, 6) la forma de realizar el seguimiento y control y 7) la forma de realizar inspecciones. Aportando una alternativa al siguiente requisito:

> *Flexibilidad***:** Los procesos no deben ser secuenciales ni rígidos, de tal manera que el proceso pueda modificarse, incluso aún cuando se realicen transformaciones automáticas entre modelos.

Combinamos en AGD los conceptos y contribuciones de los trabajos relacionados, descritos en las áreas de desarrollo de software ágil, desarrollo iterativo e incremental y mejora del proceso de software. AGD proporciona la mayoría de las características deseables para un método de desarrollo de software:

- Organizando el trabajo en equipo mediante ciclos que permiten frecuentes entregas de incrementos del software.
- Promoviendo la interacción estrecha entre el cliente y el desarrollador, desde el modelado del negocio, pues los diagramas que propone reflejan las prácticas utilizadas en la operación diaria.

- Procurando que el producto de software tenga documentación suficiente, mediante una estructura de fólderes de unidad (de trabajo), para facilitar la operación y modificación.
- Tendiendo a usar transformación entre modelos, para atender cambios y posibilitar la reacción adecuada ante modificaciones de último momento.

Satisfaciendo la siguiente necesidad:

*Agilidad***:** El proceso debe permitir la adecuación dinámica y realización en forma ágil: incremental, cooperativo, directo y adaptable, es decir adecuar productos, sub.-procesos y formas.

7.2Trabajos Futuros

Mencionamos, en esta sección, algunos de los trabajos para continuar con nuestra línea de investigación. Presentamos primero los desarrollos del área del producto de software y posteriormente los proyectos en el área del proceso de software.

7.2.1 Conjunto de Productos

Existen diversas herramientas que realizan transformaciones entre modelos, pero ninguna de ellas toma en cuenta los mismos conceptos o características. Para pasar de un modelo a otro, se necesita tanto de lenguajes para describir transformaciones entre modelos, como de una forma de representación permanente de los modelos, que permitan mantener: consistencia, el orden adecuado, y el grado de relación establecido entre ellos.

Requerimos realizar una clasificación de los elementos que intervienen, para unificar la representación de las transformaciones y establecerá detalladamente los límites entre cada tipo de modelo (CIM, PIM, PSM, IM, OM), evitando los traslapes entre estos. Asegurando que sean seguibles los elementos considerados, del modelo objetivo al modelo fuente, cuidando la bidireccionalidad de las transformaciones y manteniendo la consistencia que permita que información agregada al modelo objetivo persista, para regenerarlo.

En el desarrollo de software dirigido por modelos las transformaciones entre modelos se consideran activos vitales que deben ser manejados con principios de ingeniería de software. Las transformaciones deben ser analizadas, diseñadas, implementadas, probadas, mantenidas y estar sujetas a la administración de configuración. Necesitamos identificar técnicas y métodos que permitan el desarrollo de transformaciones y su mantenimiento.

Requerimos ayudar a los usuarios del método AGD, para que a partir de un modelo, propongamos transformaciones posibles y realicemos algunas en forma semiautomática. Asegurando la posibilidad de automatizar las transformaciones. De tal manera que los desarrolladores empleen menor esfuerzo, tengan mejor consistencia en el proceso de desarrollo de software y produzcan software con menor cantidad de defectos.

Tomando en consideración el estado descrito, promovemos el desarrollo de un proyecto que tenga como objetivo:

Definir un esquema de caracterización de los elementos de transformación de los diagramas del método AGD, conservando los detalles adicionales en representaciones permanentes estándar [CH03] [TEL05].

Realizaremos trabajo experimental para evaluar y complementar el Conjunto de Productos (PS), con el propósito de identificar las diferencias requeridas en varios dominios de aplicación.

7.2.2 Proceso Dirigido por Modelos

Los proyectos de desarrollo de software difieren en las características: alcance, funcionalidad requerida, dominio de aplicación, complejidad, recursos y requerimientos tecnológicos. Para considerar las diferencias mencionadas, en AGD identificamos siete áreas principales de toma de decisiones:

- El conjunto de resultados a obtener (cada uno construido en un paso específico, tipo PSP);
- La estructura general del producto de software (conjunto de niveles ensamble-parte), deberá correlacionarse con los componentes de la arquitectura del software (ubicando los componentes de la arquitectura en los niveles de la estructura a usar);
- El conjunto de sub-procesos a realizar para obtener cada producto (conjunto resultado);
- La estructura del equipo de trabajo (como el número de participantes, roles, asignación de productos a realizar);
- La forma de realizar el desarrollo (colaboratívamente o individualmente), de cada resultado a obtener;
- La forma de realizar el seguimiento y control del proceso (periodicidad de reuniones, protocolo de reuniones, resultados a obtener);
- La forma de realizar inspecciones (revisiones técnicas formales) y su relación con el producto y los eventos del proceso.

Es posible que no consideren las áreas anteriores, los desarrolladores de software y apliquen un mismo proceso sin considerar las características particulares del proyecto.

La omisión de alguna áreas puede provocar que el proceso no sea el más adecuado (ya sea al inicio del proyecto o cuando se presente algún cambio importante en las características del proyecto), redundando en retrasos en la entrega, factores inapropiados de calidad y en costos de desarrollo más altos.

Pretendemos contar con un modelo de clasificación de proyectos de software que posibilite empatar las características del proyecto de software a desarrollar con un patrón de proyecto, permitiendo a los líderes del proyecto tomar la decisión del proceso a seguir para desarrollar una aplicación.

Tomando en consideración el estado descrito, promovemos el desarrollo de un proyecto que tenga como objetivo:

Construir patrones que identifiquen a proyectos de software, para determinar el proceso adecuado a seguir**,** utilizando las opciones disponibles del método AGD [BT03] [K04-A].

Requerimos realizar trabajo experimental para evaluará y complementará al Proceso Dirigido por Modelos (MDP), identificando las diferencias requeridas en varios dominios de aplicación y diversas situaciones.

8 Anexo A.- Patrones de Modos de Trabajo en Grupo

Los cuatro modos de trabajo en grupo se describen utilizando el patrón de procesos de acuerdo con la norma MOPROSOFT [O03]. El patrón para el modo de proceso en Equipo, se incluye en el capítulo cinco § 5.4.

Se incluyen las secciones siguientes:

- Definición del Patrón de Proceso (MOPROSOFT)
- Proceso: (MP-CL) Modo de Proceso Colaborativo
- Proceso: (MP-CP) Modo de Proceso Cooperativo
- Proceso: (MP-CT) Modo de Proceso de Control

8.1Definición del Patrón de Proceso (MOPROSOFT)

Proceso: Nombre de proceso

Definición general de proceso

PROCESO	Nombre del proceso precedido por el acrónimo establecido en la definición del modelo de procesos.
Categoría/Modo	Nombre de la categoría/modo al que pertenece el proceso y el acrónimo entre paréntesis.
Propósito	Objetivos medibles y resultados esperados de la implementación efectiva del proceso.
Descripción	Descripción de las actividades y productos que componen el flujo de trabajo del proceso.
Objetivos	Objetivos específicos cuya finalidad es asegurar el cumplimiento del propósito del proceso. Identificados como O1, O2, etc.
Indicadores	Definición de los indicadores de evaluación del cumplimiento de los objetivos del proceso. Los indicadores se identifican por I1, I23, etc. y entre paréntesis se especifica una o más objetivos a los que dan respuesta.
Metas cuantitativas	Valor numérico o rango de satisfacción por indicador.
Responsabilidad y autoridad	El rol principal responsable de la ejecución del proceso. Autoridad es el rol que valida la ejecución del proceso y el cumplimiento del propósito.
Subprocesos (opcional)	Lista de procesos de los cuales se compone el proceso en cuestión.
Procesos relacionados	Nombres de las áreas de proceso relacionadas.

Entradas

Nombre	Fuente
Nombre del producto o recurso	Referencia al origen del producto o recurso

Salidas

Nombre	Descripción	Destino
Nombre del producto o recurso	Descripción y características del producto o recurso	Referencia al destinatario del producto o recurso

Productos internos

Nombre	Descripción
Nombre del producto generado y utilizado en el propio proceso	Descripción y características del producto

Referencias bibliográficas

Bibliografía que Sustenta el Proceso: Normas Modelos de Referencia, Libros y Otras Fuentes

Prácticas
Identificación de roles involucrados y capacitación requerida

Rol	Abreviatura	Capacitación
Nombre del rol	Abreviatura del rol	Capacitación requerida por el rol para poder ejecutar el proceso

Actividades
Se asocian a los objetivos y describen las tareas y roles responsables

Rol	Descripción
Abrev. del(os)	A1. Nombre de la actividad (O1, O2, …)

rol(es)

A1.1 Descripción de la tarea 1. Si la actividad es una verificación o validación se hará referencia a la identificación de la misma.

A1.2 Descripción de tarea 2

A2. Nombre de la actividad (O1, O2, …)

A2.1 Descripción de la tarea 1.

A2.2 Descripción de la tarea 2.

Diagrama de flujo de trabajo

Diagrama de colaboración de UML, donde se especifican los actividades del flujo de trabajo y los productos.

Tabla de Petri indicando eventos importantes

Siguiendo las sugerencias de Kenneth J. Hints y Daniel Tabak [HT92], acerca de implementación de aplicaciones con microcontroladores.

Para interpretar las tablas de Petri se considera:

NOMBRE: Una descripción de estado del proceso (sus marcas).

LUGAR: Representan condiciones, a las que se asignan identificadores cortos. En nuestro caso será Inicio (I), Terminación (F) o un producto de entrada (P1, P2, …). Asumiendo que el lugar puede albergar al desarrollador de la transición asociada (Ej. DES: Desarrollador).

MARCA: Puede ser una o varias, se pasan de lugares a transiciones y de transiciones a lugares. Son interrupciones o eventos (ej. terminación de producto, terminación de inspección, recepción de lista de defectos, la disponibilidad del desarrollador requerido, etc.), que ocurren asincrónicamente con el proceso de desarrollo.

Se asume una marca (Ej. MGG: Moisés González García) que indica que existe el recurso (el desarrollador) y que puede realizarse la transición (la práctica específica).

MARCAS POSIBLES: Marcas que pueden existir en este lugar.

MARCAS HABILITAN: Marcas que habilitan a la transición (prácticas específicas) listada en la columna de la derecha.

TRANSICION*:* Representa el cálculo a efectuar y/o una acción a realizar. En nuestro caso una práctica específica (PE1, PE2, …).

MARCAS DE SALIDA: Las marcas que la transición de la columna a la izquierda pasa al siguiente lugar. En nuestro caso al terminarse una práctica y obtener el resultado requerido, se libera a su desarrollador (Ej. MGG).

ACCION: Representa sub-acciones (realizadas durante la transición), específicas para la situación a representar.
LUGAR SIGUIENTE: El lugar siguiente (que contendrá al producto a obtener) al ejecutar la transición asociada. Este lugar contendrá también al desarrollador que realizó la transición asociada (pues queda libre para hacer otra transición.
ΔT: Restricción en el tiempo

Tabla de Petri para modo de proceso MP-TE

NOMBRE	LUGAR	MARCA	MARCA HABILITA	TRANSISION	MARCA SALIDA	ACCIONES	LUGAR SIGUIENTE	ΔT

Tabla de Marcas

Tipo	Etiqueta	Significado

Verificaciones y Validaciones

Se definen las verificaciones y validaciones asociadas a los productos generados en las actividades que se mencionan.

En la verificación como en la validación se identifican los defectos que deben corregirse antes de continuar las actividades posteriores.

La validación de un producto puede ser interna (dentro de la organización) o externa (por el cliente) con la finalidad de obtener su autorización.

Se recomienda que las validaciones se efectúen una vez que las verificaciones asociadas al producto sean realizadas.

Verificación o validación	Actividad	Producto	Rol	Descripción
Identificación de la verificación o validación Ver1 o Val1	Identificación de la tarea	Nombre del producto	Abreviatura del rol responsable de realizar la verificación o validación	Descripción de la verificación o validación que se hará al producto

Incorporación a la base de conocimiento

Producto	Forma de aprobación
Nombre de producto	Identificación de la verificación, validación o descripción de otra forma de aprobación, en caso de no requerirse alguna de éstas escribir la palabra Ninguna. Estas aprobaciones definen el momento a partir del cual el producto estará bajo control delConocimiento de la Organización

Recursos de Infraestructura

Actividad	Recurso
Identificación	Requerimientos de herramientas de software y hardware de la actividad o tarea

Mediciones

Mediciones que se establecen para evaluar los indicadores del proceso. Las mediciones se identifican como M1, M2, etc. y entre paréntesis se especifica la identificación del indicador que le corresponde

Capacitación

Definición de las reglas para proporcionar la capacitación necesaria a los roles involucrados en el proceso.

Situaciones excepcionales

Definición de los mecanismos para el manejo de las situaciones excepcionales durante la ejecución del proceso.

Lecciones aprendidas

Definición de los mecanismos para aprovechar las lecciones aprendidas durante la ejecución del proceso.

Guías de ajuste

Descripción de posibles modificaciones al proceso que no deben afectar sus objetivos.

Identificación de la guía Descripción

En las secciones siguientes presentamos el patrón de proceso para cada uno de los cuatro modos de trabajo en grupo.

8.2Proceso: (MP-CL) Modo de Proceso Colaborativo

Definición general del proceso

Proceso	MDP-CL: Modo de Proceso Colaborativo.
Categoría/Modo	Modos de trabajo en grupo.
Propósito	Coordinar el trabajo Colaborativo: con el que se construyen los resultados requeridos. Los actores que intervienen en cualquier modo de trabajo realizan en modo Colaborativo los resultados requeridos para cumplir con los compromisos de los *role*s que interpretan. El trabajo Colaborativo se asigna por pareja, preferentemente, para

desarrollar partes específicas de los modelos o productos-del-trabajo. Los miembros del grupo que trabaja en forma colaborativa son como un organismo inteligente coherente trabajando con una mente y responsable de todo el producto.

Descripción

Este modo de trabajo supone: 1) una actividad colocalizada síncrona (todos los participantes en: el mismo lugar, y el mismo tiempo); 2) dicha actividad realizada por un actor que toma el *rol* de **driver** (conductor) y uno o más actores que toman los *role*s de **co-driver** (co-conductor) y **assistant** (ayudante).

El **driver** usa los dispositivos de entrada para construir el modelo o producto.

El modelo o producto-del-trabajo, se va produciendo colaborativamente por el **driver**, los **co-drivers**, y los **assistants** dentro de los límites acordados de productividad.

El **co-driver** toma forzosamente el papel de **driver** alternándose. Compartiendo el **driver** y el **co-driver** la responsabilidad y el compromiso acerca de la tarea que realizan.

Objetivos

O1 - Realizar los resultados requeridos por la dinámica de los sub-procesos del modo de trabajo en equipo.

O2 - Desarrollar los resultados utilizados en los modos Cooperativo y de Control.

O3 - Tomar las decisiones acerca de la forma para construir el modelo conforme se avanza (en la marcha), sin obstaculizar el avance, cumpliendo con las metas de calidad preestablecidas.

Indicadores

Para las áreas de proceso de la categoría de Ingeniería, se consideran los indicadores siguientes:

Contenido de MDP-SUMQ:

I1 Percent Defect Free

I2 Defect/page

I3 Defects/KLOC

I4 Defect ratios

I5 Development time ratios (%)

I6 A/FR

I7 Personal review rates

I8 Inspection rates

I9 Tasas de inserción de defectos (Defectos insertados por hora)

I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)

I11 Yields de las fases (Defectos eliminados en la fase / Defectos eliminados en la fase + ESCAPES)

I12 Yield del proceso

Para las demás áreas se establecen ad hoc, documentándose los

indicadores utilizados.

Metas cuantitativas

Para las áreas de proceso de la categoría de Ingeniería, se consideran los indicadores siguientes:

I1 Percent Defect Free (PDF)

Compilación	> 10%
Prueba de unidad	> 50%
Prueba de integración	> 70%
Prueba de sistema (aceptación)	>90%

I2 Defect/page

I3 Defects/KLOC

Total de defectos inyectados	76 – 150
Compilación	< 10
Prueba de unidad	< 5
Integración	< 0.5
Prueba de s	< 0.2

I4 Defect ratios

Defectos revisión DLD/Defecto en prueba de unidad > 2.0

Defectos revisión de código/Defectos Compilación > 2.0

I5 Development time ratios (%)

Inspección de requerimientos/Tiempo requerimientos	> 0.25
Inspección HLD/Tiempo HLD	> 0.5
DLD/Tempo codificación	> 1.0
Revisión DLD/Tiempo DLD	> 0.5
Revisión codificación/Tiempo codificación	> 0.5

I6 A/FR

I7 Personal review rates

I8 Inspection rates

Páginas de requerimientos/Hora	< 2
Páginas HLD/Hora	< 5
Líneastexto DLD/Hora	< 100
LOCs/Hora	< 200

I9 Tasas de inserción de defectos (Defectos insertados por hora)

Defectos requerimientos/Hora	0.25

	Defectos HLD/Hora 0.25
	Defectos DLD/Hora 2.0
	Defectos código/Hora 4.0
	Defectos compilación 0.3
	Defectos prueba unidad 0.2
	I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)
	Defectos inspección requerimientos/Hora 0.5
	Defectos inspección HLD/Hora 0.5
	Defectos inspección DLD/Hora 2.0
	Defectos revisión código/Hora 6.0
	Defectos inspección código/Hora 1.0
	I11 Yields de los sub-procesos
	Inspección de requerimientos - 70%
	Revisiones e inspecciones de diseño - 70%
	Revisiones e inspecciones de código - 70%
	Compilación - 50%
	Prueba sistema con 5 o menos defectos/K LOCs - 90%
	Prueba sistema a < 1.0 defectos/KLOC - 80%
	I12 Yield del proceso
	Antes compilación >75%
	Antes prueba unidad > 85%
	Antes de integración > 97.5%
	Antes de prueba sistema > 99%
Responsabilidad y autoridad	*Driver y co-driver* comparten la responsabilidad acerca del resultado obtenido. Los *co-drivers* toman el papel de *driver* alternativamente, procurando que cada uno de los actores ocupe forzosamente el rol de *driver* por un periodo igual. El rol responsable por validar la ejecución del proceso y el cumplimiento de su propósito, depende del producto que se elabore colaborativamente. La autoridad depende directamente del modo de trabajo en que se haya requerido el producto-del-trabajo: Para modo de trabajo En Equipo.- el Administrador de Desarrollo. Para modo de trabajo Cooperativo.- el Facilitador. Para modo de trabajo de Control.- el Líder de la reunión.
Subprocesos (opcional)	*Preparación*.- Definir y conseguir los recursos necesarios. *Uso de recursos y herramientas*.- Uso alternada de recursos

	disponibles por el driver y co-driver. Terminación.- sumarizar y registrar recursos utilizados, entregar producto terminado.
Procesos relacionados	Modo de trabajo En Equipo Modo de trabajo Cooperativo Modo de trabajo de Control

Entradas

Nombre	Descripción	Fuente
Entrada-1 del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate
Entrada-2 del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate
Entrada-n del trabajo específico	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate

Salidas

Nombre	Descripción	Destino
Producto-1 del trabajo espec	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate
Producto-2 del trabajo espec	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate
Producto-n del trabajo especí	Requerido en la categoría del PS que se atiende por el desarrollador	Depende de la categoría del PS de que se trate

Productos internos

Nombre	Descripción
MDP-LOGT Time Recording Log	Para registrar el tiempo ocupado en cada tarea del proyecto

MDP-LOG Defect Recording Log	Para mantener datos acerca de los defectos encontrados y corregidos
MDP-SUMP Plan Summary	Mantiene datos planeados y reales para ensambles / productos

Referencias bibliográficas

GM04 **Moisés González-García, Ana María Martínez Enríquez; The Architectural And Group Development Method: An Experimentation; Workshop On Quantitative Techniques For Software Agile Process (QUTE-SWAP), Acm Sigsoft 2004 / Fse-12 (Foundations On Software Engineering); Oct. 31 – Nov. 6, 2004.**

DW02 **P. David Stotts, Laurie A. Williams: Distributed Pair Programming. XP/Agile Universe 2002: 283**

W00 Laurie Ann Williams, The Collaborative Software Process, PhD dissertation, Department of Computer Science, The University of Utah, August 2000

Prácticas

Identificación de roles involucrados y capacitación requerida

Rol	Abreviatura	Capacitación
Driver (Piloto)	CL-DR	Usa los dispositivos de entrada para construir el modelo, tomando las decisiones acerca de las sugerencias hechas por los **co-driver***s* acerca de la manera más apropiada para realizar el trabajo. Comparte la responsabilidad acerca del resultado obtenido con los **co-drivers**.
Co-driver (Co-piloto)	CL-CDR	Observa las acciones del **driver** y comunica su acuerdo o desacuerdo sugiriendo alternativas de solución y compartiendo la responsabilidad, sobre los resultados, con el **driver**. Los **co-driver**s toman el papel de **driver** alternativamente, procurando que cada uno de los actores ocupe forzosamente el rol de **driver** por un periodo igual.
Assistant (Ayudante)	CL-AS	Colaboran proporcionando comentarios, datos o insumos necesarios para la actividad, sin poder ser **Driver.** Los assistants no tienen responsabilidad directa de la tarea realizada.

