HERRAMIENTA CASE PARA LA TRANSFORMACIÓN DE MODELOS DE LÍNEAS DE PRODUCCIÓN MEDIANTE MDA

JORGE IVÁN RAMÍREZ HERNÁNDEZ

FUNDACIÓN UNIVERSITARIA SAN MARTÍN

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ

2009

HERRAMIENTA CASE PARA LA TRANSFORMACIÓN DE MODELOS DE LÍNEAS DE PRODUCCIÓN MEDIANTE MDA

JORGE IVÁN RAMÍREZ HERNÁNDEZ

042045

JR042045@INGENIERIA.SANMARTIN.EDU.CO

ANTEPROYECTO DE GRADO

ASESOR TÉCNICO

ANDRÉS FELIPE SOLARTE IMBACHI

GRUPO DE TRABAJO

FUNDACIÓN UNIVERSITARIA SAN MARTÍN

FACULTAD DE INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ

2009

CONTENIDO

pág.

LISTA DE TABLAS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

1 PROBLEMA

Las líneas de producción reflejan los procesos necesarios para la creación de productos, donde generalmente pueden existir errores, que se reflejan en pierdas de tiempo y productividad, generando costos innecesarios a las empresas, por lo que en ocasiones se quiere automatizar esta línea de producción por medio de productos de software, los cuales son realizados por ingenieros que deben recolectar información importante de las personas que son expertas en el negocio. A menudo estas personas no saben transmitir correctamente esta información o el ingeniero no la comprende bien, así que al momento de desarrollar la aplicación se presentan problemas o peor, al entregar la aplicación el cliente no la recibe porque no era lo que quería.

Si la comunicación con los expertos en líneas de producción no es clara al momento de especificar la aplicación de automatización, la solución no serviría y habría que hacerla de nuevo, esto conlleva multas y compensaciones.

Cuando se analiza y especifica la solución a un problema, el cliente y especialmente el experto del negocio, son personas muy importantes, ya que son estas las que proporcionan toda la información necesaria, pero podría ser mejor si ellas participaran en el desarrollo de la solución, creando para la organización un diagrama donde no omitan detalles y además de esto se pueda transformar el diagrama en código fuente de un lenguaje de propósito general automáticamente, entonces no se perdería información y si sucediera la información perdida se minimizaría en un alto grado, además de crear una codificación casi completa a la automatización del problema para una optimización más exacta.

2 JUSTIFICACIÓN

Los artículos que se encuentran en el mercado tratan de ofrecer al consumidor la mejor calidad posible con el mínimo costo asociado, esto sin tener en cuenta que existen más empresas que fabrican el mismo artículo y la única forma de conseguir más clientes es ofreciendo más calidad y/o menos precio.

Los fabricantes al darse cuenta de esto deben ofrecer una mayor calidad, verificando que las utilidades sean las esperadas para ofrecer un menor costo a los clientes finales, esto es posible si se piensa en optimizar la línea de producción, ya que es en esta donde se puede encontrar muchas ventajas en tiempo, utilización de materia prima, manejo de desperdicios, etc.

La optimización de una línea de producción es casi una ciencia en sí, ya que hay que aplicar modelos matemáticos que resultan complejos para las personas que no los conocen o no los saben aplicar y hay que contratar a personas que ofrecen este servicio, pero estas personas al no conocer completamente del negocio no son capaces de optimizar al máximo la línea de producción y los costos son elevados.

Es entonces que la herramienta que se planea desarrollar ayudará a la solución de este problema, ya que permite a cualquier persona con conocimientos básicos de computación realizar un diseño de una línea de producción de forma gráfica para posteriormente optimizarla; esto ayudará a una disminución de costos al momento de contratar expertos que tal vez demoren más tiempo en la optimización.

Por otra parte la Fundación Universitaria San Martín obtendrá un reconocimiento al mostrar este como parte de un proyecto macro de optimización de modelos matemáticos, el cual sera un proyecto que la universidad podrá exhibir en el campo académico y ofrecer como un servicio para personas interesadas en usarlo.

Una parte que vale la pena no olvidar es que este proyecto servirá de base para otras personas interesadas en el uso de tecnologías, ya que esta es la primera vez que se usan tecnologías de transformación de modelos en la universidad, así que servirá como un apoyo a otras personas que deseen hacer una investigación sobre temas similares.

Por último cabe mencionar que existen herramientas computacionales cuya funcionalidad es resolver modelos matemáticos, pero las personas interesadas en optimizar una línea de producción, no están familiarizadas con estas herramientas y obviamente no conocen la forma correcta de crear un modelo adecuado para su línea de producción.

3 OBJETIVOS

3.1.1 OBJETIVO GENERAL

Crear una herramienta CASE para la representación gráfica de una línea de producción, permitiendo la generación de un modelo matemático que represente la información particular del mismo, y transformarlo por medio de herramientas MDA hasta llevarlo a su representación de metamodelo en la plataforma AMM para su optimización, generando la visualización de los resultados.

3.1.2 OBJETIVOS ESPECÍFICOS

- Crear una herramienta CASE, que permita la creación de diagramas, que representen líneas de producción arrastrando los componentes al área de trabajo.

- Generar un metamodelo que represente los conceptos de una línea de producción, basado en la abstracción de conceptos comunes y sus relaciones en diferentes escenarios o tipos de líneas.

- Seleccionar un modelo matemático que esté comprobado en la industria o en escenarios similares usando cuadros comparativos, con el objetivo de crear un conjunto de datos de entrada para alimentarlo desde la capa del metamodelo de negocios y los subsecuentes metamodelos de nivel inferior.

- Generar un metamodelo para representar el modelo matemático seleccionado previamente, con la característica de poder extenderlo a la representación de cualquier modelo matemático, abstrayendo los conceptos comunes a todos los modelos matemáticos.

- Desarrollar un componente de software con la responsabilidad de generar el código necesario para extraer la información almacenada en el metamodelo de modelos matemáticos usando Acceleo y/o Velocity, y alimentar la plataforma AMM a través de la interfaz de comunicación expuesta para ello.

- Identificar las reglas y mapeos necesarios para la transformación entre modelos, para lograr que el modelo encaje en el metamodelo matemático por medio de MDA.

- Identificar cuántos y cuáles son los modelos intermedios necesarios para lograr un modelo matemático con la información y organización requerida por la herramienta de optimización.

- Crear un componente de software que sea capaz de capturar la salida de la herramienta AMM con el modelo ya optimizado, el cual mostrara esta información de forma gráfica al usuario en forma de reporte.

4 MARCO REFERENCIAL

4.1 ANTECEDENTES

4.1.1 Optimización de una línea de producción mediante el estudio de métodos y tiempos y propuestas de mejora (Lafuente,2007)

Este es un proyecto que se centra en la optimización de una línea de producción de carpintería de aluminio maximizando la producción y analizando la el orden en el que se realizan los procesos.