Actividades

Se asocian a los objetivos y describen las tareas y roles responsables

Rol	Descripción
Abrev. del(os) rol(es)	A1 Preparación
CL-DR CL-CDR	A1.1 Planeación del trabajo
CL-DR CL-CDR	A1.2 Asignación de recursos
	A2. Uso de recursos y herramientas
CL-DR	A2.1 Desarrollo con herramientas
CL-CDR	A2.2 Observar y proponer alternativas
	A3. Terminación
CL-DR CL-CDR	A3.1 Llenado de formas
CL-DR CL-CDR	A3.2 Entrega de producto

Diagrama de flujo de trabajo

En la figura 8-1 observamos que la pareja o grupo que trabaja en modo colaborativo, para construir un modelo o producto cualquiera, distribuye a sus actores en los roles mostrados en la parte inferior del diagrama de estados. Recuérdese que el trabajo individual se trata como un caso especial del modo colaborativo en el que no se cuenta con el *rol* **Co-driver**.

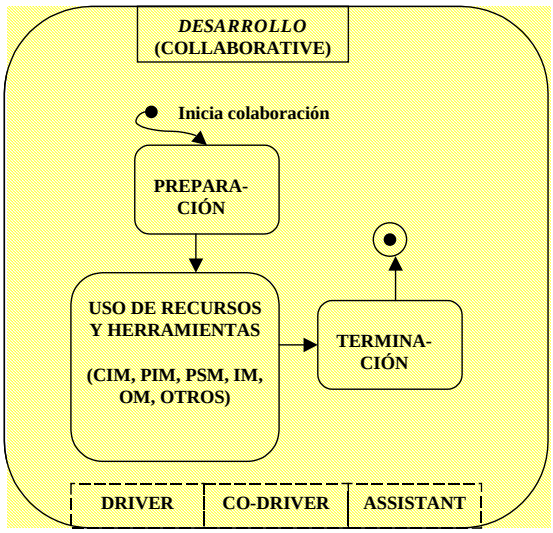


Figura 8-1 Sub-procesos y Roles del Trabajo Colaborativo (Collaborative).

El grupo que colabora, para obtener el modelo requerido, organiza el trabajo ajustándose a los sub-procesos (del modo de trabajo En Equipo) iterando los sub-procesos si es necesario. Cuando se termina la labor, en el sub-estado Terminación, se pide que se programe una Revisión Técnica (desarrollada en modo Cooperativo).

El grupo productor (que trabajó en modo colaborativo) seguramente ya tiene otra asignación (modelo o producto), así que se realiza el modo Cooperativo (Revisión Técnica) de un modelo concurrentemente con el modo Colaborativo (Desarrollo) de otros modelos, asignados ya sea al grupo productor del modelo que se revisa o a otros grupos.

La figura 8-2 muestra una representación estática del modo Colaborativo de trabajo en grupo, utilizado para elaborar cualquiera de los modelos sugeridos por el método **AGD**. En el modo Colaborativo se coordina el trabajo con el que se construyen los resultados requeridos por el cualquiera de los otros tres modos de trabajo en grupo. El contexto en que se realiza este modo de trabajo lo establece el modo de trabajo En Equipo (§ 5.4), que coordina los sub-procesos y los roles necesarios para producir cualquier modelo o conjunto de modelos. Favor de ver § 5.8 para el detalle de las etiquetas asociadas a las relaciones (flechas).

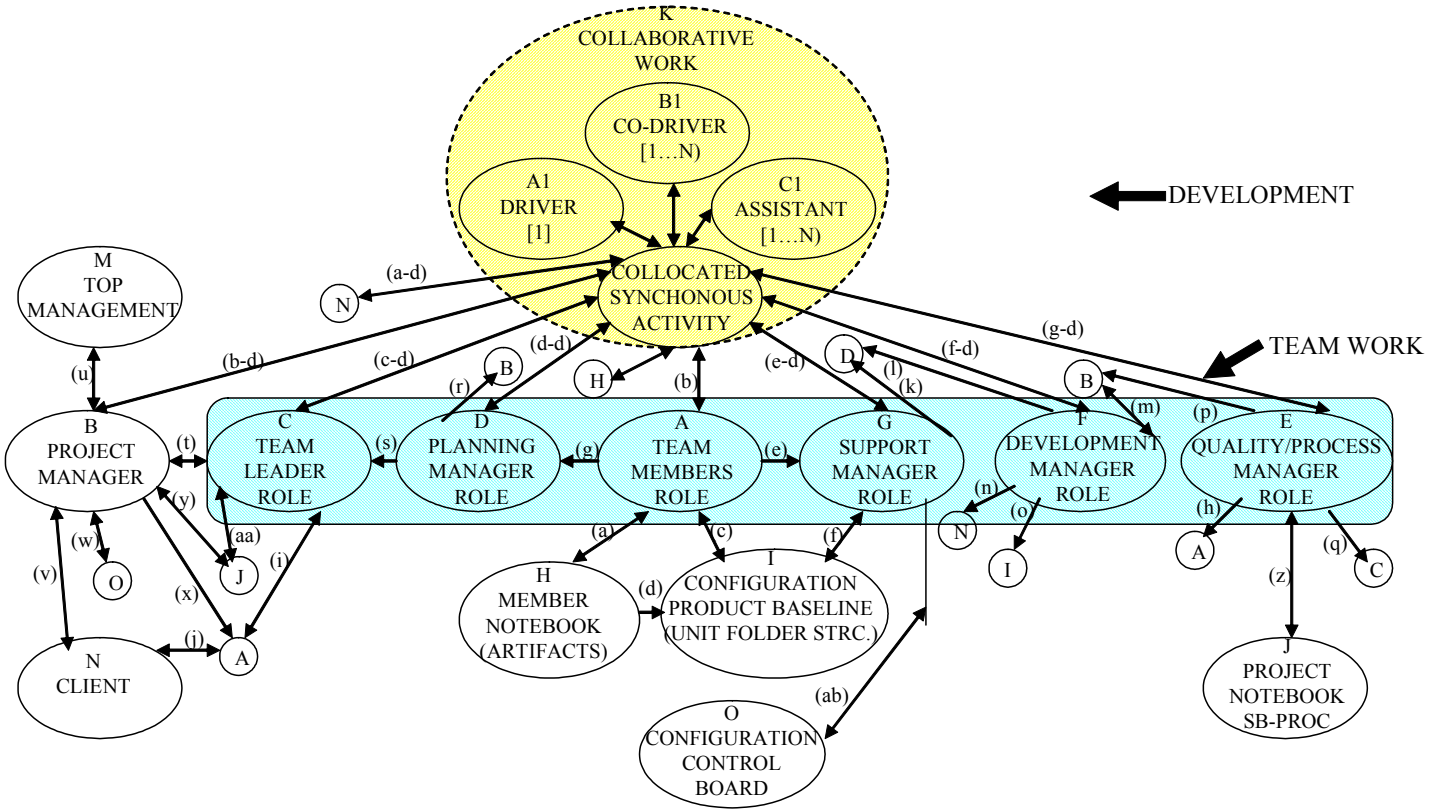


Figura 8-2 Representación estática de Roles del Trabajo Colaborativo (Collaborative) y su relación con el modo de trabajo En Equipo.

Verificaciones y Validaciones

Verificación o validación	Actividad	Producto	Rol	Descripción
Revisión técnica	Solicitada en el sub-estado Terminación	Producto-del-trabajo (terminado)	(CL-DR = CL-CDR = CP-PR = TE-TM) TE-QPM CP-SL	Revisión técnica

8.3Proceso: (MP-CP) Modo de Proceso Cooperativo

Definición general del proceso

Proceso	MDP-CP: Modo de Proceso Cooperativo.
Categoría/Modo	Modos de trabajo en grupo.
Propósito	Para realizar revisiones técnicas formales, documentando y resolviendo conflictos entre las opiniones de los participantes. Se puede utilizar para revisar formalmente cualquier modelo , resultado o artefacto producido en el proceso de desarrollo de software. El término formal implica que se documentan y difunden las decisiones técnicas tomadas, así como los comentarios o sugerencias de los participantes.
Descripción	Este modo de trabajo supone: 1) una actividad colocalizada síncrona (todos los participantes en: el mismo lugar, y el mismo tiempo); 2) dicha actividad realizada por un equipo de revisión cuyos miembros pueden tomar alguno de cinco roles específicos: 1) **facilitator** (facilitador); 2) **writer** (apuntador); 3) **reviewer** (revisor); 4) **producer** (productor) y 5) **observer** (observador). En este modo de trabajo se revisa un producto, se hacen comentarios técnicos pertinentes, y se acuerda una decisión relativa a la aceptación o rechazo del producto de acuerdo a las normas establecidas para el proyecto. Reglas básicas para el modo **cooperative work** (preparación): El **producer** (productor), o grupo productor, del artefacto a revisar pide la sesión de revisión cuando da por terminado su producto.

Solicita la revisión al **quality/process manager** (administrador dr calidad/proceso, *rol* del modo En Equipo).

El **quality/process manager** funge como **facilitator** o designa al **facilitator**, asigna una fecha y designa los miembros del equipo del proyecto que tomarán los *roles* necesarios para ejecutar la revisión.

Cada uno de los cinco *roles* tiene una descripción que detalla sus metas y sus actividades principales; así como una tabla que enumera las actividades que dirige y en las actividades en que participa.

Las sesiones del trabajo cooperativo deben tener la prioridad más alta en los planes de trabajo. Deben empezar a tiempo contando con la puntual asistencia de los participantes.

Requiriéndose un grupo mínimo, además del productor, de tres miembros y un máximo recomendado de siete miembros. Sin contar a los **observers**.

Durante las sesiones se evitan interrupciones de cualquier tipo: para dar continuidad al trabajo.

El **producer** debe estar presente sin excepción.

El **producer** proporciona a cada uno de los participantes designados un ejemplar del artefacto a revisar con una anticipación suficiente para su revisión, dependiente de las características del resultado a revisar. La anticipación mínima es de dos días antes a la fecha acordada para la reunión.

Los **reviewers** analizan el producto y preparan sus comentarios antes de la reunión.

Reglas básicas para el modo **cooperative work** (desarrollo):

El **facilitator** propone al equipo de revisión la forma de realizar la revisión, para hacerla en el tiempo disponible. Acuerda con el equipo la granularidad de la revisión (líneas, párrafos, secciones, figuras, etc.).

El **writer** escribe los comentarios que surjan durante la sesión sobre un medio visible síncronamente (en el momento de ocurrir el comentario o sugerencia) por todos los miembros del equipo de revisión. Además realiza los reportes de la sesión.

La responsabilidad del grupo que hace la revisión, en trabajo cooperativo, es obtener con calidad (a tiempo y con información fidedigna) los reportes del trabajo de la revisión con los comentarios y la decisión tomada.

La revisión debe concentrarse en aspectos técnicos no en la forma. Estableciendo “que” se puede mejorar del producto sin indicar “como” hacerlo.

Los participantes deben tomar una actitud critica constructiva; no

destructiva. Haciendo comentarios únicamente acerca del producto.

Los **producers** y **reviewers** deben estar abiertos a los comentarios técnicos, para mejorar la calidad del producto.

Reglas básicas para el modo **cooperative work** (toma de decisiones):

Las decisiones posibles son:

Se acepta como está.- se acuerda en forma unánime, quedando a criterio del **producer** si modifica el producto de acuerdo a los comentarios menores resultantes.

Se acepta con comentarios menores,- cuando menos un **reviewer** considera que los comentarios requieren de seguimiento (un miembro del grupo que revisó se asigna para asegurar que se incorporaron los cambios necesarios al producto para atender a los comentarios resultantes) para asegurar que se incorporen modificaciones que mejoren al producto.

No se acepta con comentarios mayores (forzosa otra revisión).- cuando menos un **reviewer** considera que debe revisarse el producto después de que el **producer** termine las modificaciones que atiendan a los comentarios emitidos.

No se acepta se requiere rehacer.- cuando menos un **reviewer** considera que el producto se tiene que rehacer, pues atender a los comentarios modificando al producto existente requiere más recursos (humanos, materiales o tiempo.

Revisión incompleta.- cuando menos un **reviewer** requiere de un recurso que no esta disponible (humano, material o tiempo), para completar las metas de la revisión.

Objetivos

O1 – Obtener resultados de calidad del proceso de revisión.

O2 –Mejora continua del proceso y delos productos.

O2 – Comunicación técnica de temas técnicos.

O3 – Distribución de información y difusión de experiencias.

Indicadores

(Se consideran los mismos que para el modo En Equipo)

Para las áreas de proceso de la categoría de Ingeniería, se consideran los indicadores siguientes:

Contenido de MDP-SUMQ:

I1 Percent Defect Free

I2 Defect/page

I3 Defects/KLOC

I4 Defect ratios

I5 Development time ratios (%)

I6 A/FR

I7 Personal review rates

I8 Inspection rates

I9 Tasas de inserción de defectos (Defectos insertados por hora)

I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)

I11 Yields de las fases (Defectos eliminados en la fase / Defectos eliminados en la fase + ESCAPES)

I12 Yield del proceso

Para las demás áreas se establecen ad hoc, documentándose los indicadores utilizados.

Metas cuantitativas

(Se consideran las mencionadas en el modo En Equipo)

Para las áreas de proceso de la categoría de Ingeniería, se consideran los indicadores siguientes:

I1 Percent Defect Free (PDF)

Compilación	> 10%
Prueba de unidad	> 50%
Prueba de integración	> 70%
Prueba de sistema (aceptación)	>90%

I2 Defect/page

I3 Defects/KLOC

Total de defectos inyectados	76 – 150
Compilación	< 10
Prueba de unidad	< 5
Integración	< 0.5
Prueba de sistema	< 0.2

I4 Defect ratios

Defectos revisión DLD/Defecto en prueba de unidad	> 2.0
Defectos revisión de código/Defectos Compilación	> 2.0

I5 Development time ratios (%)

Inspección de requerimientos/Tiempo requerimientos	> 0.25
Inspección HLD/Tiempo HLD	> 0.5
DLD/Tempo codificación	> 1.0
Revisión DLD/Tiempo DLD	> 0.5
Revisión codificación/Tiempo codificación	> 0.5

I6 A/FR

I7 Personal review rates

I8 Inspection rates

Páginas de requerimientos/Hora	< 2
Páginas HLD/Hora	< 5
Líneastexto DLD/Hora	< 100
LOCs/Hora	< 200

I9 Tasas de inserción de defectos (Defectos insertados por hora)

Defectos requerimientos/Hora	0.25
Defectos HLD/Hora	0.25
Defectos DLD/Hora	2.0
Defectos código/Hora	4.0
Defectos compilación	0.3
Defectos prueba unidad	0.2

I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)

Defectos inspección requerimientos/Hora	0.5
Defectos inspección HLD/Hora	0.5
Defectos inspección DLD/Hora	2.0
Defectos revisión código/Hora	6.0
Defectos inspección código/Hora	1.0

I11 Yields de los sub-procesos

Inspección de requerimientos	- 70%
Revisiones e inspecciones de diseño	- 70%
Revisiones e inspecciones de código	- 70%
Compilación	- 50%
Prueba sistema con 5 o menos defectos/K LOCs	- 90%
Prueba sistema a < 1.0 defectos/KLOC	- 80%

I12 Yield del proceso

Antes compilación	>75%
Antes prueba unidad	> 85%
Antes de integración	> 97.5%
Antes de prueba sistema	> 99%

Responsabilidad y autoridad

La responsabilidad del grupo de revisión técnica en trabajo Cooperativo, es obtener con calidad los reportes de la revisión con los comentarios y la decisión tomada.

La responsabilidad del (los) productor (es), que utilizó el modo Colaborativo, es el producto-del-trabajo (o artefacto)

desarrollado.Para modo de trabajo Cooperativo.- el Facilitador.

Para modo de trabajo de Control.- el Líder de la reunión.

En su momento el *CCB* pasa su recomendación y el administrador del proyecto (**Project Manager)** toman las decisiones relativas al programa de actividades y la utilización de recursos.

Procesos relacionados

Modo de trabajo En Equipo

Modo de trabajo Colaborativo

Modo de trabajo de Control

La relación que existe entre trabajo Colaborativo y trabajo Cooperativo se puede expresar de la siguiente manera: los productos desarrollados con trabajo Colaborativo se revisan formalmente con trabajo Cooperativo.

El productor (**producer),** en base a los comentarios técnicos resultantes de la revisión, llena una forma MDP-CCR: Configuration Change Request (Solicitud de Cambio de Configuración) y la pasa al Comité de Control de Configuración (CCB: Configuration Control Borrad).

Reglas para relacionarse con el modo En Equipo:

El grupo al que se le asigno, en el modo En Equipo, el producto a revisar tiene el rol de **producer** en la sesión cooperativa, sus miembros no pueden asumir ningún otro *rol*.

El **quality/process manager** (rol del modo En Equipo) funge como **facilitator** (rol del modo Cooperativo), o coordina la asignación del *rol* **facilitator**.

Los eventos del modo En Equipo que ocasionan la programación de una sesión de trabajo cooperativo son:

Terminación de los *sub-procesos* siguientes: SP-REQUIREMENTS (Requerimientos, Análisis), SP-ANÁLISIS & DESIGN (Análisis y Síntesis), SP-IMPLEMENTATION (Implementación), SP-TEST (Prueba).

Terminación de la elaboración de normas y procesos a seguir

Entradas

Nombre	*Descripción*	*Fuente*
Producto-del-trabajo (artefacto)	*Producido en la categoría del PS que se atiende por el desarrollador*	*Depende de la categoría del PS de que se trate*

Documentación relacionada al Producto-del-trabajo a revisar	*Variado según se requiera*	*Depende de la categoría del PS de que se trate*

Salidas

Nombre	Descripción	Destino
MDP-INS-DET Inspection Report-Detail (Reporte Detallado de Inspección)	Recolecta los datos detallados acerca de una inspección/revisión	Producer (Productor) Miembros del equipo afectad
MDP-INS-SUM Inspection Report-Su (Resumen de Inspección)	Para reportar el sumario del análisis hecho a los datos de una inspección/revisión	Project manager (Administrador del Proyecto) Client (Cliente) Top management (Administración superior)

Referencias bibliográficas

GM04	**Moisés González-García, Ana María Martínez-Enríquez; The Architectural And Group Development Method: An Experimentation; Workshop On Quantitative Techniques For Software Agile Process (Qute-Swap), Acm Sigsoft 2004 / Fse-12 (Foundations On Software Engineering); Oct. 31 – Nov. 6, 2004.**
A84	**R. Axelrod; The Evolution of Cooperation. Basic Books, 1984**
A97-b	R. Axelrod; The Complexity of Cooperation; Princeton University Press , 1997
IEEE-1028	Std. 1028 - 98; IEEE Standard for software reviews; IEEE Computer Society, Software Engineering Standards Committee, March 1998.

Prácticas

Identificación de roles involucrados y capacitación requerida

Rol	Abrevia-tura	Capacitación
Producer	CP-PR	Responsable del producto; decide cuando está listo para la revisión; responsable de resolver los comentarios de

(productor)		la revisión; demostrar que el producto-del-trabajo está completo.
Sesion lider (líder de sesión, o facilitador))	CP-SL	Responsable de obtener una revisión buena, es decir una evaluación exactas del estado del producto (o reportar las razones que lo evitaron); asegurar que se siguen los procedimientos establecidos par las revisiones.
Recorder (Apuntador)	CP-RC	Asegurar que la información capturada refleja en forma precisa los comentarios y resultados acordados en la sesión; tomar notas en tiempo real y en medio visible por todos los participantes.
Reviewer (Revisor)	CP-RV	Prepararse bien para la revisión; ser honesto y actuar en forma recta; enfocarse en temas técnicos relevantes; ser considerado y pensar bien sus comentarios. Todos los revisores comparten la responsabilidad de la revisión, determinando si el producto está completo.
Observer (observador)	CP-OB	Asistir puntualmente y permanecer durante toda la sesión; no intervenir a menos que se le solicite.

Actividades
Se asocian a los objetivos y describen las tareas y roles responsables

Rol	Descripción
	A1. Calendarizar la sesión (materiales requeridos mínimo dos días antes)
CP-PR	A1.1. Productor solicita sesión al terminar el producto
CP-PR TE-QPM	A1.2. Acuerdo de fecha y hora
CP-SL	A2. Recolectar material relevante, asegura producto listo para inpección
TE-QPM CP-SL	A3. Seleccionar participantes
CP-SL CP-PR	A4. Distribuir materiales necesarios
	A5. Guiar la sesión (o darla por terminada)
CP-SL	A5.1. Inicio de sesion
CP-RC CP-RV	A5.2. Recopilación de comentarios

	A5.3. Cierre de sesión
CP-SL	A6. Obtener consenso
CP-RC	A7. Revisar los comentarios
CP-RC	A8. Completar los reportes de la sesión
CP-RC	A9. Transmitir el reporte
CP-SL TE-QPM	A10. Verificar los seguimientos

Diagrama de flujo de trabajo

En la figura 8-3 observamos que el grupo que trabaja en modo cooperativo para revisar técnicamente, un modelo o producto cualquiera, distribuye a sus actores en los roles mostrados en la parte inferior del diagrama de estados y sigue las reglas estipuladas para este modo de tarbajo en grupo. Una sesión de revisión técnica se programa a solicitud del grupo **Producer** cuando termina su producto (desarrollado colaborativamente).

También se programan revisiones técnicas cuando se terminan todos los modelos de cada *sub-proceso* del modo En Equipo, en puntos de medición de avance (**milestones**) preestablecidos al terminar el:

- Folder de requerimientos y del plan de prueba de aceptación
- Folder de diseño y del plan de prueba de integración
- Folder de unidad
- Resultado de prueba de integración
- Resultado de prueba de aceptación y documentación de usuario

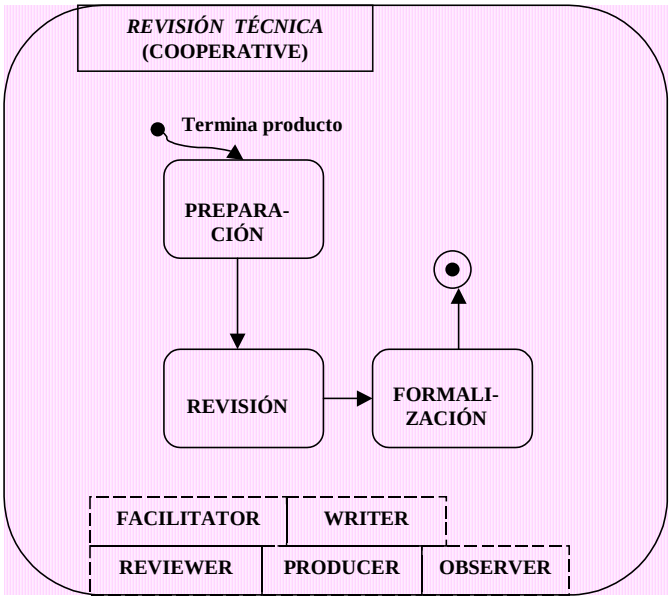


Figura 8-3 Sub-procesos y Roles del Trabajo Cooperativo (Cooperative).

En el sub-estado Formalización se producen documentos con dos destinos:

- Documentos técnicos.- con los comentarios y métricas técnicos, que circulan libremente entre los roles del equipo de desarrollo (modo En Equipo), principalmente dirigidos al grupo **producer**.
- Documentos administrativo.- dirigido al **Project Manager** con la conformación del grupo de revisión y su desempeño en la sesión, datos identificando al producto revisado y la decisión tomada con respecto al producto.

En la figura 8-4 se muetra el modo Cooperativo, que se sigue en las revisiones técnicas, documentando y resolviendo conflictos entre las opiniones de los participantes. Se utiliza para revisar formalmente cualquier modelo, resultado o artefacto producido en el proceso de desarrollo **MDP**. Favor de ver § 5.8 para el detalle de las etiquetas asociadas a las relaciones (flechas).

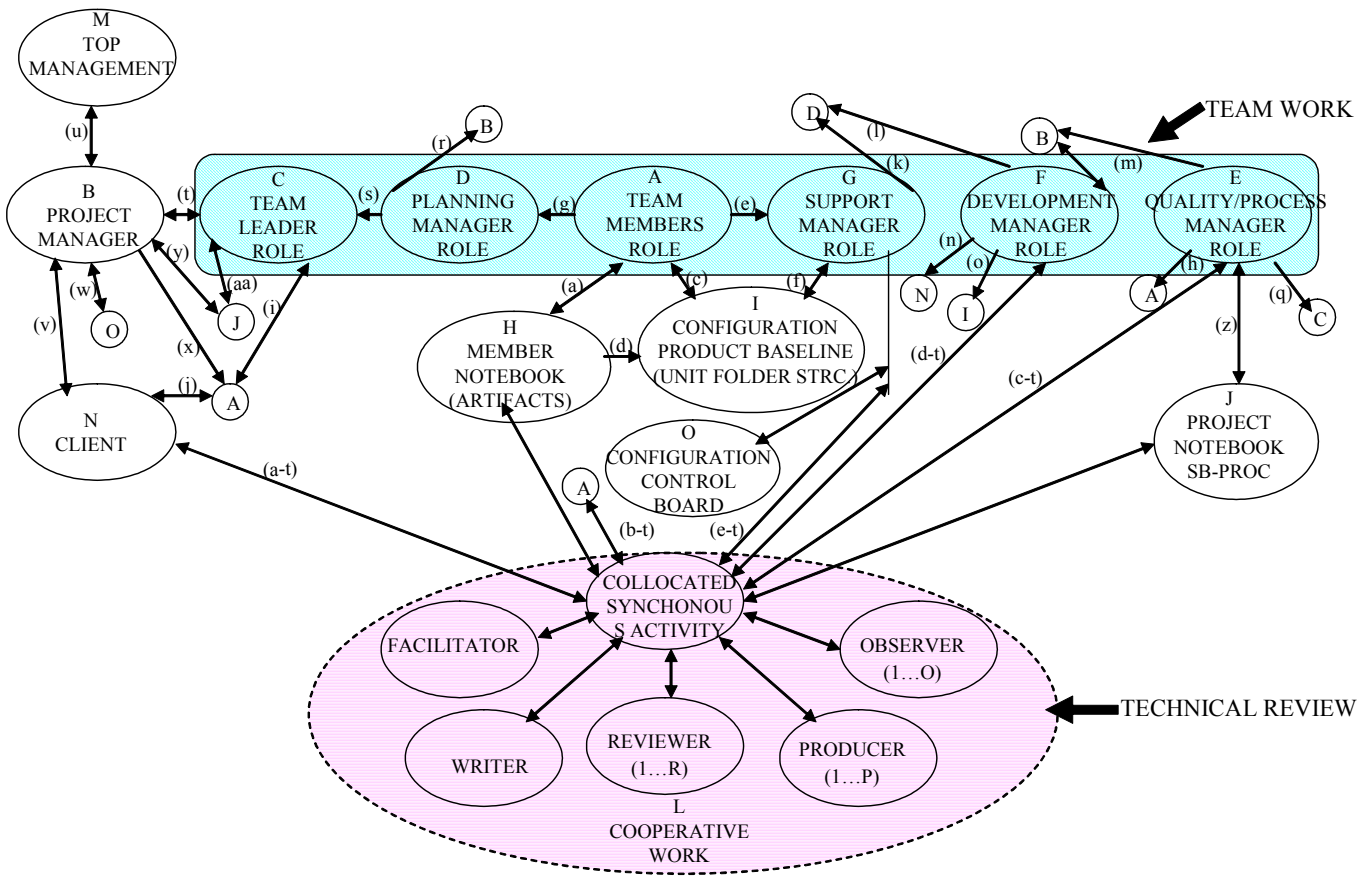


Figura 8-4 Representación estática del modo de trabajo Cooperativo y su relación con el modo de trabajo En Equipo.

Verificaciones y Validaciones (relacionadas al modo En Equipo)

Verificación o validación	Actividad	Producto	Rol	Descripción
P-REQUIRE-MENTS (Reque-mientos, Análisis)	A8	carpeta de requerimientos y del plan de prueba de aceptación	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (Validación asociada, con el cliente)
SP-ANÁL-ISYS & DESIGN (Análisis y Síntesis)	A8	carpeta de diseño y del plan de prueba de integración	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD
SP-IMPLEMEN-TATION (Implemen-tación)	A7	diseño detallado y plan de prueba de la unidad	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD
	A9	folder de unidad	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (verificación)
	A11	folder de unidad	TE-DM TE-QPM (CP-PR = CL-DR = CL-CDR = TE-TM)	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (validación)
SP-TEST (Prueba	A6	resultados de prueba de integración	TE-DM TE-QPM TE-TL	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (verificación)

A9	Resultados de prueba de aceptación y documentación de usuario	TE-DM TE-QPM TE-TL	MDP-INS-DET, MDP-INS-SUM, MDP-LOGD (validación)

8.4Proceso: (MP-CT) Modo de Proceso de Control

Definición general del proceso

Proceso	MDP-CT: Modo de Proceso: de Control.
Categoría/Modo	Modos de trabajo en grupo.
Propósito	Establecer el mecanismo básico para comunicación, planificación y toma de decisiones del equipo de desarrollo. Usualmente se realiza cada semana en forma sistemática. Coordina el trabajo de control con que se recopilan y analizan los datos del desempeño del equipo incluyendo indicadores del proceso y del producto. Los datos de la semana anterior y del ciclo de desarrollo actual se utilizan para planificar las tareas a realizar la semana siguiente.
Descripción	Este modo de trabajo supone: 1) una actividad colocalizada síncrona (todos los participantes en: el mismo lugar, y el mismo tiempo); 2) dicha actividad realizada por todos los miembros del equipo de desarrollo (del modo En Equipo). Existiendo tres *roles*: **meeting leader**, **meeting recorder** y **meeting member**.
Objetivos	O1 - Obtener un informe exacto y verás del estado del proceso de desarrollo. O2 - Obtener un informe acerca de las tareas comprometidas para realizarse la semana siguiente O3 -. Documentar el desempeño del equipo, los riesgos e inquietudes de los participantes y las decisiones tomadas.
Indicadores	Para las áreas de proceso de la categoría de Ingeniería, se consideran los indicadores siguientes: Contenido de MDP-SUMQ: I1 Percent Defect Free I2 Defect/page I3 Defects/KLOC I4 Defect ratios I5 Development time ratios (%) I6 A/FR

I7 Personal review rates

I8 Inspection rates

I9 Tasas de inserción de defectos (Defectos insertados por hora)

I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)

I11 Yields de las fases (Defectos eliminados en la fase / Defectos eliminados en la fase + ESCAPES)

I12 Yield del proceso

Para las demás áreas se establecen ad hoc, documentándose los indicadores utilizados.

Metas cuantitativas

Para las áreas de proceso de la categoría de Ingeniería, se consideran los indicadores siguientes:

I1 Percent Defect Free (PDF)

Compilación	> 10%
Prueba de unidad	> 50%
Prueba de integración	> 70%
Prueba de sistema (aceptación)	>90%

I2 Defect/page

I3 Defects/KLOC

Total de defectos inyectados	76 – 150
Compilación	< 10
Prueba de unidad	< 5
Integración	< 0.5
Prueba de sistema	< 0.2

I4 Defect ratios

Defectos revisión DLD/Defecto en prueba unidad	> 2.0
Defectos revisión de código/Defectos Compilación	> 2.0

I5 Development time ratios (%)

Inspección de requerimientos/Tiempo requerimientos	> 0.25
Inspección HLD/Tiempo H	> 0.5
DLD/Tempo codificación	> 1.0
Revisión DLD/Tiempo DL	> 0.5
Revisión codificación/Tiempo codificación	> 0.5

I6 A/FR

I7 Personal review rates

I8 Inspection rates

Páginas de requerimientos/Hora	< 2
Páginas HLD/Hora	< 5
Líneastexto DLD/Hora	< 100
LOCs/Hora	< 200

I9 Tasas de inserción de defectos (Defectos insertados por hora)

Defectos requerimientos/Hora	0.25
Defectos HLD/Hora	0.25
Defectos DLD/Hora	2.0
Defectos código/Hora	4.0
Defectos compilación	0.3
Defectos prueba unidad	0.2

I10 Tasas de eliminación de defectos (Defectos eliminados entre tiempo)

Defectos inspección requerimientos/Hora	0.5
Defectos inspección HLD/Hora	0.5
Defectos inspección DLD/Hora	2.0
Defectos revisión código/Hora	6.0
Defectos inspección código/Hora	1.0

I11 Yields de los sub-procesos

Inspección de requerimientos	- 70%
Revisiones e inspecciones de diseño	- 70%
Revisiones e inspecciones de código	- 70%
Compilación	- 50%
Prueba sistema con 5 o menos defectos/K LOCs	- 90%
Prueba sistema a < 1.0 defectos/KLOC	- 80%

I12 Yield del proceso

Antes compilación	>75%
Antes prueba unidad	> 85%
Antes de integración	> 97.5%
Antes de prueba sistema	> 99%

Responsabilidad y autoridad

Reglas para preparación:

Cada uno de los tres roles: 1) **meeting leaderr**; 2) **meeting recorder**, y 3) **meeting member**, tiene una descripción que detalla sus metas y sus actividades principales; así como una tabla que enumera detalladamente las actividades que coordina y las

actividades en que participa.

Deben asistir todos los miembros del equipo de desarrollo (modo En Equipo).

Todos los miembros del equipo de desarrollo le entregan al **planning manager** (modo En Equipo), cuando menos ocho horas hábiles antes de la reunión las formas con los datos del estado del proceso (formas **MDP-TASK**, **MDP-SCHEDULE** *y* **MDP-WEEK**).

El **planning manager** (modo En Equipo) produce reportes integrados con los datos de los miembros del equipo (forma **MDP-WEEK**)

Reglas para su desarrollo):

El **team leader** (modo En Equipo) se desempeña como **meeting leader** (modo Control) y controla el tiempo; Distribuye una agenda para la reunión con los temas que los asistentes sugirieron previamente.

El **quality/process manager** (modo En Equipo) funge como **meeting recorder** (modo Control) y registra todos los tópicos que se tratan en la sesión.

Si algún miembro del equipo no esta preparado para su reporte o tema a tratar se pospone la reunión.

Cada miembro del equipo reporta su trabajo en el *rol* que ocupa (modo En Equipo) y su trabajo de desarrollo.

Generalmente el *rol* que reporta primero es el **development manager** (modo En Equipo).

El **meeting recorder** (modo Control) escribe sobre un medio visible por todos los asistentes a la reunión los comentarios que surjan durante la sesión . Además realiza los reportes de la sesión.

La responsabilidad del grupo, que participa en la sesión de **control work,** es obtener con calidad los reportes del trabajo hecho en la reunión, con los comentarios y las decisiones tomadas.

La revisión debe concentrarse en aspectos del estado del proceso estableciendo “que” se puede mejorar sin indicar “como” hacerlo.

Los participantes deben tomar una actitud crítica constructiva; no destructiva. Haciendo comentarios únicamente acerca del proceso.

Los actores involucrados deben estar abiertos a los comentarios, para mejorar la calidad del proceso.

Reglas para la toma de decisiones:

Se toman decisiones acerca de:

Las tareas con compromisos establecidos y que se han llevado mas tiempo o recursos de los establecidos.

	Todos los riesgos y asuntos (formas **MDP-ITL, MDP-PIP**) que se han documentado y su evolución. Discrepancias y tendencias descubiertas en las bitácoras y resúmenes (formas MDP-LOGD, MDP-LOGT, MDP-LOGTEST, MDP-CCR, MDP-SUMDI, MDP-SUMDR,MDP-SUMP, MDP-SUMQ, MDP-SUMS, MDP-SUMT, MDP-SUMTASK). Identificación y asignación de las tareas de la siguiente semana. Determinar asuntos, comentarios, y resúmenes a incluir en el reporte de la sesión. Reglas relacionadas con el modo En Equipo: Todas las bitácoras y formas de control que aportan los datos de control acerca del proceso de desarrollo se llenan y sumarizan semanalmente. Las **status meetings** (modo de control) se realizan semanalmente. Para analizar los datos sumarizados (del modo En Equipo), identificando tendencias y decidiendo que acciones correctivas tomar. En las **status meeting**s (modo de cortrol) se identifica el avance del desarrollo (modo En Equipo) y se planifican las tareas para la siguiente semana. Se reporta el estado del proceso (modo En Equipo) al **project manager**.
Subprocesos (opcional)	Inicio con agenda.- Reporte de roles y desarrollo.- Toma de decisiones y planificación.- Formalización.-
Procesos relacionados	Modo de trabajo En Equipo Modo de trabajo Cooperativo Modo de trabajo Colaborativo

Entradas

Nombre	Descripción	Fuente
MDP-LOGT Time Recording Log	Para registrar el tiempo ocupado en cada tarea del proyecto	TE-TM
MDP-LOG Defect Recording Log	Para mantener datos acerca de los defectos encontrados y corregidos	TE-TM
MDP-LOGTEST Test Log	Contiene un sumario de las pruebas realizadas y los resultados obtenidos	TE-TM TE-DM TE-QPM

MDP-INFO Member Information Sheet	Para describir los intereses y experiencia de un individuo	TE-TM
MDP-STRAT Strategy Form	Para registrar las decisiones estratégicas	TE-PM
MDP-PEER Team and Peer Evaluation	Contiene la evaluación del equipo y de los compañeros	TE-TM
MDP-PIP Process Improvement Proposal	Para registrar problemas del proceso e ideas de mejora	TE-QPM
MDP-SUMDI Defect Injected Summary	Sumariza datos acerca de defectos insertados en las partes de un ensamble	TE-PM
MDP-SUMDR Defects Removed Summary	Sumariza los datos acerca de deferctos eliminados de las partes de un ensamble	TE-PM
MDP-SUMP Plan Summary	Mantiene datos planeados y reales para ensambles del producto	TE-PM
MDP-SUMQ Quality Plan	Mantiene datos de calidad planeados y reales para partes y ensambles	TE-PM TE-QPM
MDP-SUMS Size Sumary	Sumariza datos acerca del tamaño del producto	TE-PM
MDP-SUMT Development Time Summary	Sumariza datos acerca del tiempo de desarrollo ocupado en partes de los ensambles	TE-PM
MDP-SUMTASK Task Summary	Mantiene datos para facilitar la planeación y seguimiento de las tareas	TE-PM
MDP-TASK Task Planning Template	Para estimar el tiempo de desarrollo, el valor planeado y la fecha de terminación de cada tarea	
MDP-WEEK Weekly Status Report	Para preparer el reporte del estado, semanal	TE-PM TE-TL
MDP-SCHEDULE Schedule Planning Template	Para registrar horas estimadas y reales semanalmente	TE-PM
MDP-CSR Comfiguration Status Report	Para proveer información acerca del estado seminal del sistema para administración de configuración del software (SCM)	TE-SM
MDP-ITL Issue Traking Log	Para registrar y dar seguimiento a los riesgos y temas pendientes del proyecto	TE-TL

MDP-INS-DET Inspection Report-Detail	Recolecta los datos detallados acerca de una inspección/revisión	TE-QPM
MDP-INS-SUM Inspection Report-Sum	Para reportar el sumario del análisis hecho a los datos de una inspección/revisión	TE-QPM
MDP-CCR Configuration Change Request	Para enviar un elemento al Comité de Control de Cambios (CCB) para su inclusión en la líneabase	TE-SM

Salidas

Nombre	Descripción	Destino
Reporte de Estado: sumario (resultado de la sesión de estado del proceso)	Reporte de horas del proyecto, valor ganado, horas de tareas completadas, horas ocupadas en cada rol, seguimiento de riesgos y oportunidades de mejora.	Administradores (PROMA, CLIENT, TOPMA)
Reporte de Estado: detallado (resultado de la sesión de estado del proceso)	Temas y comentarios detallados, propuestas de mejora, decisiones tomadas.	(TE-TM = CL-DR, CL-CDR)
Formas de entrada (MDP-…) actualizadas y verificadas	Formatos llenos, revisados y reportados durante la sesión.	Carpeta del proyecto

Referencias bibliográficas [SK01] [IEEE-1028]

GM04	**Moisés González-García, Ana María Martínez-Enríquez; The Architectural and Group Development Method: an Experimentation; Workshop on Quantitative Techniques for Software Agile Process (QUTE-SWAP), ACM SIGSOFT 2004 / FSE-12 (Foundations on Software Engineering); Oct. 31 – Nov. 6, 2004.**
SK01	**Andrew P. Snow, Mark Keil; The Challenges of Accurate Project Status; 34 Annual Hawaii International Conference on System Sciences (HICSS-34), Vol. 8, 2001, pp. 3133-3142**
IEEE-1028	IEEE Std. 1028 - 98; IEEE Standard for software reviews; IEEE Computer Society, Software Engineering Standards Committee, March 1998.

Prácticas

Identificación de roles involucrados y capacitación requerida

Rol	Abreviatura	Capacitación

Session leader	CT-SL	Responsible de: 1) tareas administrativas de la sesión, 2) planificación y preparación de la sesión, 3) conducir la reunión en forma ordenada logrando sus objetivos, 4) emitir los reportes de la sesión.
Session recorder	CT-RC	Documentar anomalías, elementos de acción y las recomendaciones hechas por los asistentes.
Session member	CT-TM	Proveer la información necesaria para la mejora contínua y administración del proceso.

Actividades

Se asocian a los objetivos y describen las tareas y roles responsables

Rol	Descripción
(CT-SL = TE-TL)	A1. Revisión de agenda
(CT-RC = TE-QPM)	A1.1. Se revisa la agenda y se pregunta por adiciones y cambios
(CT-SL = TE-TL)	A1.2. Verificación de que todos los miembros del equipo (modo En Equipo) están preparados y de pospone la reunión si no es así
(CT-RC = TE-QPM)	A2. Reporte de Roles
Todo rol	A2.1 Asuntos o preocupaciones del rol
Todo rol	A2.2 Estado de tareas o actividades asociadas al rol
Todo rol	A2.3 Estado de cualquier riesgo o inquietud que el ingeniero está siguiendo
TE-DM	A2.4 Artefactos diseñados, revisados, inspeccionados, implementados y probados
TE-PM	A2.5 Horas/hombre del equipo y estado del valor ganado contra lo planeado
TE-QPM	A2.6 Todos los defectos de las inspecciones, y pruebas de integración y aceptación
TE-QPM	A2.7 Porcentaje de ingenieros siguiendo el proceso
TE-QPM	A2.8 Cualquier problema de calidad
TE-SM	A2.9 Artefactos enviados a control de configuración (en la semana), cambios hechos e inventario del sistema

(CT-RC = TE-QPM) TE-TM	A3. Reporte del estado del desarrollo
	A3.1 Las horas trabajadas esta semana y ciclo, comparadas con lo planeado
	A3.2 Valor ganado esta semana y ciclo comparado con lo planeado
	A3.3 Tiempos de las tareas teminadas esta semana y los tiempos planeados
	A3.4 Tareas a realizar la próxima semana
	A3.5 Horas a trabajar la semana siguiente
	A3.6 Cualquier área problema o tema de interés al equipo
(CT-SL = TE-TL) (CT-RC = TE-QPM)	A4. Cierre de la reunión
	A4.1 Verifica que se reportaron todas las tareas comprometidas
	A4.2 Verifica que se reportaron todos los riesgos e inquietudes
	A4.3 Se asegura que las tareas para la semana siguiente se identificaron y asignaron
	A4.4 Discute los elementos a incluir en el reporte semanal del equipo

Diagrama de flujo de trabajo

En la figura 8-5 observamos el trabajo en grupo que se realiza en una reunión para control de avance del proceso, desempeñando los actores involucrados los roles mostrados en la parte inferior del diagrama de estados. Se siguen las reglas del modo de Control, usualmente programando, la reunión de control de avance del proceso, semanalmente en un horario de bloque preestablecido.

La reunión se aprovecha para presentar los reportes del desarrollo de modelos o productos de cada uno de los miembros del equipo, así como el reporte semanal de cada *rol* del modo En Equipo. La segunda parte de la sesión se utiliza para tomar decisiones y planificar el desarrollo de la semana siguiente, en función de los datos de desempeño presentados.

En el sub-estado Formalización se genera el Reporte de Desempeño del Equipo que se pasa al **Project Manager** y se mantiene al día el cuaderno o expediente del proyecto (**Project Notebook**).

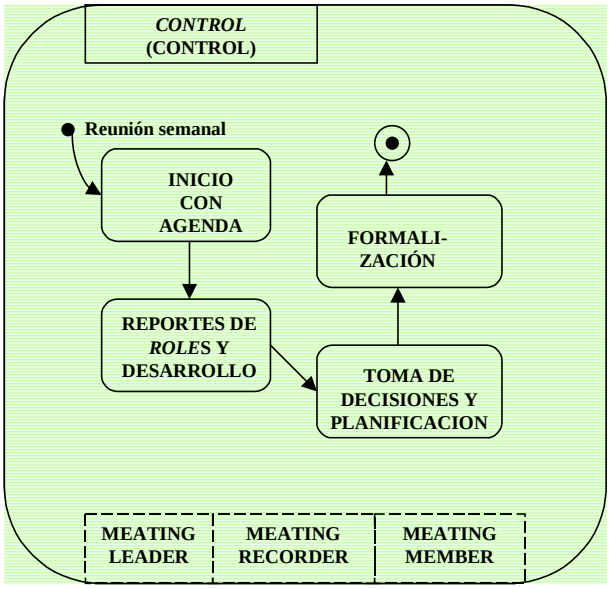


Figura 8-5 Sub-procesos y Roles del Trabajo de Control.

La figura 8-6 muestra e modo de Controll mediante el cual se realiza el procedimiento para comunicación, planificación y toma de decisiones del equipo de desarrollo. Usualmente se realiza cada semana en forma sistemática.

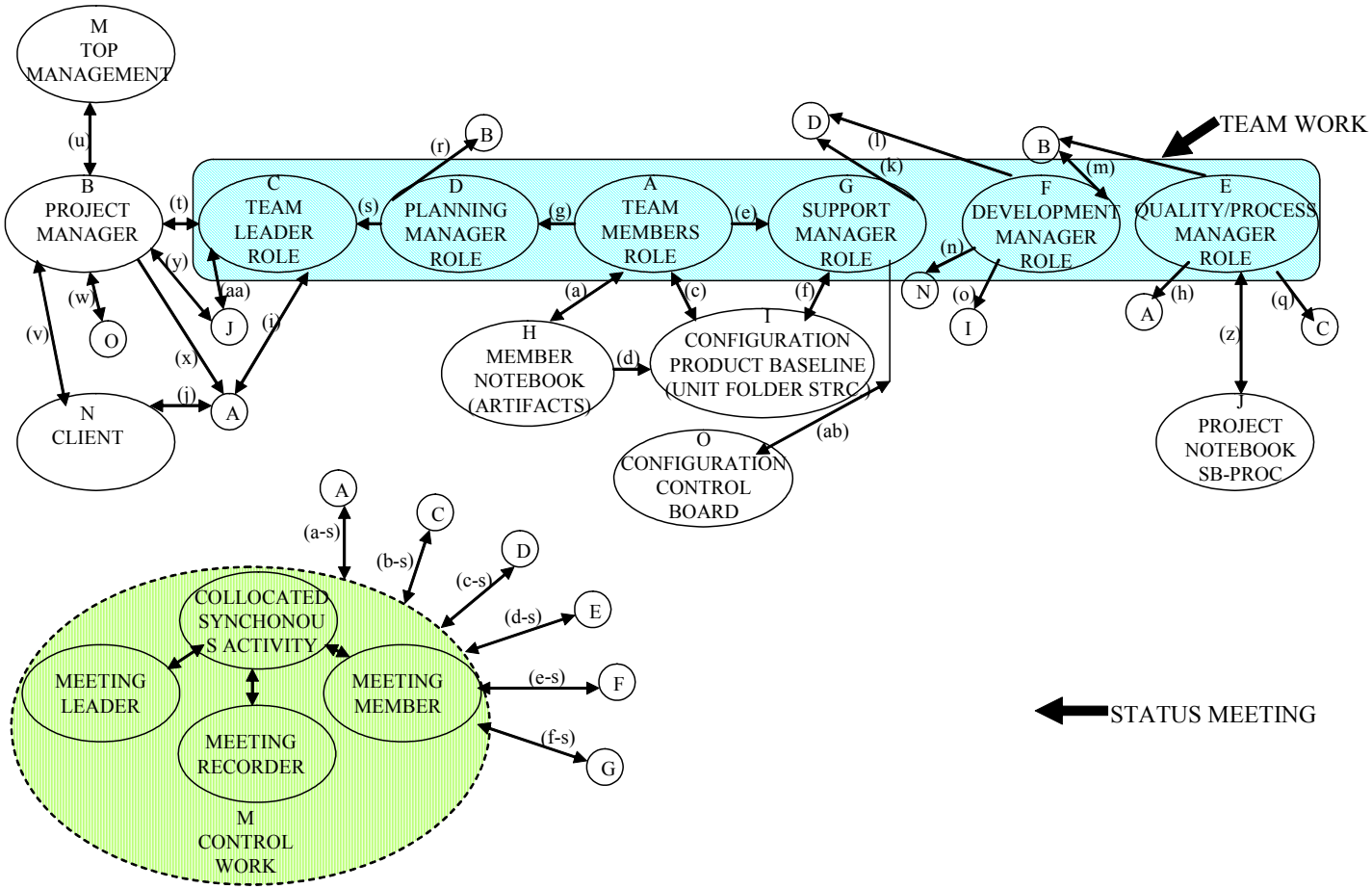


Figura 8-6 Representación estática del modo de Control y su relación con el modo En Equipo.

El modo de Control coordina el trabajo de control mediante el cual se recopilan y analizan los datos del equipo acerca del trabajo realizado durante la semana anterior y durante el ciclo de desarrollo actual, hasta la fecha. Utilizando los datos recopilados de desempeño para planificar las tareas de la semana siguiente.

9 ANEXO B.- Modelo de Referencia para Productos de Software

9.1Introducción

Uno de los resultados de esta investigación es el modelo Trabajo Cooperativo: Almacenamiento, Lógica y Recursos / Cooperative Work: Storage Logic and Resources (CWSLR) [GJM99]. CWSLR sirve de referencia, durante el diseño del Conjunto de Productos PS (capítulo cuatro), para satisfacer los requerimientos siguientes:

Nombre	Descripción
Conceptos de grupo	Se requiere incluir los elementos necesarios para soportar el trabajo en grupo, en el software desarrollado (ej. memoria de grupo, cooperación).
Modelos de Referencia	Durante la construcción de un producto de software se requieren modelos de referencia para identificar los productos-del-trabajo a desarrollar.
Transformación entre modelos	Se requiere identificar las transformaciones entre modelos y habilitar su realización automática.
Modelado conceptual del negocio	Para producir un Modelo de Proceso del Negocio se requieren elementos explícitos en UML, que sean parte del Modelo Conceptual para el componente en desarrollo.

El modelo CWSLR describe sistemas de información cooperativos (CWIS: Cooperative Work Information System). El modelo define de la estructura, los elementos incluidos y las relaciones entre las partes. Se definió con el propósito de servir como referencia para la identificación de los componentes de la arquitectura de software, de una aplicación computacional, evitando omitir los elementos del modelo.

CWSLR se ubica, en el tercer nivel de abstracción de un entorno de desarrollo de software. A este ambiente se le denominó Proceso de Software mediante Trabajo Cooperativo / Cooperative Work Software Process (CWSP) [GM02].

En el tercer nivel del entorno CWSP (figura 9-1), se incluye junto al modelo CWSLR un estilo arquitectónico (§ 9.8) basado sobre la Arquitectura Dirigida por Modelos / Model Driven Architecture (MDA) [S05]. El estilo arquitectónico EA(MDA, …), ademas de incluir a la MDA, combina otros estilos. La elipsis "…" supone cualquier estilo arquitectónico [GLNS03].

Integramos en los sistemas de información que soportan a la cooperación (CWIS), a los sistemas de información (IS) y a los sistemas de trabajo cooperativo soportados por computadora (CSCW) (§ 9.2). Para representar sistemas CWIS definimos el modelo Trabajo Cooperativo: Almacenamiento, Lógica y Recursos (CWSLR) [GJM99] (§ 9.3). CWSLR incluye tres dimensiones (§ 9.4), una para representar los componentes requeridos en la aplicación (§ 9.5), la se-

gunda para ubicar los componentes en una red de computadoras (§ 9.6) y la tercera para ubicar la aplicación en la organización (§ 9.7).

La determinación de la arquitectura del software del componente a desarrolla, se realiza con ayuda el modelo CWSLR (§ 9.8). Otro elemento generado a partir del CWSLR es el catálogo para tipos de componentes mínimos (§ 9.9), útil para generar una clasificación de componentes.

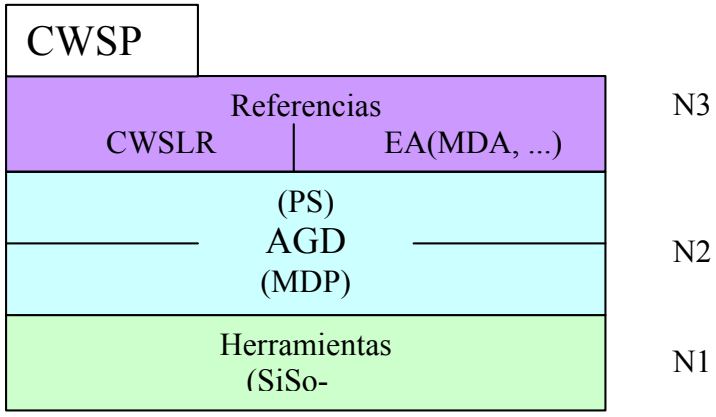


Figura 9-1 Estructura del ambiente de desarrollo de software CWSP y los elementos principales de cada nivel.

9.2Modelo para Soportar al Trabajo Cooperativo

El modelo CWSLR [GJM00] [GJM99] proporciona una referencia, para la generación del conjunto de productos-del-trabajo, necesaria para desarrollar un producto de trabajo. Se creó identificando tres dimensiones: de almacenamiento (**STORAGE-D**), lógica (**LOGIC-D**) y Recursos (**RESOURCE-D**). Inicialmente, se diseña la estructura de la dimensión **STORAGE-D** [GMJ98]. A partir de este estado inicial, el modelo evolucionó homogeneizando sus dimensiones y validándolo con otros modelos relacionados [GJM99]: como es el caso del Modelo de Referencia para Proceso Distribuido Abierto (RM-ODP). En una tercera etapa

se completan las tres dimensiones del modelo con un nivel de detalle similar (entre ellas) [GJM99B].

En el modelo CWSLR se incluyen las áreas: a) Proceso; b) Diálogo; c) Memoria Organizacional[35]; d) Coordinación, y e) Cooperación; que se requieren para el trabajo en grupos, en el contexto de redes de computadoras con aplicaciones Web. Estas áreas son adicionales en relación a las existentes en los modelos relacionados siguientes:

1. Para sistemas distribuidos: ISO/IEC 10746-1 a 4 Procesamiento Distribuido Abierto – Modelo de Referencia / Open Distributed Processing – Reference Model (RM-ODP) [ASF04] [BD01] [ODP98], establecido como norma internacional
2. Los marcos de referencia internacional, para objetos, del Grupo de Administración de Objetos / Object Management Group (OMG): **a**) Ubicación de Objetos de Negocio / Business Object Facility [S97], **b**) Objetos Comunes / Common Objects [OMG02A] [OMG95], **c**) Arquitectura Dirigida por Modelos / Model Driven Architecture (MDA) [OMG03A], y **d**) Ubicación de Meta Objetos / Meta Object Facility (MOF) [OMG97B]
3. Los marcos de referencia internacionales, para sistemas distribuidos, de El Grupo Abierto / The Open Group y la OMG: **a**) Marco de Arquitectura del Grupo Abierto / Open Group Architecture Framework (TOGAF) [BSW04] [OG02] [OG03], **b**) el Entorno de Computo Distribuido / Distributed Computing Environment [OG97], y **c**) el Modelo de Referencia para Administración de Sistemas / System Management: Reference Model [OG93]

El modelo CWSLR tiene como finalidad facilitar el diseño y especificación de un sistema de información, provee un marco para ubicar todos los elementos de los tipos de componente utilizados. De esta forma, da la posibilidad de enumerar elementos faltantes. Estos elementos vacíos son una guía para establecer metas y tareas, que se requiere realizar en el sistema de información. El desempeño de las personas en la organización puede mejorar, pues completan todos los elementos.

Mediante el modelo CWSLR se estructura el conjunto de objetos con los que se construyen los componentes de software. Los objetos, incluidos en tres dimensiones del modelo, se clasifican en:

1. Objetos-Información: Almacenan datos (basados-en-registros o basados-en-documentos) y sus métodos de mantenimiento (ej. manejadores de base de datos).
2. Objetos-Software: Almacenan datos y métodos de servicios de software (ej. soporte a grupos, planificación estratégica y herramientas para desarrollo).

[35] Memoria organizacional – información acerca de decisiones tomadas y la forma de resolver problemas [WU91]

3. Objetos-de-Estante: Almacenan objetos-software, preexistentes, usados para soportar la toma de decisiones y la resolución de problemas.

Para guiar la cooperación entre desarrolladores, se asume que se coordina por resultados[36] [CG90] [CG89]. Este tipo de coordinación proporciona lineamientos para identificar las relaciones entre los diagramas (modelos) a utilizar en las dimensiones del CWSLR. Debido principalmente a que cuando se cuenta claramente con los objetivos, materializados en los resultados, se facilita la evaluación y mejora del desempeño.

9.3Estructura del modelo CWSLR

Cada una de las tres dimensiones del modelo CWSLR (figura C.2) tiene cinco niveles, tres de ellos corresponden a una estructura de datos (su identificador tiene el sufijo –S (Structure) y se representan mediante un círculo negro), alternados con dos niveles de funciones (su identificador tiene el sufijo –F (Function) y se representan mediante una flecha). Resultando el orden siguiente S-F-S-F-S. Las dimensiones son:

- De almacenamiento **STORAGE-D:** Representa el almacén de los datos, las instrucciones y los artefactos que definen esta dimensión. Se considera el componente de software contenido completamente en una sola computadora (aunque se trate de una aplicación distribuida). Esta dimensión se explica en § 9.5.
- De recursos **RESOURCE-D:** Incluye datos acerca del componente software: distribuido en la red de computadoras y los equipos periféricos que utiliza. Además contiene información relativa a los inmuebles, los empleados o trabajadores que usan al componente, los servicios y los recursos de oficina, involucrados en los procesos del dominio de aplicación. Esta dimensión se explica en § 9.6.
- De lógica **LOGIC-D:** Describe la arquitectura del componente y su construcción, además de contener la información acerca de su ubicación dentro de la organización, y los protocolos empleados en la cooperación. Abarca información acerca de las relaciones operativas y la organización. Esta dimensión se explica en § 9.7.

Los niveles-de-estructura de las tres dimensiones se relacionan como se muestra en la figura 9-3. Las flechas verticales dirigidas hacia arriba, en cada una de las dimensiones, representan

[36] Mapeando el método de Carriero y Gelernter para escribir un programa coordinado de una tarea más abstracta, como es desarrollar un sistema de información que soporte la cooperación.

un conjunto de las funciones necesarias para organizar los elementos que pertenecen al nivel inferior. Los elementos del nivel (nodo) inferior se organizan mediante las funciones representadas por la flecha que conecta con el nivel (nodo) superior: En el nodo sauperior (nodo) se ubican los elementos de información que relacionan a los elementos del nivel inferior (nodo).

Ejemplo de la dimensión **STORAGE-D**, en el que se describe como se transforman los elementos de los niveles-de-estructura **BASIS-S**, **PROCESS-S** y **MEMORY-S**.

Se describe el ejemplo mediante las aseveraciones siguientes: 1) Se tienen los datos de los alumnos en el nivel **BASIS-S**; 2) Mediante funciones del proceso "Inscripciones" en el nivel **OPERATE-F** se mantiene una lista de "Alumnos Inscritos", ubicada en el nivel **PROCESS-S,** 3) El reporte "Alumnos Inscritos" se evalúa en la revisión específica "Administrativa-032" con funciones del nivel **DIALOG-F** y 4) el resultado de la revisión se incluye en el reporte "REV-ADM-032" y se almacena en el nivel **MEMORY-S**.

En el nivel-de-estructura superior se mantienen únicamente los atributos de las relaciones entre los elementos del nivel-de-estructura inferior, así como las restricciones que caracterizan a sus ordenamientos. No se duplican los atributos de los elementos en el nivel-de-estructura inferior, para economizar en el uso de recursos.

En el ejemplo anterior: la lista "alumnos Inscritos" mantiene en una estructura a los identificadores de los alumnos que cumplen con la condición de estar inscritos en un periodo escolar, no duplica los datos que caracterizan a cada uno de los alumnos. La lista almacenada en el nivel PROCESS-S contiene solamente apuntadores a los datos de los alumnos almacenados en el nivel BASIS-S.

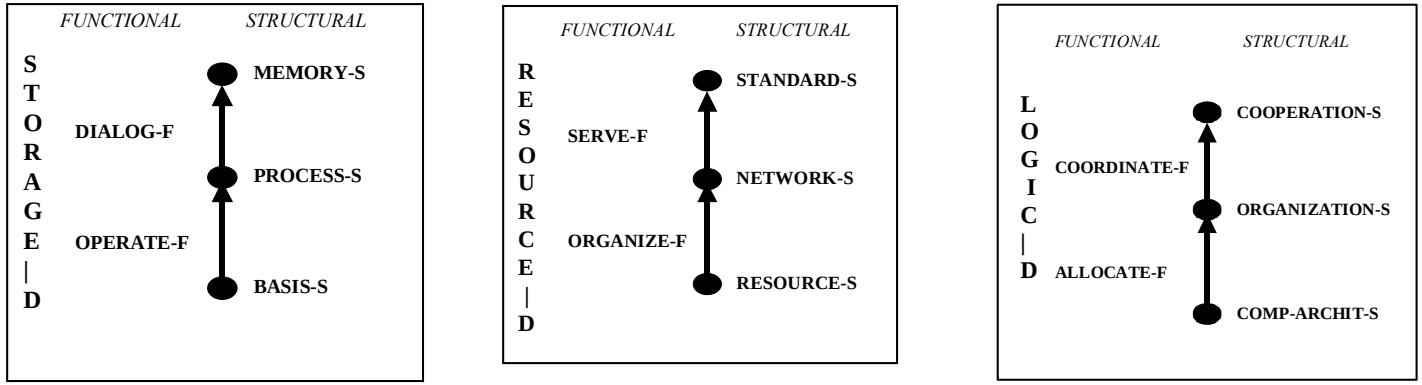


Figura 9-2 Las tres dimensiones del modelo CWSLR

Las flechas horizontales relacionan niveles-de-estructura en la misma posición en dimensiones adyacentes (ej. primer nivel-de-estructura en dimensión **STORAGE-D** se relaciona con el primer nivel-de-estructura de la dimensión **RESOURCES-D**), representando una relación explícita entre elementos de los dos niveles relacionados.

Para comprender el significado de estas relaciones, se muestran las conexiones existentes entre los tres niveles-de-estructura: **PROCESS-S** (de **STORAGE-D**), **NETWORK-S** (de **RESOURCE-D**) y **ORGANIZATION-S** (de **LOGIC-D**). El ejemplo, extraído de una aplicación específica, consiste en: el proceso de "Nómina de Confianza", se ubica en el nodo "Servidor-Recursos-Humanos, que pertenece al área del negocio "Gerencia de Recursos Humanos".

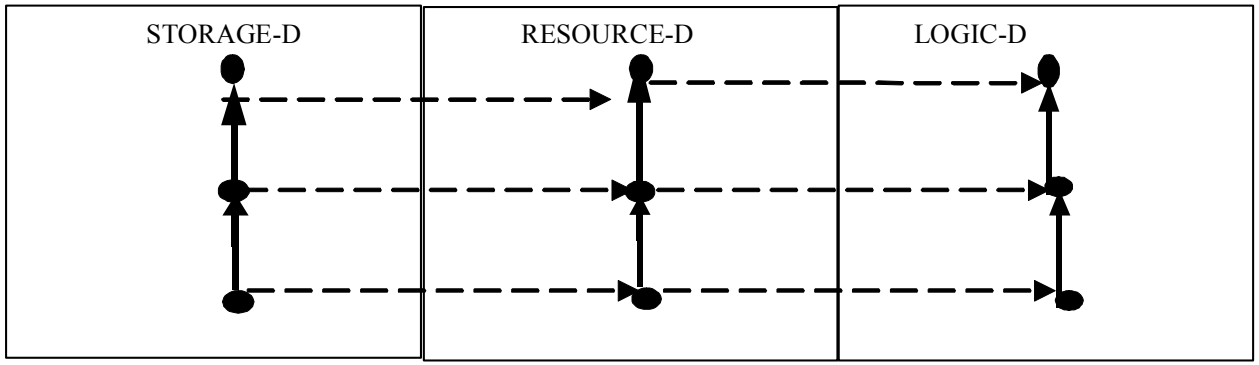


Figura 9-3 Estructura horizontal del modelo CWSLR

El nivel-de-estructura a la derecha de la flecha horizontal mantiene sólo los atributos de las relaciones existentes entre los elementos cuyos datos están almacenados en un nivel-de-estructura posicionado a la izquierda de la flecha (ej. El proceso "Nomina de Confianza" de PROCESS-S se ubica en el nodo Servidor-Recursos-Humanos de NETWORK-S). También el nivel-de-estructura de la derecha, mantiene las restricciones que dan significado a las relaciones (ej. computadora con 20 Gbytes en disco). Para economizar en el uso de recursos, no se duplican en el nivel-de-estructura de la derecha los atributos de los elementos existentes en el nivel-de-estructura de la izquierda.

Después de describir la estructura general del modelo CWSLR, se establece a continuación el concepto de tipo de componente mínimo, que nos proporciona las características del elemento constructivo más elemental. Esto es útil dado que un sistema de información (IS) puede construirse ensamblando varios componentes (ej. nómina, adquisiciones, ventas, etc) cuya funcionalidad se almacena en **STORAGE-D**. Para identificar estos componentes conviene contar con el concepto de tipo de componente mínimo (figura 9-4).

En el modelo CWSLR, un componente mínimo se puede ubicar como una combinación de: a) un nivel de funciones F – ej. **OPERATE-F**—constituido por las acciones que transforman las entradas en salidas, y b) uno o dos niveles-de-estructura S -ej. **PROCES-S-**, de los dos asociados al nivel funcional elegido. De los nivéleles-de-estructura S, se obtienen las entradas y hacia estos se envían las salidas.

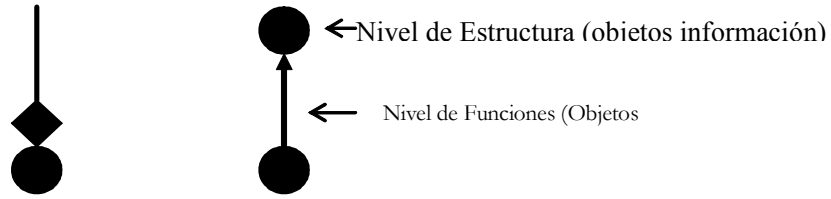


Figura 9-4 Componentes mínimos

Podemos identificar dos tipos de componentes mínimos: *unarios*, en los que la entrada y la salida se llevan acabo en un nivel-de-estructura único; y *binarios, en los que* la entrada la toma desde un nivel-de-estructura y la salida la envían a otro nivel-de-estructura.

En las secciones siguientes se describen las tres dimensiones del modelo CWSLR.

9.4STORAGE-D: Dimensión de Almacenamiento

La dimensión **STORAGE-D** consta de dos niveles de funciones F: **OPERATE-F** y **DIALOGUE-F**; y tres niveles-de-estructura S: **BASIS-S**, **PROCESS-S** y **MEMORY-S** (figura 9-5). Los niveles de funciones F transforman información de entradas a salidas, ambas en niveles-de-estructura. Añadiendo elementos para estructurar (forma[37], razones[38]y reglas). Por ejemplo, el nivel-de-estructura **PROCESS-S** es la salida de algunas funciones del nivel de funciones **OPERATE-F** y además la entrada para algunas funciones del nivel de funciones **DIALOG-F**.

[37] **Forma** es el conjunto de propiedades y relaciones que restringen tanto la elección como la ubicación de los elementos.
[38] **Razones** son las motivaciones para las elecciones hechas cuando se define una instancia de arquitectura.

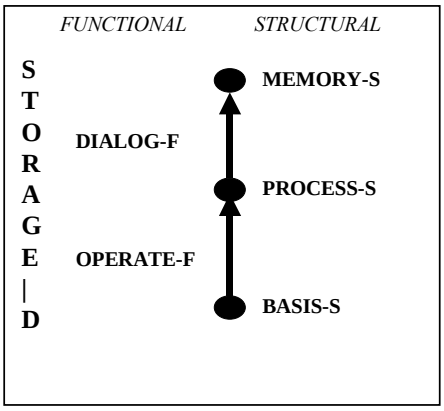


Figura 9-5 Dimensión STORAGE-D del modelo CWSLR

STORAGE-D esta constituida por un conjunto ordenado de niveles:

$$\text{STORAGE-D} = (\text{BASIS-S, OPERATE-F, PROCESS-S, DIALOG-F, MEMORY-S}) \quad (8)$$

Donde:

- **BASIS-S** (Almacenamiento Básico – nivel-de-estructura / Basic Storage – Structure level): es una estructura constituida por un conjunto de objetos-información clasificados en internos o en externos a la organización [MS98]. También en este nivel existen objetos-software u objetos-estante clasificados en cuatro clases: estratégicos, tácticos, operacionales [FSD82] [T88], y de argumentación [DB02] [KP01] [NWC97] [TRS93].
  Donde:
  - Estratégicos.- usados por la administración superior de la empresa para determinar: los objetivos a largo plazo, los recursos necesarios y las políticas establecidas que gobiernan la adquisición y uso de recursos en la organización, con plazo de dos años en adelante.
  - Tácticos.- objetos usados por la administración media de la empresa para la asignación y control del uso de los recursos de la organización. Su plazo varía entre un mes y dos años.
  - Operacionales.- creados por la administración operativa para soportar la toma de decisiones a corto plazo.
  - Argumentación.- objetos usados en cualquier nivel para capturar la memoria colectiva del grupo o de organización incluyendo: asuntos, posiciones, argumentos, discusiones, decisiones y justificaciones como elementos base.
- **OPERATE-F** (Soporte a Operaciones – nivel Funcional / Operations Support – Function level): Nivel funcional que consta de un conjunto de objetos-software u objetos-estante

de soporte para grupos [I95] [MS02], administración de documentos electrónicos, procedimientos de producción, y procesos ad hoc [T88], que transforman los objetos de **BASIS-S** a objetos más estructurados añadiendo contenido semántico al nivel **PROCESS-S** (ej. procedimiento interactivo para registrar a un alumno en una materia, donde los datos del alumno y de la materia se consideran en **BASIS-S** y los datos del registro del alumno se incluyen en **PROCESS-S**).

- **PROCESS-S** (Estructuras de Procesos – nivel-de-estructura / Processes Structures – Structure level): Estructura que consta de un conjunto de objetos-información, orientada a formar procedimientos o a organizar resultados [D00] [D03] [P84]. En otras palabras, en este nivel están los objetos-información necesarios para formar estructuras de resultados-en-vértice y de actividades-en-vértice (ej. el proceso de registrar un alumno en una materia).
- **DIALOGUE-F** (Soporte al Diálogo – nivel Funcional / Dialogue Support – Function level): Nivel funcional que consta de un conjunto de objetos-software u objetos-estante de servicios y de herramientas cuya función es transformar objetos de **PROCESS-S**, a objetos de **MEMORY-S** por medio de un diálogo. Este dialogo para guardar la historia de un proceso, cuenta con una base de conocimientos del usuario, un lenguaje de acciones y otro de presentación [B77].
- **MEMORY-S** (Memoria Organizacional – nivel-de-estructura / Organization Memory – Structure level): Estructura formada por un conjunto de objetos-información que añaden semántica a la ya dada por el nivel **PROCESS-S**. Almacena en forma persistente la información de la comunicación, que habilita la interacción entre las personas para: tomar decisiones, resolver problemas y coordinar los procesos [A94] [AH99] [CML94]. Se almacenan las decisiones tomadas y la información que las justifica.

Las funciones de **OPERATE-F** apoyan al usuario en la definición del problema durante su resolución y en la interacción con el sistema. Estas funciones usan métodos para transformar el conocimiento localizado en **BASIS-S,** añadiendo a **PROCESS-S** elementos que estructuren dicho conocimiento.

Un situación similar se tiene con **DIALOG-F** para transformar conocimiento almacenado en **PROCESS-S** añadiendo elementos al conocimiento y almacenando en **MEMORY-S**.

La memoria organizacional es muy importante para cualquier empresa. Comprende desde la bitácora de las operaciones hasta la documentación del proceso, a través de la representación estadística del desempeño de la compañía.

A continuación se incluye el ejemplo de un componente software y la ubicación de sus subcomponentes en la dimensión **STORAGE-D**:

Supongamos que se trata de un componente que forma parte de un ambiente para desarrollar software.

1. En el nivel-de-estructura **BASIS-S** se encuentran varios tipos de objetos, como son: 1) productos del trabajo (ej. Diagrama–de-Entidades-a-nivel-Usuario, Diagrama-de-Entidades-Combinado–a-Nivel-Usuario); 2) entradas (ej. Forma ITINERARIO), y 3) opciones (ej. Desarrollo de la Arquitectura de Software).

2. Esta información puede ser interna o externa a la organización y también organizada en documento o registro.
3. En el nivel-de-estructura **PROCESS-S** se localizan las dependencias y relaciones entre: 1) resultados (ej. Diagrama-de-Entidades-Combinado–a-Nivel-Usuario depende de haber hecho previamente el resultado Diagrama–de-Entidades-a-nivel-Usuario), y 2) opciones (ej. "Desarrollo de la Arquitectura del Componente" después de la opción "Requerimientos de Recursos"). Estas relaciones se pueden representar mediante grafos, restringiendo tanto la elección como la ubicación de los elementos.
4. En el nivel-de-estructura **MEMORY-S** se ubican los resultados de actividades de revisión en los que se documentan lasdecisiones tomadas (ej. Reporte de Revisión-Comentarios-e-Inquietudes y Reporte-de-Revisión-Detallado). También se incluyen resultados obtenidos en otros procesos en los que se llevan a cabo una toma de decisiones (ej. Solicitud-de-Cambio-en-Configuración).

9.5RESOURCE-D: Dimensión de recursos

La dimensión de recursos **RESOURCE-D** da estructura a las ocurrencias físicas de personas participantes, hardware, software y otras entidades pertenecientes a la organización (ej. la computadora ubicada en la oficina del jefe de Recursos Humanos). **RESOURCE-D** está constituida por cinco niveles de organización de los recursos y servicios, ubicándolos en la red que soporta las relaciones resultantes.

RESOURCE-D está formada por dos niveles de funciones F: **ORGANIZE-F** y **SERVE-F** como se muestra en la figura C96. Ambas funcionalidades transforman entradas a salidas. Se añaden elementos para estructurar (forma, razones y reglas). Por ejemplo **NETWORK-S** es la salida de **ORGANIZE-F** y la entrada para **SERVE-F**.

La dimensión de recursos incluye las bases conceptuales originalmente planteadas por Einar Stefferud [SFD82] para sistemas distribuidos. En el marco conceptual: Usuario Individual, Múltiples Usuarios, Instalación Remota / Single User, Multiple Users, Remote Utility (SUMURU).

El marco SUMURU [SFD82] incluye cuatro conjuntos de elementos: 1) procesadores (ubicados en el nivel **RESOURCE-S**); 2) redes (ubicadas en el nivel **NETWORK-S**); 3) servicios (ubicados en el nivel **SERVE-F**) y 4) normas (ubicadas en el nivel **STANDARD-S**).

Para estructurar esta dimensión se añade a los elementos del SUMURU, un quinto conjunto para las funciones utilizadas para *organizar* a los recursos en redes (conjunto ubicado en el nivel **ORGANIZE-F**). Así mismo, se renombra el conjunto de procesadores como **RESOURCE-S** para incluir en forma más general a cualquier elemento que se requiera. Cada uno de estos conjuntos representa un nivel de abstracción en la figura 9-6.

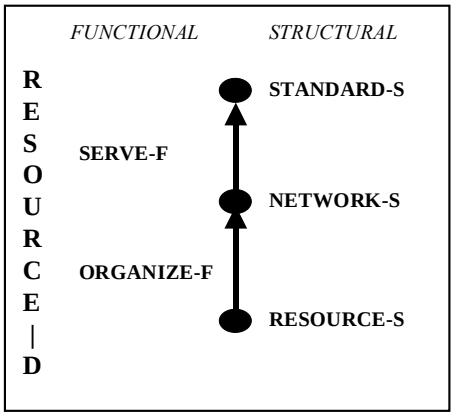


Figura 9-6 Dimensión RESOURCE-D del modelo CWSLR

Esta dimensión consta de un conjunto ordenado de niveles:

$$\text{RESOURCE-D} = (\text{RESOURCE-S, ORGANIZE-F, NETWORK-S, SERVE-F, STANDARD-S}) \quad (9)$$

Donde:

- **RESOURCE-S** (Recursos – nivel-de-estructura / Resources – Structure level), incluye los elementos necesarios para modelar los recursos de una organización. En este nivel se encuentran personas participantes, procesadores, lugares, herramientas y eventos.

  Hay tres tipos de procesadores. Usualmente asociados al almacenamiento de la información:

  1. Sistema para un usuario / Single-User System (SUS), este tipo de procesador puede operar en modo independiente / stand-alone pero también estarán conectados a redes locales / Local Networks (LNs). Los SUS son los clientes en las aplicaciones Cliente-Servidor.
  2. Sistema Multi-usuario / Múltiple-User System (MUS), da servicio a grupos locales de usuarios. Los servidores MUS respaldan a otros MUS, proveen cómputo pesado para SUS, además de bibliotecas para si mismos y para los SUS, y administran base de datos centralizadas.
  3. Sistemas Utilitario Remoto / Remote Utility System (RUS), provee cómputo pesado, administración de bases de datos corporativas y respaldo para los MUS. Para la mayoría de las organizaciones los RUS son los sistemas principales proporcionando servicios de red comercial de valor agregado (ej. espacio para publicar páginas Web).

- ***ORGANIZE-F*** (Organiza – nivel Función / Organize – Function level), Organiza los elementos principales formando redes. Incluye: a) el proceso requerido para analizar los requisitos y las aplicaciones de la organización para distribuir sus elementos asignándolos a los recursos de las redes, y b) la síntesis de las topologías viables para los niveles de desempeño deseados.
- ***NETWORK-S*** (Redes – nivel-de-estructura / Networks – Structure level), Hay dos tipos de redes:

  1. Las redes locales / Local Networks (LN), Transfieren gran volumen de información, así como de fuerte acoplamiento entre varios SUS y un MUS local. MUS puede proveer los archivos personales, archivos compartidos, y bibliotecas de programas para los SUS. También los MUS pueden ser las compuertas / gateways entre las redes locales LN y las redes remotas / Remote Networks (RN).
  2. Las redes remotas / Remote Networks (RN), proveen conexión entre MUS y conexión a RUSs internas y comerciales. Normalmente la velocidad de transferencia de las RNs es menor que, de las LNs.

- ***SERVE-F*** (Servicios – nivel Funcional / Services – Function level), hay cinco tipos principales de servicios:

  1. Servicios tipo TCP/IP, que proporcionan acceso a terminales desde una ligada a un MUS o a un SUS (la terminal debe ser capaz de acceder cualquier MUS, RUS, o SUS sujetándose sólo a restricciones de administración); proporcionan habilidad para transferir archivos, permitiendo que los usuarios envíen y reciban archivos completos; así como servicios de correo electrónico.
  2. Servicios para cooperación / Cooperative Services (CS) que puede incluir control dinámico del grupo, comunicación digital (ej. Audio), interfaces y administración de objetos comunes.
  3. Servicios de edición de texto, gráficos e imágenes.
  4. Servicios de administración de configuración.
  5. Servicios para generar interfases de usuario.

- ***STANDARD-S*** (Normas – nivel-de-estructura / Standards – Structure level), incluye normas corporativas en cinco áreas:

  1. Los valores meta, para evaluación del desempeño de los procesos.
  2. Sistemas operativos. Idealmente el sistema operativo seleccionado debería ejecutar en la mayoría de los equipos.
  3. Protocolos de comunicación para acceso a terminales, transferencia de archivos y servicio de correo electrónico.
  4. Sistemas de administración de base de datos (DBMS, ODBMS).
  5. Servicios para cooperación en procesos, soportando la coordinación, sincronización y flujo de trabajo (Workflow).

A continuación se presenta el ejemplo de un componente, que forma parte de un ambiente para desarrollar software, y la ubicación de sus elementos en la dimensión **RESOURCE-D**:

1. En el nivel-de- estructura **RESOURCE-S** se pueden encontrar: 1) personas (ej. Francisco López); 2) roles (ej. Líder del Equipo, Miembro del Equipo,...); 3) procesadores (ej. PC-1 del laboratorio de desarrollo); 4) equipo de comunicaciones (ej. Switch-3,); 5) canales de comunicación (ej. FO-mecánica-electrónica); 6) equipo periférico (ej. Impresora-1-RH); 7) espacios físicos (ej. Cubículo E-6); 8) edificios (ej. Edif-Electrónica E), y 9) campus (ej. Campus Mérida).

   También se almacena la ubicación de componentes de software en los equipos (ej. computadoras, equipo periférico ...): 1) de funciones (ej. opción "Búsqueda", ...), 2) resultados y entradas (almacenados en el nivel-de-estructura BASIS-S de la STORAGE-D),.
2. En el nivel-de-estructura NETWORK-S se encuentran las dependencias y relaciones entre procesadores, canales de comunicación y equipo de comunicaciones (ej. PC21, PC27 y PC33 conectadas a canal-2, y este a Concentrador–1, ...).

   Pudiendo representar los objetos y relaciones por medio de un grafo. Restringiendo tanto la elección como la ubicación de los elementos.
3. En el nivel-de-estructura STANDARD-S se almacenan: 1) los valores meta estadísticos, a alcanzar (ej. número de defectos menor a uno por cada 10,000 líneas de código, ...); Estos valores se utilizan cuando se evalúan tanto los procesos de producción como el proceso de toma de decisiones; y 2) los datos de métricas obtenidas en los procesos reales (ej. promedio de defectos del equipo: uno por cada 1000 líneas).

   Se incluyen también los protocolos normalizados a utilizar para operar las redes: (ej. TSP/IP, ...).

9.6LOGIC-D: Dimensión de lógica

La dimensión de lógica **LOGIC-D,** organiza las relaciones entre las tres dimensiones: **STORAGE-D**, **RESOURCE-D** y **LOGIC-D**. A su vez **LOGIC-D,** está compuesta de varios niveles lógicos, su nivel de abstracción se incrementa de abajo hacia arriba.

Esta dimensión contiene reglas, componentes, conectores, combinaciones e información de estado. Los niveles lógicos, de LOGIC-D, formalizan el: ambiente, áreas de competencia, casos de uso, procesos internos y el software de sistemas distribuido en ubicaciones específicas de la empresa.

LOGIC-D consiste de dos niveles de funciones **ALLOCATE-F** y **COORDINATE-F** como se muestra en la figura 9-7. Las funciones transforman la entrada a salida añadiendo elementos de estructura (forma, razones y reglas). El nivel-de-estructura **ORGANIZATION-S** es la salida del nivel de funciones **ALLOCATE-F** y es la entrada del nivel de funciones **COORDINATE-F**.

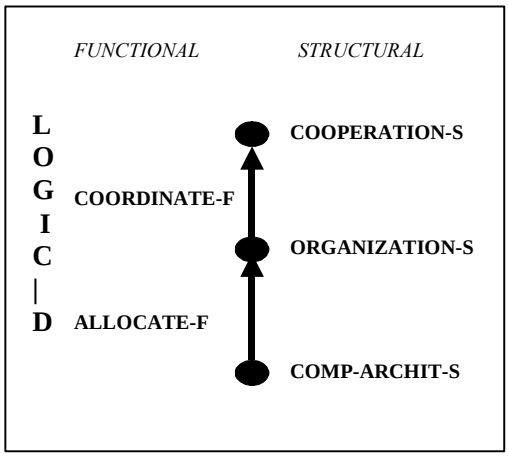


Figura 9-7 Dimensión LOGIC-D del modelo CWSLR

Esta dimensión está formada por un conjunto ordenado de niveles:

$$\text{LOGIC-D} = (\text{COMP-ARCHIT-S, ALLOCATE-F, ORGANIZATION-S, COORDINATE-F, } \mathbf{\text{COOPERATION-S}}) \quad (10)$$

Donde:

- **COMP--ARCHIT-S** (Arquitectura del Componente – nivel-de-estructura / Component Architecture – Structure level): es la estructura completa del componente o aplicación, y representa una arquitectura de software [A97] [DGD99] [NDD01], conteniendo instancias de componentes y conectores.

  Este nivel provee un modelo con las propiedades estructurales generales de los sistemas. Este tipo de propiedades incluyen la descomposición en partes y la interacción entre éstas, así como aspectos globales del sistema, como la sincronización y el desempeño.

  La interacción entre componentes necesita una definición precisa de las interfases de los componentes, así como del orden en el cual se llevan a cabo las acciones. Patrones de estructura con interacción parecida resultan en sistemas similares.
- **ALLOCATE-F** (Ubicar – nivel Funcional / Allocate – Function level), es un nivel de funciones que permite ubicar componentes de software en los elementos de la organización que los utilizarán. Se colocan los componentes en relación con las personas, los roles, los cubículos de los edificios respectivos, así como en áreas funcionales o áreas propietarias de recursos.
- **ORGANIZATION-S** (Organización – nivel-de-estructura / Organization – Structure level), es un nivel-de-estructura para modelos que representan a la organización, definiendo el universo de elementos, conectores y combinaciones de elementos. Este nivel mues-

tra las características requeridas por el estilo de cooperación estipulado, en el nivel-de-estructura **COOPERATION-S**.

- **COORDINATE-F** (Coordinación – nivel Funcional / Coordination – Function level), en este nivel de funciones deben existir varios protocolos, reglas que dirigen la interacción y cambios de estado de la organización y reglas para guiar a los integrantes de la organización, en su trabajo en grupos soportados por herramientas de cómputo [CDR96] [DG98] [DGV99] [FGR03] [MC90] [MC94] [B96].

  Este nivel determina el ambiente de la organización y como actúa la compañía en relación al ambiente [CSM03] [JEJ95]; Realiza lo que se debe hacer, y cuando se debe hacer por los empleados en todos los niveles de la estructura estática de la organización. Es importante que el modelo incluya la dinámica de la organización.
- **COOPERATION-S** (Cooperación – nivel-de-estructura / Cooperation –Structure level), en este nivel-de-estructura se almacena el resultado del comportamiento cooperativo de los grupos de trabajo (ej. comentarios técnicos y asuntos pendientes), así como los resultados de la toma de decisiones [GHBKB03] [CGK01] [CLN00] [V95]. Este nivel incluye las reglas de la organización, que estructuran y restringen la composición e interacción entre componentes.

La estructura de esta dimensión incluye en **COMP--ARCHIT-S** una jerarquía de conceptos: el estilo arquitectónico, la arquitectura y la configuración [A97] [AG97] [ADG98] [PW92] [P01] [SW01] [GS93] [CGLNS03] [SG95].

Los modelos de organización incluidos en **ORGANIZATION-S** deben concordar con los elementos que intervienen en las reglas almacenadas en **COOPERATION-S,** para la resolución de conflictos. Del mismo modo en el cual dichos modelos organizacionales restringen el universo de componentes, conectores, interacciones, eventos y estados que pueden ser parte de cualquier arquitectura incluida en **COMP-ARCHI-S**.

En esta dimensión lógica, los elementos del nivel más bajo heredan los atributos (reglas, estructura, y componentes) del nivel superior.

En seguida damos un ejemplo de la estructura de un fragmento representativo de la dimensión **LOGIC-D**:

Supongamos que se trata de un componente que forma parte de un ambiente para desarrollar software.

1. En el nivel-de-estructura **COMP-ARCHI-S** se agrupan de acuerdo a un orden adecuado los modelos que se incluyen en las otras dos dimensiones (ej. 1 - Modelos de componentes con relación: llamado-por, es un, necesita a, ... , ej. 2 - Ubicación de los elementos del software en la topología de la red de computadoras …).
2. En el nivel-de-estructura **ORGANIZATION-S** se encuentran las dependencias y relaciones entre los elementos que integran a la organización (ej. 1 desarrolladores de software en un grupo coordinado por un Propietario de Recursos: “Desarrolladores”; ej. 2 los *Project* Leaders contratan desarrolladores solicitándolos a los Propietario de Recursos, …). El contenido de este nivel-de-estructura permite la representación en un grafo, restringiendo la elección y la ubicación de los elementos (forma).

3. En el nivel-de-estructura **COOPERATE-S** se encuentran las reglas que gobiernan el comportamiento de la organización cuando hay conflictos (ej. Se rechaza para modificación.- cuando un miembro del grupo considere que existe un defecto técnico; la programación de actividades con duración de 5 a 10 Hrs., ...).

El modelo CWSLR proporciona una base estructural, para el conjunto de productos, que guía las tareas a realizar en un ambiente de desarrollo. Así mismo, el estilo arquitectónico de la Arquitectura Dirigida por Modelos (MDA), complementa al modelo de referencia CWSLR. CWSLR y MDA constituyen una base para el método AGD, especialmente para obtener la arquitectura del componente de software, como se explica a continuación.

9.7Arquitectura del Componente

La estructura o estructuras del sistema, que comprende los componentes de software, las propiedades visibles externamente de esos componentes y las relaciones entre ellos, constituyen la definición de arquitectura de software de un componente.

La importancia de este concepto, en el desarrollo de software, esta dada por varios hechos:

- La arquitectura de software es un resultado principal del conjunto de productos utilizados en el desarrollo de software. Pues proporciona información para planificar los recursos y el esfuerzo necesario para el desarrollo del componente en el contexto tecnológico y de acuerdo con las características de la empresa que lo desarrollará.
- En el proceso de desarrollo de software el primer resultado donde se evalúa si el producto diseñado posee los atributos[39] de calidad planteados por el cliente y por la empresa desarrolladora, es la arquitectura del software.

Como se muestra en la figura 9-8, además de contar con un modelo de referencia, se requiere establecer un estilo arquitectónico para tener un punto de partida en el proceso para obtener una arquitectura de software.

El Estilo de Arquitectura se define como una descripción de tipos de componentes permitidos y un patrón de transferencias de control y/o datos, llevado a cabo durante la ejecución. Un estilo es un conjunto de restricciones sobre una arquitectura –limitaciones en los tipos de componentes y sus patrones de interacción-- que definen un conjunto o familia de arquitecturas que las satisfacen.

[39] Atributo de calidad, es una característica que afecta la calidad de un elemento. En una jerarquía de atributos de calidad, los atributos de mayor nivel se pueden llamar factores de calidad, los atributos de menor nivel se pueden llamar atributos de calidad [IEEE-610.12]. En el caso de la norma ISO/IEC 9126 -1 se mencionan características y sub-características [ISO-9126].

Para obtener la arquitectura de software comenzamos por definir un *Modelo de referencia. Se requiere para* mostrar una división de la funcionalidad junto con el flujo de datos entre las piezas funcionales. En nuestro caso utilizamos el modelo CWSLR (explicado en las §s de la 9.2 hasta la 9.7).

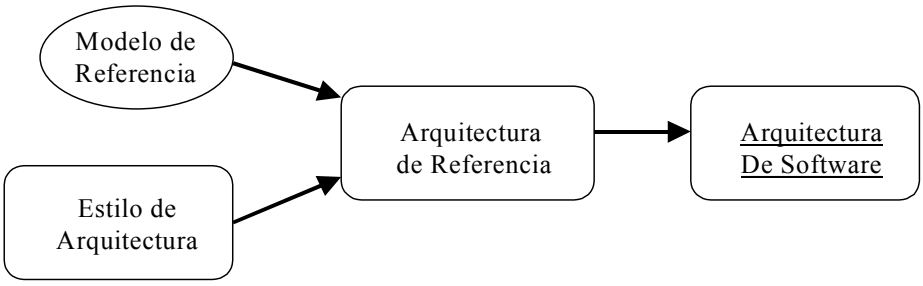


Figura 9-8 Elementos en el desarrollo de una arquitectura de Software

El estilo de arquitectura básico que se utiliza en el ambiente de desarrollo CWSP es la Arquitectura Dirigida por Modelos / Model Driven Architecture (MDA). Donde un modelo representa un aspecto particular empleando un sistema en construcción, operación o mantenimiento. Un modelo está escrito en el lenguaje de un meta-modelo específico.

El meta-modelo representa la especificación de una abstracción, basada en acuerdos tomados. Un meta-modelo, actúa como filtro para extraer algunos aspectos relevantes de un sistema e ignorar detalles. Un meta-meta-modelo define un lenguaje para escribir meta-modelos.

Usar la arquitectura dirigida por modelos MDA significa manejar modelos para interpretar la comprensión, diseño, construcción, desarrollo, operación, mantenimiento y modificación de la arquitectura del sistema de software resultante.

Dar importancia a la MDA se justifica pues la tecnología de modelos incluye a la tecnología de objetos, ofreciendo una trayectoria de migración de las soluciones actuales de objetos y componentes hacia organizaciones más ambiciosas [S05]. El costo creciente de la integración, durante el desarrollo de software y los efectos de los errores en el software, motivan la tendencia a la generación automática de código a partir de descripciones formalizadas basadas en modelos.

La *Arquitectura de Referencia,* se define como un modelo de referencia mapeado a componentes de software que proporcionan las funciones del modelo de referencia y al flujo de datos generado entre los componentes (figura 9-6).

El proceso planteado en la figura 9-8, para llegar a una arquitectura de software, muestra que se requiere tener un estilo arquitectónico, probablemente compuesto, que en nuestro caso es inicialmente dirigido a modelos, complementado con algún otro estilo. Esta mezcla de estilos se determina principalmente en función de los aspectos de calidad de software (que satisfacen los requerimientos del cliente), requeridos.

En la figura 9-9 incluimos un catálogo de estilos de arquitectura propuesto por Mary Shaw y David Garlan [GS93], obtenido mediante el estudio de las arquitecturas más usadas. Formando

una jerarquía que incluye en su primer nivel los estilos de arquitectura siguientes: Componentes Independientes, Flujo de Datos, Centrado en Datos, Máquina Virtual y Call y return.

Para la determinación del estilo de arquitectura se cuenta con una primera aproximación de los atributos de calidad de software asociados a los principales estilos arquitectónicos, esta relación de atributos con estilos se muestra en la tabla 9-1. Los atributos de calidad, que se considera están relacionados con la arquitectura [BCK98], son: 1) Desempeño, 2) Seguridad, 3) Disponibilidad, 4) Usabilidad, 5) Modificabilidad, 6) Portabilidad, 7) Reusabilidad, 8) Integrabilidad, 9) Pruebabilidad.

La relación entre estilos arquitectónicos y los atributos de calidad ayuda a tomar decisiones en la elección de la arquitectura de software. Durante el desarrollo del PS3Set **General-Design: Component Architecture** se dirige el trabajo hacia los estilos arquitectónicos, teniendo en cuenta la obtención de los atributos de calidad deseados.

Un ejemplo del uso de la tabla 9-1 es en el caso de que se requieran los atributos de calidad Reusable y Modificable. Buscándolos en la segunda columna (atributos de calidad) y determinando cual es el estilo arquitectónico que se los fomenta. En este caso el estilo asociado es el Call-Return.

En el trabajo de diseño necesario para obtener arquitectura de software se requiere identificar y proponer sus componentes.

Tabla 9-1 Estilos de Arquitectura y los Atributos de Calidad que promueven

Estilo de Arquitectura		Atributos de Calidad
	Sub-estilo	
Centradas en Datos		Integrable en datos
Flujo de Datos		Reusable y Modificable
Máquina Virtual		Portable
Arquitecturas Call-Return		Modificable (escalable)
	Programa Principal y Subrutina	Modificable
	Llamado a Procedimiento Remoto	Desempeño
	Objetos (Tipo Abstracto d Datos)	Reusable y Modificable
	En Capas	Modificable, Portable, Parametrizable
Componentes Independientes		Modificable (desacoplando)
	Procesos comunicantes.	Modificable (escalable)

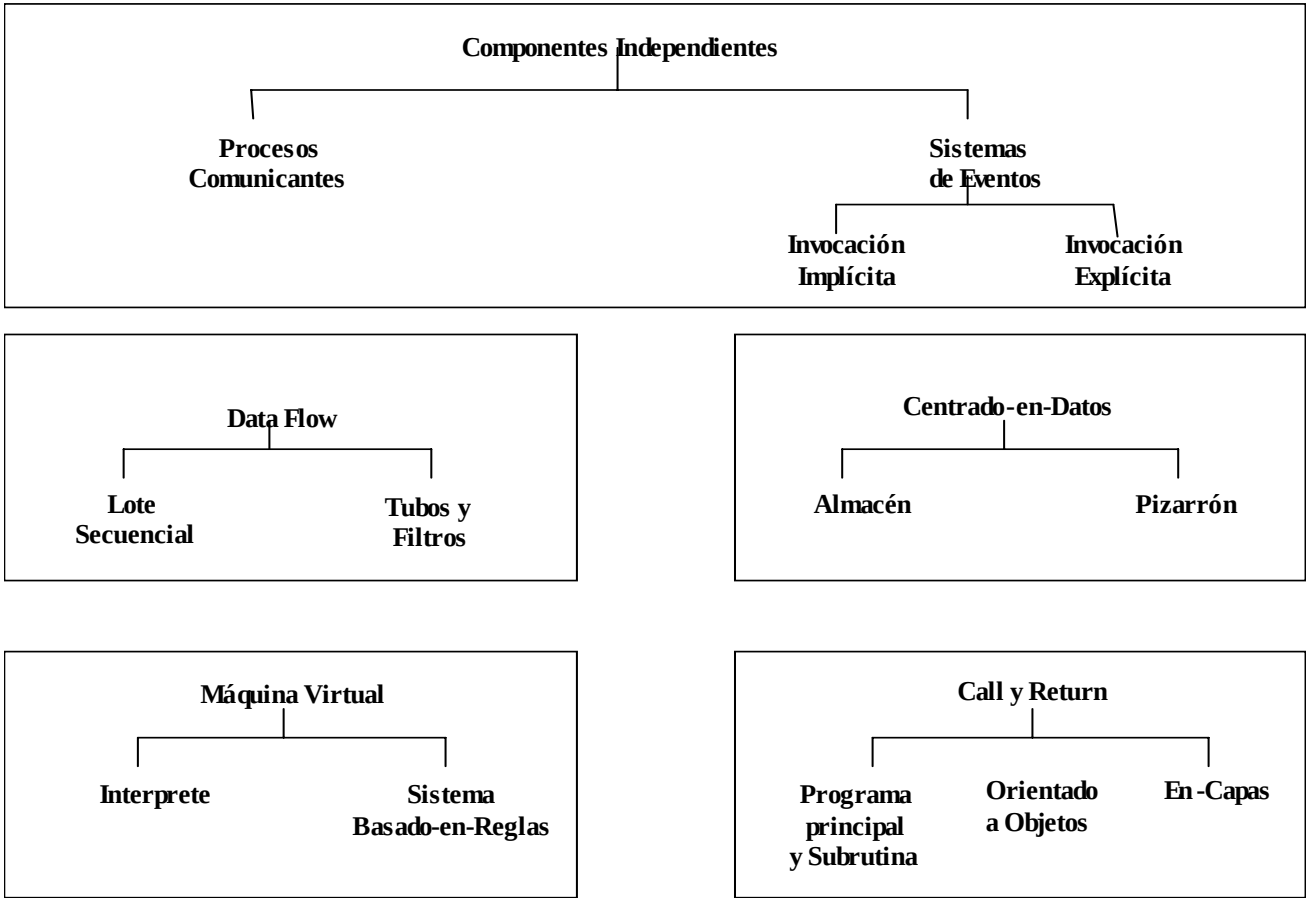


Figura 9-9 Estílos Arquitectónicos, con relaciones “es-un”

Cuando se trabaja con la arquitectura a nivel de descripciones de objetos y clases, se recomienda considerar a los patrones de diseño, que son descripciones de objetos y clases comunicantes que se construyen según especificaciones, para resolver un problema de diseño general en un contexto particular [GHJ94 pp. 3].

Tabla 9-2 Clasificación de Patrones de Diseño [GHJ94]

		Propósito		
		Creación	Estructura	Comportamiento
Alcance	Clase	Método de Fabrica / Factory Method	Adaptador (clase) / Adapter (class)	Interprete / Interpreter Método Plantilla / Template Method
	Objeto	Fabrica Abstracta / Abstract Factory Constructor / Builder Prototipo / Prototype Único / Singleton	Adaptador (objeto) / Adapter (object) Puente / Bridge Compuesto / Composite Decorador / Decorator Fachada / Facade Peso Mosca / Flyweight Apoderado / Proxy	Cadena de Responsabilidad / Chain of Responsability Instrucción / Command Iterador / Iterator Mediador / Mediator Recuerdo / Memento Observador / Observer Estado / StateEstrategia / Strategy Visitante / Visitor

Los patrones de diseño varían en su granularidad y nivel de diseño. Los patrones de diseño se clasifican de tal manera que podemos referirnos a familias de patrones relacionados, como se muestra en la figura 9-2. La clasificación resultante ayuda a aprender los patrones más rápidamente y a encontrar nuevos patrones.

La clasificación que se incluye se realizó con dos criterios: 1) **Propósito**, se refiere a la función que realiza el patrón (***creación*** – concerniente al proceso de creación de objetos, ***estructura*** – relacionado a la composición de clases y objetos y comportamiento – caracteriza las formas en que interactúan y se distribuyen la responsabilidad, las clases u objetos; 2) **Alcance**, especifica si el patrón se aplica principalmente a ***clases*** (tratan con relaciones entre clases y sus subclase) o a ***objetos*** (tratan con relaciones entre objetos, que se pueden cambiar en tiempo de ejecución y son más dinámicas).

En el diseño de la arquitectura de software se requiere identificar y proponer sus componentes. En la sección siguiente se describe un catálogo que representa una taxonomía útil durante la realización de dicha actividad.

9.8Catálogo de Tipos de Componentes Mínimos

Los componentes mínimos (definidos en § 9.3) se agrupan mediante el reconocimiento de patrones, de características comunes, en tipos de componentes mínimos. Esta generalización ayuda a la clasificación de los componentes y puede facilitar su almacenamiento y recuperación. Analizando el modelo CWSLR se obtuvo el Catálogo de Tipos de Componentes Mínimos (MCTC: Minimal Component Type Catalog), que se puede usar para organizar su almacenamiento, soportando el re-uso de los modelos producidos.

MCTC ayuda a estructurar el almacenamiento de todos los resultados obtenidos en la construcción del componente, organizándolos en una Librería de Componentes Re-usables / Reusable Component Library (RCL). Si los componentes se guardan con una estructura predefinida en la RCL, se facilita su reutilización.

Se construyó un catálogo a partir del modelo CWSLR considerando sus seis niveles funcionales: S1, S2, R1, R2, L1 y L2, en la figura 9-10. Los niveles funcionales se convierten en las ramas de primer nivel, de un catálogo organizado como un arbol invertido de cuatro niveles de abstracción, con mayor detalle en el cuarto nivel.

Se refinó cada una de las seis ramas del primer nivel del MCTC, y ayudándose de los elementos contenidos en cada uno de los niveles estructurales del modelo CWSLR, se obtuvo un segundo nivel que contiene 23 elementos. A su vez se refinaron los 23 elementos y se obtuvieron 38 elementos para el tercer nivel. Por último se refinaron los 38 elementos para obtener 64 elementos, en el cuarto nivel.

El catálogo se usa como versión inicial (o semilla) en el momento en que se comience a utilizar. Conforme se desarrollen componentes de software, el catálogo se actualiza dando de alta nuevos tipos de componentes mínimos o eliminando tipos no usados o cambiando la información de algún tipo de componente existente.

9.9Conclusión

El modelo Trabajo Cooperativo: Almacenamiento, Lógica y Recursos / Cooperative Work Storage Logic and Resources (CWSLR) establece una referencia abstracta para organizar los niveles del ambiente de desarrollo CWSP.

El modelo CWSLR agrega, además de los elementos incluidos en otros modelos, áreas de interés incluidas en el estudio del trabajo cooperativo y en las aplicaciones computacionales contemporáneas, tales como:

- (PROCESS-S) - Organización en procesos: entradas, funciones y salidas
- (MEMORY-S) - Memoria organizacional o de grupo: acerca del desempeño y decisiones dentro del proceso
- (DIALOG-F) - Dialogo: acerca del proceso actual y memoria organizacional

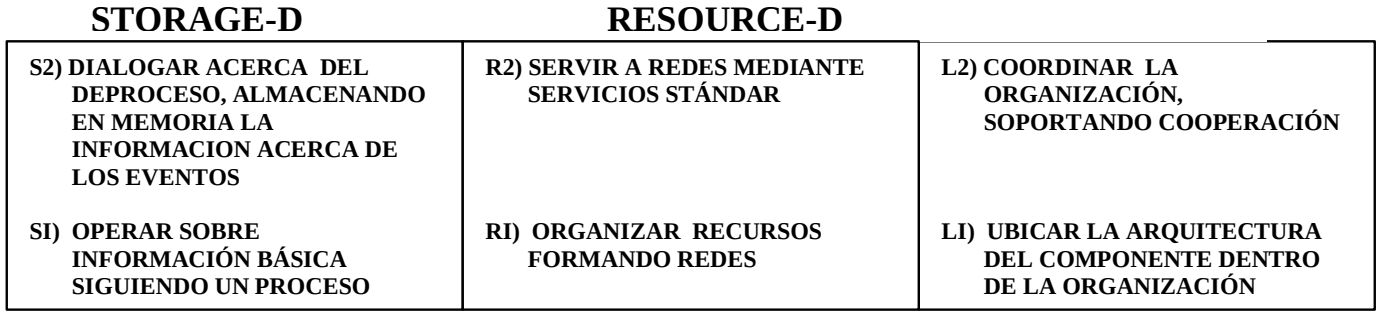


Figura 9-10 Estructura del primer nivel del Catalogo de Tipos de Componentes Mínimos (MCTC)

- (STANDARD-S) - Normalización del desempeño deseado para la organización
- (COOPERATION-S) - Reglas y servicios para cooperar, resolviendo conflictos
- (COORDINATION-F) - Coordinación de la organización: para alentar el trabajo cooperativo

Estas áreas adicionales son 6, de los 15 niveles que constituyen al CWSLR.

Con el modelo CWSLR y el estilo arquitectónico basado en la Arquitectura Dirigida a Modelo / Model Driven Architecture (MDA), se obtiene una estructura de referencia para otros niveles del ambiente de desarrollo CWSP. Los elementos que se ven influenciados más por CWSLR son: a) la estructura de productos, explicada en el capítulo 4; y b) la estructura de procesos, descrita en el capítulo 5.

Este trabajo de investigación se enfoca en la definición de los productos o modelos a desarrollar para obtener una aplicación de software. La definición de los productos se dirige utilizando al CWSLR como modelo de referencia, lo que asegura que se contará con los elementos requeridos por un sistema de información que soporte a la cooperación

10Anexo C.- Contenido del CD-ROM

En el CD-ROM anexo se encuentra:

El archivo SiSoProS.zip - Que contiene el directorio SiSoProS empaquetado.

Procedimiento para utilizarlo:

1. Se requiere desempacar el archivo SiSoProS.zip en el directorio raíz del disco C:
2. C:\
3. Ya desempacado se requiere explorar el directorio "instrucciones de intalacion" y abrir con el Micosoft Word el archivo "Instrucciones de intalacion SiSoProS".
4. Seguir las instrucciones incluidas en el archivo "Instrucciones de intalacion SiSoProS"·.

   Para ejecutar el prototipo SiSoProS, se da un clic sobre Inicio (en la barra de Windows XP), se coloca el cursor del ratón sobre **Todos los programas** y se elige **prototipo SiSoProS**.

Al ejecutarse SiSoProS aparece la primera ventana: **Acceso al Sistema SiSoProS**. Esta ventana muestra dos casilleros donde se indicará el usuario: **admon** y el password: **admon**. En seguida de un clic sobre el botón aceptar y aparecerá la interfaz principal del SiSoProS.

Para navegar la interfaz principal del SiSoProS se sugiere acceder a la opción **Ayuda** que se encuentra en el menú **Configura**. Al dar clic sobre **Ayuda,** encontrará el **Manual de Usuario del SiSoProS**, donde encontrará una explicación breve de los menús y opciones que ofrece el SiSoProS.

Información complementaria al contenido de este documento de tesis.

- El detalle de todos los elementos, del catálogo de tipos de componentes mínimos, se puede consultar en el prototipo incluido en el CD-ROM del anexo B. Siguiendo la secuencia de opciones siguiente: Configura – Manejo de Productos del Trabajo – Despliegue / Impresión – PS1 – COMPONENT-CONTROL – COMP-KIND. Las opciones, en la lista anterior se separan por la secuencia " – ".
- Las formas de captura, pueden obtenerse del prototipo incluido en el CD anexo, siguiendo las opciones siguientes: **Docs-Soporte > Formas**, ver figura 10-1.

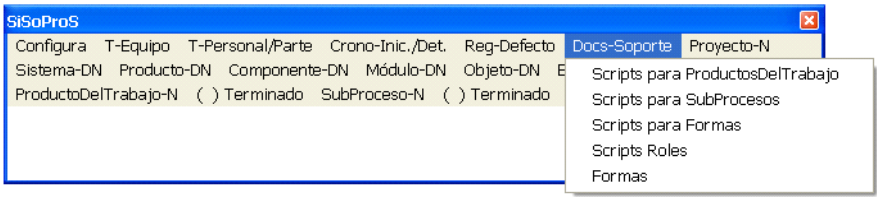


Figura 10-1 ***Ventana CPrincipal mostrando el menú Documentos de Soporte.***

Requerimientos

Para utilizar el prototipo se requiere una computadora personal, con el sistema operativos Windows XP de Microsoft, y un navegador Web.

REFERENCIAS

[A00] Robert Axelrod; On Six Advances in Cooperation Theory; Analyse & Kritik, 2000, pp. 130-151.

[A01] Scott W. Ambler; Agile Modeling and the Unified Process; Copyright 2001-2002. www.agilemodeling.com/essays/agileModelingRUP.htm

[A02] S.Ambler; Agile Modeling: Effective Practices for Extreme Programming and the Unified Process ; New York:Joh Wiley &Sons,I c.New York, 2002.

[A79] A. J. Albrecht, "Measuring Application Development Productivity", In Proc. of the IBM Application Development Symposium, GUIDE Int and Share Inc., pp 83-92, Monterrey, USA, October 14-17, 1979.

[A84] R. Axelrod; The Evolution of Cooperation. Basic Books, 1984

[A94] Ackerman M. S.; Definitional and Contextual Issues in Organizational and Group Memories; Proceedings of the Twenty-seventh IEEE Hawaii International Conference of System Sciences (HICSS 94), pp. 191-200.

[A97] Robert J. Allen; A Formal Approach to Software Architecture; Ph.D. thesis, School of Computer Science, Carnegie Mellon University, May 1997

[A97-b] R. Axelrod; The Complexity of Cooperation; Princeton University Press , 1997

[A98] G. Antoniol, F. Catzolari, L. Cristoforetti, R. Fiutem, G Caldiera, "Adapting Function Points to Object Oriented Information Systems", In Proc. Advanced Information Systems Engineering: CAISE'98, Lecture Notes in Computer Science, LNCS Vol. 1413, pp. 59-74, 1998.

[ADG98] Robert Allen, Rémi Douence, David Garlan: Specifying and Analyzing Dynamic Software Architectures. FASE (Fundamental Approaches to Software Engineering, , 1st Internationsl Conference) 1998: 21-37

[AFGP02] Silvia Mara Abrahão, Joan Fons, Magalí González, Oscar Pastor: Conceptual Modeling of Personalized Web Applications. AH (Adaptive Hypermedia and Adaptive Web-Based Systems) 2002: 358-362;, Malaga, Spain, May 29-31, 2002, Proceedings. Lecture Notes in Computer Science 2347 Springer 2002, ISBN 3-540-43737-1

[AG97] Robert Allen, David Garlan: A Formal Basis for Architectural Connection. TOSEM (ACM Transactions on Software Engineering and Methodology) 6(3): 213-249 (1997)

[AH99] Mark S. Ackerman, Christine Halverson: Organizational Memory: Processes, Boundary Objects, and Trajectories. HICSS (32nd Annual Hawaii International Conference on System Sciences) 1999 (IEEE Computer Society); Track 1: New Curriculum and Courses and Collaboration Systems and Technology. http: //computer.org/proceedings/hicss/0001/00011/0001toc.htm

[ASF04] João Paulo Almeida, Marten van Sinderen, Luís Ferreira Pires; The role of the RM-ODP Computational Viewpoint Concepts in the MDA approach; Workshop on ODP for Enterprise Computing (WODPEC 2004), In conjunction with IEEE EDOC 2004

[AWSR03] Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen*, J.;* New directions on agile methods: comparative analysis; Software Engineering, 2003. Proceedings. 25th International Conference on, Page(s): 244- 254

[B01] Beck, K.; Cockburn, A.; Jeffries, R.; and Highsmith, J.; Agile Manifesto; http://www.agilemanifesto.org, 2001.

[B02] Lahouria Bendoukha; Towards A Model for Understanding Cooperative Work in Developing Information Systems; Proceedings of the 7th Maghrebian Conference on Software Engineering and Artificial Intelligence (MCSEAI 02) at Annaba, Algeria, May, 6-8, 2002.

[B77] Bennett J.; User-Oriented Graphics, Systems for Support in Structured Tasks; in User Oriented Design of Interactive Graphics Systems; S. Treud editor, New York: Association for Computing Machinery, 1977.

[B85] B. Boehm; A Spiral Model of Software Development and Enhancement; Proc. Int'l Workshop Software Process and Software Environments, ACM Press, 1985.

[B91] Patrick G. Brown; QFD: Echoing the Voice of the Customer; AT&T Technical Journal, Vol. 70 No. 2, pp. 18-32, Mar/Apr 1991.

[B92] V. R. Basili, "Software Modeling and Measurement: The Goal Question Metric Paradigm," *Computer Science Technical Report Serie*s, CS-TR-2956 (UMIACS-TR-92-96), University of Maryland, College Park, MD, September 1992.

[B94] J. W. Barnes, "Statistical Analysis for Engineers and Scientists: A Computer-Based Approach", McGraw-Hill Inc., pp. 199-200 and 341-343, New-York, 1994.

[B96] John A. Boyd; Floor Control in Synchronous Groupware; PhD dissertation, Graduate School of the Ohio State Univ., 1996.

[BCG95] D. Batory, Lou Coglianese, M.Goodwin, and S. Shaver. Creating Reference Architectures: An Example from Avionics. *Symposium on Software Reusability (SSR)*, Seattle Washington, April 1995.

[BCK98] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice. Addison Wesley, 1998, ISBN 0-201-19930-0.

[BCP01] Clarke M. Blair G. Coulson G. Parlavantzas N. An efficient component model for the construction of adaptive middleware. IFIP / ACM International Conference on Distributed Systems Platforms (Middleware'2001), November 2001.

[BD01] Xavier Blanc, Raymonde Le Delliou: Information System architecture with RM-ODP: an on-the-field experience. WOODPECKER (International Workshop on Open Distribute Processing: Enterprise, Computation, Knowledge, Engineering and Realisation) 2001: 27-37; In conjunction with ICEIS 2001, Setúbal, Portugal, July 6, 2001. ICEIS Press 2001, ISBN 972-98050-5-9

[BGK06] Krisnakumar Balasubramanian, Aniruddha Gokhale, Gabor Karsia, Janos Sztipanovits, Sandeep Neema; Developing Applications Using Model-Driven Design Environments; IEEE Computer, February 2006, pp. 33 – 40.

[BH94] S. Bayer, J. Highsmith; RADical software development; American Programmer, vol. 7, pp. 35=42, 1994.

[BKOMGM85] Robert G. Babb II, Richard B. Kieburtz, Ken Orr, Ali Mili, Susan Gearhart, Nancy N. Martin: Workshop on Models and Languages for Software Specification and Design. IEEE Computer 18(3): 103-108 (1985)

[BP01] Barry Boehm, Daniel Port; Balancing Discipline and Flexibility With the Spiral Model and MBASE; CROSSTALK The Journal of Defense Software Engineering, Diciembre del 2001.

[BSW04] Terence J Blevins, John Spencer, Fred Waskiewicz; TOGAF ADM and MDA® The Power of Synergy; The Open Group white paper w052, May 2004; http://www.opengroup.org/bookstore/catalog/w052.htm

[BT03] B. Boehm, R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods", *IEEE Computer***,** IEEE Computer Society, June 2003 (Vol. 36, No. 6), pp. 57-66.

[C04] Mary Beth Chrissis, Mike Konrad, Sandy Shrum; CMMI Guidelines for Process Integration and Product Improvement, Addison-Wesley, 2004.

[C79] Crosby, P. B; Quality is Free; New York, New York: McGraw-Hill, 1979.

[C97] E. Comer, "Alternative Software Life Cycle Models," in *Software Engineering*, M. Dorfmann and R. Thayer (eds.), IEEE CS Press, 1997.

[CDR96] Chaudhury A., Deng P., Rathnam S.; A Computational Model of Coordination; IEEE Transactions on Systems, Man, and Cybernetics – part A: Systems and Humans, Vol. 26, No. 1, January 1996.

[CG89] Nicholas Carriero, David Gelernter: How to Write Parallel Programs: A Guide to the Perplexed. ACM Computing Surveys 21(3): 323-357 (1989)

[CG90] N. Carreiro and D. Gelernter; How to Write Parallel Programs: A First Course; MIT Press, Cambridge, MA, 1990.

[CGK01] Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David V. Pynadath, Thomas A. Russ, Milind Tambe; Electric Elves: Applying Agent Technology to Support Man Organizations; Proceedings Thirteenth Innovative Applications of Artificial Intelligence Conference on Artificial Intelligence (IAAI), 2001.

[CGLNS03] Paul C. Clements, David Garlan, Reed Little, Robert L. Nord, Judith A. Stafford: Documenting Software Architectures: Views and Beyond. ICSE 2003: 740-741; 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA. IEEE Computer Society 2003.

[CGM99] CARRASCO, R.; GALINDO, J.; MEDINA, J.M.; VILA, M. Amparo (1999). Clustering and Classification in a Finantial Data Mining Environnment. En: *Proceedings 3th International ICSC Symposium on Soft Computing SOCO'99*, pp. 713-72. Genova (Italy).

[CH03] Krzysztof Czarnecki and Simon Helsen ; Classification of Model Transformation Approaches; In 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, 2003.

[CLN00] Jan Chomicki, Jorge Lobo, Shamim A. Naqvi: Conflict Resolution Using Logic Programming. TKDE (IEEE Transactions on Knowledge and Data Engineering)15(1): 244-249 (2003).

[CML94] Chen H., McHenry W. K., Lynch K. J., Goodman S. E.; A Textual Database/Knowledge-Base Coupling Approach to Creating Computer-Supported Organizational Memory; International Journal of Man-Machine Studies, July 1994.

[CS97] M. A. Cusumano, R. W. Selby; How Microsoft build software; Communications of the ACM, vol 40, pp. 53-61, 1997.

[CSM03] Jaelson Castro, Carla T. L. L. Silva, John Mylopoulos; Modeling Organizational Architectural Styles in UML. CAiSE 2003: 111-126; Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003, Klagenfurt, Austria, June 16-18, 2003, Proceedings. Lecture Notes in Computer Science 2681 Springer 2003, ISBN 3-540-40442-2

[CSN05] Kai Chen, Janos Sztipanovits, Sandeep Neema: "Toward a Semantic Anchoring Infrastructure for Domain-Specific Modeling Languages," EMSOFT 2005, September, 2005 Newark, NJ

[D00] Wolfgang Deiters: Information Gathering and Process Modeling in a Petri Net Based Approach. Business Process Management 2000: 274-288; Techniques, and Empirical Studies. Lecture Notes in Computer Science 1806 Springer 2000, ISBN 3-540-67454-3

[D03] Jan L. G. Dietz: Generic Recurrent Patterns in Business Processes. BPM 2003: 200-215; Business Process Management, International Conference, , Eindhoven, The Netherlands, June 26-27, 2003, Proceedings. Lecture Notes in Computer Science 2678 Springer 2003, ISBN 3-540-40318-3

[D06] Software Process Dashboard Initiative; http://processdash.sourceforge.net/
[D86] Deming, W. Edward. Out of the Crisis. Cambridge, MA: MIT Center for Advanced Engineering, 1986.
[D88] Drucker P. F.; The Coming of the New Organization; Harvard Business Review, Harvard Busines School, Boston, MA 02163, January-February 1988, pp. 45-53
[DB02] José Maria N. David, Marcos R. S. Borges: COPSE-Web: An Infrastructure for Developing Web-Based Groupware Applications. CRIWG 2002: 275-284; Groupware: Design, Implementation and Use, 8th International Workshop, CRIWG 2002, La Serena, Chile, 1.-4. September 2002, Proceedings. Lecture Notes in Computer Science 2440 Springer 2002, ISBN 3-540-44112-3
[DBC88] A. Davis, E. Bersoff, and E. Comer; A Strategy for Comparing Alternative Software Development Life Cycle Models; *IEEE Transactions on Software Engineering*, vol. 14, no. 10, pp. 1453-1461, 1988.
[DG98] Deiters W., Gruhn V.; Process Management in Practice Applying the FUNSOFT Net Approach to Large-Scale Processes; Automated Software Engineering 5, 7-25 1998
[DGD99] Drira Khalil, Gouezec Frédéric, Diaz Michael; Design and Implementation of coordination protocols for distributed cooperating objects – A general graph-based technique applied to CORBA; www.khalil.laas.fr/~khalil/fmoods99.pdf
[DGV99] Drira K., Gradit P., Vernadat F.; Graph-based coordination of cooperative agents; 1999.
[DMH02] Noopur Davis, Jim McHale, Watts Humphrey; Relating the Team Software Process (TSP) to the Capability Maturity Model for Software (SW-CMM); CMU/SEI-2002-TR-008, SEI Joint Program Office, Carnegie Mellon Software Engineering Institute.
[DoD02] DoD 5000.2-R; Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Information Systems (MAIS) Acquisiotion Program; USA Department of Defense, April 5, 2002.
[DoD88] DoD-STD-2167A; Dod Standard: Defense Systems Software Development; USA Department of Defense, 29 February 1988.
[DoD94] MIL-STD-498; Military Standard: Software Development and Documentation; USA Department of Defense, 5 December, 1994.
[DW02] P. David Stotts, Laurie A. Williams: Distributed Pair Programming. XP/Agile Universe 2002: 283
[F94] Flint David; Application Delivery in the 1990s; IT Management Programme Report, Wentworth Research, Park House, Wick Road, Egham, England TW20 OHW, http://www.wentworth.co.uk, March 1994.
[FGR03] Martin Fredriksson, Rune Gustavsson, Alessandro Ricci: Sustainable Coordination**.** 203-233; Matthias Klusch, Sonia Bergamaschi, Peter Edwards, Paolo Petta (Eds.): Intelligent Information Agents - The AgentLink Perspective. Lecture Notes in Computer Science 2586 Springer 2003, ISBN 3-540-00759-8
[FSD82] E. Stefferud, D. Farber, R. Dement; SUMURU: A Network Configuration for the Future; Mini-Micro Systems, May 1982, pp. 311-312.
[G85] T. Gilb; Evolutionary Delivery versus the 'Waterfall Model'; ACM Software Requirements Eng. Notes, July, 1985.
[G89] D. R. Graham; Incremental development: review of nonmonolithic life-cycle development models; Information and Software Technology, Vol. 31, No. 1, pp.7-20, january/february 1989.

[G94] Grudin Jonathan; Computer-Supported Cooperative Work: History and Focus; Computer IEEE, May 1994.

[GB86] J. Gonzalez-Sustaeta, Alejandro P. Buchmann: An automated database design tool using the ELKA conceptual model. DAC (ACM/IEEE Design Automation Conference) 1986: 752-759; IEEE Computer Society Press

[GHBKB03] Paul Grünbacher, Michael Halling, Stefan Biffl, Hasan Kitapci, Barry W. Boehm: Repeatable Quality Assurance Techniques for Requirements Negotiations. HICSS (36th Hawaii International Conference on System Sciences) 2003: 23

[GHJ94] Erich Gamma, Richard Helm, Ralph Jonson, John Vlissides; Design Patterns: Elements of Reusable Object-Oriented Software; Addison-Wesley, 1994.

[GJM00] M. González-García, R. Jacinto-Montes, A. M. Martínez-Enríquez. CWSLR Model Used to Synthesize a Software Development Environment and an Application Design Tool. Proc. of 2000 International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'2000, special session on Coordination in Parallel and Distributed Applications and Activities, June 29 - 29, 2000, ISBN: 1-892512-23-8.

[GJM99] González-García Moisés; Jacinto-Montes Raúl, Martínez-Enriquez Ana Ma; CWSLR model for CSCW Information Systems; Proc. Of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'99, special session on Coordination in Parallel and Distributed Applications and Activities, June 28 – July 1, 1999, pp. 53-59, ISBN: 1-892512-09-2.

[GJM99B] González-García Moisés, Jacinto-Montes Raúl, Martínez-Enriquez Ana Ma.; CWSLR model: three dimensions, refined to a similar detail; Quinta Conferencia de Ingeniería Eléctrica CIE99, septiembre de 1999.

[GM02] Moisés González-García Ana María Martínez-Enríquez; Integration of Concepts in a Software Development Environment; The 2002 International Multiconferences in Computer Science, The 2002 International Conference on Software Engineering Research and Practice SERP'02, Las Vegas, June 24-27, 2002, pp. 184 – 190, ISBN: 1-892512-99-8.

[GM04] Moisés González-García, Ana María Martínez Enríquez; The Architectural and Group Development Method: an Experimentation; Workshop on Quantitative Techniques for Software Agile Process (QUTE-SWAP), ACM SIGSOFT 2004 / FSE-12 (Foundations on Software Engineering); Oct. 31 – Nov. 6, 2004.

[GMJ98] González-García Moisés, Martínez-Enriquez Ana Ma., Jacinto-Montes Raúl; CWSLR model for CSCW Information Systems: Storage Dimension; Cuarta Conferencia de Ingeniería Eléctrica CIE98, 9-11 septiembre de 1998.

[GS93] David Garlan and Mary Shaw; An introduction to a software architecture; In V. Ambriola and G. Tortora, editors, Advances in Software Engineering and Knowledge Engineering, pp. 1-39, Singapore, 1993. World Scientific Publishing Company.

[H01] Wolfgang Hesse; Dinosaur Meets Archaeopteryx?- Seven Theses on Rational's Unified Process (RUP); Proc. CAiSE'98/LFIP 8.1 Int. Workshop on Evaluation of Modeling Methods in System Analysis and Design (EMMSAD'01),Interlaken 2001

[H03] W. Hesse; "Dinosaur meets Archaeopteryx? or: Is there an alternative for Rational's Unified Process?", Software and System Modeling Vol. 2, No. 4, pp. 240-247, 2003.

[H89] Humphrey, Watts S; Managing the Software Process; Addison-Wesley, 1989.

[H95] Watts S. Humphrey; A Discipline for Software Engineering; Addison-Wesley, 1995.

[H99] Watts S. Humphrey; Introduction to Team Software Process; Addison-Wesley 1999, ISBN: 0-201-47719-X.

[HC01] Jim Highsmith, Alistair Cockburn; Agile Software Development: The Business of Innovation; IEEE Computer Vol.34 No. 9, pp 120 – 122, September 2001.

[HT92] Kenneth J. Hintz, Daniel Tabak; Microcontrollers: Architecture, Implementation, and Programming; McGraw-Hill, 1992, ISBN 0-07-028977-8.
[I87] Frank S. Ingrassia. The Unit Development Folder (UDF): A Ten-Year Perspective. EH0263-4/87/0000/0405$01.00 1987 IEEE., pp. 405-415.
[I95] Iochpe C.; Improving requirements analysis through CSCW; First International Seminar CYTED-RITOS, of Cooperative Systems, Lisboa Portugal, september 1995, pp. 29-37.
[IEEE-610] IEEE; Standard Computer Dictionary- A compilation of IEEE Standard Computer Glossaries; ISBN 1-55937-079-3, 1990
[IEEE-610.12] IEEE, Standard Glossary of Software Engineering Technology, ANSI/IEEE Std. 610.12, 1990.
[IEEE-1028] IEEE Std. 1028 – 1997; IEEE Standard for Software Reviews; IEEE Software Engineering Standards Committee, 1997.
[IEEE-1220] IEEE 1220; IEEE Standard for Application and Management of the Systems Engineering Process; IEEE Std. 1220 – 1998; IEEE Computer Society; ISBN 0-7381-1544-4 SS94720.
[IEEE-12207] IEEE/EIA 12207.0-1996: Industry Implementation of International Standard ISO/IEC 12207: 1995; ISO/IEC 12207 Standard for Information Technology – Software Life Cycle Processes; IEEE SBN 0-7381-0428-0, ss94581, 1998
[IEEE SWEBOK] Alain Abran, James W. Moore, Editors; Guide to the Software Engineering Body of Knowledge - Trial Version 1.0; IEEE Computer Society, 2001
[ISO-15504] ISO/IEC; ISO/IEC TR 15504: Information Technology - Software Process Assessment (parts 1- 9); International Organization for Standardization and the International Electrotechnical Commission, 1998 (part 5 was published in 1999). (Available at http://www.seg.iit.nrc.ca/spice)
[ISO-9126] International Standard ISO/IEC 9126 – 1; Software engineering – Product quality – Part: 1 – Quality model; ISO/IEC 2001, www.iso.ch .
[ISO-90003] ISO/IEC 90003:2004; Software engineering -- Guidelines for the application of ISO 9001:2000 to computer software; International Organization for Standardization, 2004.
[J88] Juran, J. M. Juran on Planning for Quality. New York, New York: MacMillan, 1988.
[J95] M.A. Jackson, Software requirement & Specifications A lexicon of practice, principles and prejudices; Addison Wesley, ACM Press, [1995].
[JBR99] I. Jacobson, G. Booch, J. Rumbaugh. The Unified Software Development Process. Addison Wesley, 1999.
[JEJ95] Ivan Jacobson, Maria Ericson, Agneta Jacobson; THE OBJECT ADVANTAGE business process reeingineering with object technology; Addison-Wesley, 1995.
[KP01] Nikos I. Karacapilidis, Dimitris Papadias: Computer supported argumentation and collaborative decision making: the HERMES system. Information Systems 26(4): 259-277 (2001).
[LB03] Craig Larman, Victor R. Basili; Iterative and Incremental Development: A Brief Histoty; IEEE Computer, June 2003.
[LDM92] Axel van Lamsweerde, R. Darimont and Philippe Massonet; The Meeting Scheduler System: Preliminary Definition ; University of Louvain, Unit\'e d'informatique, B-1348 Louvain-la-Neuve (Belgium), October 1992.
[LCM93] Axel van Lamsweerde, Charles Christoph, and Philippe Massonet; Extending the Meeting Scheduler System to Support Conflict Resolution; University of Louvain, Unit\'e d'informatique, B-1348 Louvain-la-Neuve (Belgium), November 1993.

[K04] P. Kroll, “Planning and estimating a RUP project using IBM Rational SUMMIT Ascendant”, IBM Rational, 11 May 2004.
[K04-A] Frank Keenan; Agile Process Tailoring and Process AnalYlis (APTLY); Proceedings of the 26th Internacional Conference on Sopftware Engineering (ICSE’04)
[K78] E. Kreyszig, “Introductory Mathematical Statistics: Principles and Methods”, John Wiley & Sons 1970, Ch. 13, 20, pp. 219-228**,** 234-239, 403-405, Spanish version by Limusa, 1978.
[K96] Kari Kuutti; Debates in IS and CSCW Research: Anticipating System Design for Post-Fordist Work; en Information Technology and Changes in Organisational Work, W. Orlikowski, et al., Editors, Chapman & Hall, pp. 287-308, London, 1996.
[M91] J. Martin; Rapid Application Development; Macmillan, 1991.
[MC90] Thomas W. Malone, Kevin Crowston: What is Coordination Theory and How Can It Help Design Cooperative Work Systems? CSCW 1990: 357-370; Proceedings of the Conference on Computer Supported Cooperative Work, October 7-10, 1990, Los Angeles, CA, USA. ACM, 1990
[MC94] Thomas W. Malone, Kevin Crowston: The Interdisciplinary Study of Coordination. ACM Computing Surveys 26(1): 87-119 (1994)
[MKKA03] Mike Moore, Rick Kazman, Mark Klein, Jai Asundi: Quantifying the Value of Architecture Design Decisions: Lessons from the Field. ICSE 2003: 557-563
[MM03] Joaquin Miller and Jishnu Mukerji (Editords); MDA Guide Version 1.0.1; Object management Group, Document Number: omg/2003-06-01, 12th June 2003
[MS02] Peter Mangan, Shazia Sadig: On Building SubProcess Models for Flexible Processes. Australasian Database Conference 2002; Thirteenth Australasian Database Conference (ADC2002), Monash University, Melbourne, Victoria, January/February 2002. ACS Series Conferences in Research and Practice in Information Technology, Australian Computer Society Inc., 2002, ISBN 0-909-92583-6
[MS98] Barbara C. McNurlin, Ralph H. Sprague Jr.; Information Systems Management in Practice Fourth Edition; Prentice Hall, 1998
[N98] J. T. Nosek; The Case for Collaborative Programming; Comm. ACM, Vol. 41, No. 3, 1998, pp. 105-108
[NDD01] Olga Nabuco, Khalil Drira, Edeneziano Dantas: A Layered Design Model for Knowledge and Information Sharing Cooperative Systems. WETICE 2001: 305-310; 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2001), 20-22 June 2001, Cambridge, MA, USA. IEEE Computer Society 2001, ISBN 0-7695-1269-0
[NW03] Andrey Naumenko, Alain Wegmann; Two Aproaches in System Modeling and their Illustrations with MDA and RM-ODP; International Conference on Enterprise Information Systems 2003 (ICEIS’2003), Proceedings pp. 398 – 402
[NWC97] Minh N. Nguyen, Alf Inge Wang, Reidar Conradi: Total Software Process Model Evolution in EPOS (Experience Report). ICSE 1997: 390-399; 19th International Conference on Software Engineering, May 17-23, 1997, Boston, Massachusetts, USA. ACM Press, 1997, ISBN 0-89791-914-9
[O02] A. Jefferson Offutt: Quality Attributes of Web Software Applications. IEEE Software 19(2): 25-32 (2002)
[O03]Hanna Oktaba et. al.; Modelo de Procesos para la Industria del Software (MoProSoft); Secretaría de Economía (México), UNAM, AMCIS; www.software.net.mx
[O81] Keneth Orr. Structured Requirements Definition. Ken Orr and Associates Inc. 1981, ISBN 0-9605884-0-X.

[ODP98] ISO/IEC 10746-1, 2, 3, 4 | ITU-T Recommendation X.901, X.902, X.903, X.904. "Open Distributed Processing - Reference Model". OMG, 1995-98. http://isotc.iso.ch/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm

[OMG02A] OMG-CORBA Common Object Request Broker Architecture: Core Specification, December 2002, Version 3.0 formal/02-12-06, An Adopted Specification of the Object Management Group

[OMG02B] OMG-MOF Meta Object Facility Specification, Version 1.4, April 2002, An Adopted Specification of the Object Management Group

[OMG03A] OMG-MDA Model Driven Architecture Guide Version 1.0, omg/2003-05-01, 1 May 2003

[OMG03B] OMG Unified Modeling Language Specification; March 2003, Version 1.5, formal/03-03-01, An Adopted Formal Specification of The Object Management Group

[OMG95] OMG-OMA Richard Mark Soley, Christopher M. Stone; Object Management Architecture Guide, Revision 3.0, Third Edition, June 13, 1995

[OMG-SPEMS02] Object Management Group; Software Process Engineering Metamodel Specification; formal: 02-11-14, November 2002

[OG02] TOGAF Version 8 ("Enterprise Edition"), published in December 2002; The Open Group Architecture Framework; www.opengroup.org/architecture/togaf8-doc/arch/

[OG03] TOGAF Version 7 ("Technical Edition"), published in December 2001; The Open Group Architecture Framework; //www.opengroup.org/architecture/togaf7/index7.htm

[OG93] Open Group; System Management: Reference Model; G207 ISBN 1-85912-005-9 September 1993; www.opengroup.org/products/publications/catalog/g207.htm

[OG97] T151X Distributed Computing Environment DCE 1.2.2 Documentation - Full Set; November 1997: The Open Group

[OR00] G. Orrand, T. E. Reeves, "Function Point Counting: One Program's Experience", Journal of Systems and Software Vol. 53, N.3, pp.239-244, 2000.

[P01] Dewayne E. Perry: Software Architecture: Leverage for System/Program Comprehension. IWPC 2001 (9th International Workshop on Program Comprehension): 123-126; 12-13 May 2001, Toronto, Canada. IEEE Computer Society 2001, ISBN 0-7695-1131-7

[P84] Raymond R. Panko: 38 Offices: Analyzing Needs in Individual Offices. TOIS (ACM Transactions on Office Information Systems) 2(3): 226-234 (1984); ACM Transactions on Office Information Systems (TOIS)

[PAF01] O. Pastor, S. M. Abrahao, J. J. Fons; Object-Oriented Approach to Automated Web Applications Development; 2nd Int. Conference on Electronic Commerce and Web Technologies (EC-Web'01), Munich, Germany, Springere Verlang, 2001, pp. 16 – 28.

[PB04] Pamela Curtis, Bill Thomas; TSP Accelerates Software Quality Improvements at NAVAIR**;** news@sei, 2004, Number 1, http://interactive.sei.cmu.edu

[PC91] M.C. Paulk, B. Curtis, M.B. Chrissis, et al, *Capability Maturity Model for Software*, Software Engineering Institute, CMU/SEI-91-TR-24, ADA240603, August 1991.

[PF02] S.R.Palmer a d J.M.Felsi g; A Practical Guide to Feature-Driven Development, 2002.

[PW92] Dewayme E. Perry, and Alexander L. Wolf; Foundations for the study of software architecture; ACM SIGSOFT Software Engineering Notes, Vol. 17 No. 4, pp. 40-52, October 1992.

[R81] Guillermo Rodriguez Ortiz; The ELKA Model Approach to the Design of Database Conceptual Models; Ph.D. Dissertation, University of California, Los Angeles, 1981.

[R90] Walter Royce; TRW's Ada Process Model for Incremental Development of Large Software Systems; Proceedings of the 12th International Conference on Software Engineering, 1990 ICSE 1990, pp. 2-11

[R98] W. Royce; Software Project Management; Addison-Wesley, 1998.

[RADC83] Rome Air Development Center; Quality Metrics for Distributed Systems; Tech. Report: RADC-TR-83-175, Vols. I – III, 1983.

[RADC85] Rome Air Development Center; Specification of Software Quality Attributes; Tech. Report: RADC-TR-85-37, Vols. I - III, 1985.

[RAT98] Rational Software Corporation, "Rational Unified Process Best Practices for Software Development Teams," White Paper TP026B Rev 11/01, 1998, http://www.rational.com/media/whitepapers/rup_bestpractices.pdf.

[RUP03a] "RUP implementation guide Part I: Recommended strategy and typical issues and risks", The Rational Edge, 19 August 2003.

[RUP03b]"RUP implementation guide Part II: Best Practices and Key Lessons Learned from RUP implementation projects", The Rational Edge, November 2003.

[S05] Martin Soukup; Model Driven Architecture: Feasibility or Fallacy?; XML 2004 Proceedings, November 15-19, 2004;
www.idealliance.org/proceedings/xml04/index.html

[S06] Douglas C. Schmidt; Model-Driven Engineering; IEEE Computer, February 2006, pp. 25 – 31.

[S97] Oliver Sims, "The OMG Business Object Facility and the OMG Business Object", in Sutherland J. D. et al (Eds) Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings, Springer-Verlag, 1997, ISBN: 3540760962.

[S97-a] J.Stapleto; Dynamic systems development method - The method in practice :Addiso Wesley,1997.

[SB82] W. Swartout, R. Baltzer; On the Inevitable Intertwining of Specification and Implementation; Comm. ACM, July 1982, pp.438-440.

[SB92] Schmidt, K y Bannon, L; Taking CSCW Seriously: Supporting Articulation Work; Computer Supported Cooperative Work Journal, V. 1, N. 1, 1992, pp. 7-40.

[SEI-CMMI-C] Capability Maturity Model® Integration (CMMI℠), Version 1.1; CMMI℠ for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing, (CMMI-SE/SW/IPPD/SS, V1.1) - Continuous Representation; CMU/SEI-2002-TR-011, ESC-TR-2002-011

[SEI-CMMI-S] Capability Maturity Model® Integration (CMMI℠), Version 1.1; CMMI℠ for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing, (CMMI-SE/SW/IPPD/SS, V1.1) - Staged Representation; CMU/SEI-2002-TR-012 , ESC-TR-2002-012

[SG95] Mary Shaw and David Garlan; Formulation and formalisms in software architecture; In Jan van Leeuwen, editor, Computer Science Today: Recent Trends and Developments; Lecture Notes in Computer Science, Vol. 1000. Springer Verlang, 1995.

[SG95A] M .Shaw, D. Garlan, R. Allen, D. Klein, J. Ockerbloom, C. Scott, and M. Schumaker, "Candidate Model Problems in Software Architecture", Draft Technical Report 15-675, CSD CMU, 1995.

[SK01] Andrew P. Snow, Mark Keil; The Challenges of Accurate Project Status; 34 Annual Hawaii International Conference on System Sciences (HICSS-34), Vol. 8, 2001, pp. 3133-3142

[SS00] Sendall and Alfred Strohmeier, "Specifying System Behavior in UML", Technical Report DI/00/343, EPFL (Swiss Federal Institute of Technology), 2000.

[SW01] Judith A. Stafford, Alexander L. Wolf: Architecture-Level Dependence Analysis for Software Systems. International Journal of Software Engineering and Knowledge Engineering 11(4): 431-451 (2001)

[SWM03] Ya-Diane H. Sonnenwald, Mary C. Whitton, Kelly Maglaughlin: Evaluating a scientific collaboratory: Results of a controlled experiment. ACM Trans. Comput.-Hum. Interact. 10(2): 150-176 (2003)

[SWN03] P. D. Stotts, L. Williams, N. Nagappan, P. Baheti, D. Jen, A. Jackson, "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming", XP/Agile Universe, pp. 129-141, 2003.

[T88] Thierauf Robert J.; User-Oriented Decision Support Systems: Accent on Problem Finding; Prentice Hall, 1988.

[TBK99] D. P. Truex, R. Baskerville y H. Klein; Growing Systems in Emergent Organizations; Communications of the ACM, Vol. 42, pp. 117-123, 1999.

[TEL05] G. Taentzer, K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovsky, U. Prange, D. Varro and S. Varro-Gyapay. Model Transformation by Graph Transformation: A Comparative Study; Workshop Model Transformations in Practice, 2005. http://tfs.cs.tu-berlin.de/~uprange/publications.html

[TRS93] Michael Twidale, Tom Rodden, Ian Sommerville: The Designers' Notepad: Supporting and Understanding Cooperative Design. ECSCW 1993: 91-106 ; European Conference on Computer Supported Cooperative Work, ECSCW'93, Milano, 13-17 September 1993, Proceedings. Kluwer Academic Publishers, 1993.

[V95] Villemur T.; Conception de Services et de Protocoles Pour la Gestion de Groupes Coopératifs; Thèse de Docteur, préparée au Laboratoire d'Analyse et d'Architecture des Systèmes de CNRS, Janvier 1995.

[W00] Laurie Ann Williams; The Collaborative Software Process; PhD dissertation of The University of Utah, agosto del 2000.

[WK00] L. A. Williams, R. R. Kessler; All I Ever Needed to Know About Pair Programming I learned in Kinder-garten; Comm. of ACM, Vol. 43, No. 5, 200, pp. 108-114, 2000

[W01a] P. R. Wyrick, "Unified Software Project Management Using Rational ProjectConsole to Collect Microsoft Project Metrics", Rational Software White paper, 17 May 2001.

[W01b] P. R. Wyrick, B. Gilbert, and D. West, "Unified Software Project Management Rational Project Console Sample Metrics", Rational Software White paper, 2001.

[WGDGKR99] David Graham Wastell, Luuk Groenewegen, Jean-Claude Derniame, Mark Greenwood, Peter Kawalek, Ian Roberston: Software Process: Key Issues and Future Directions. 201-226; Software Process: Principles, Methodology, Technology. Lecture Notes in Computer Science 1500 Springer 1999, ISBN 3-540-65516-6.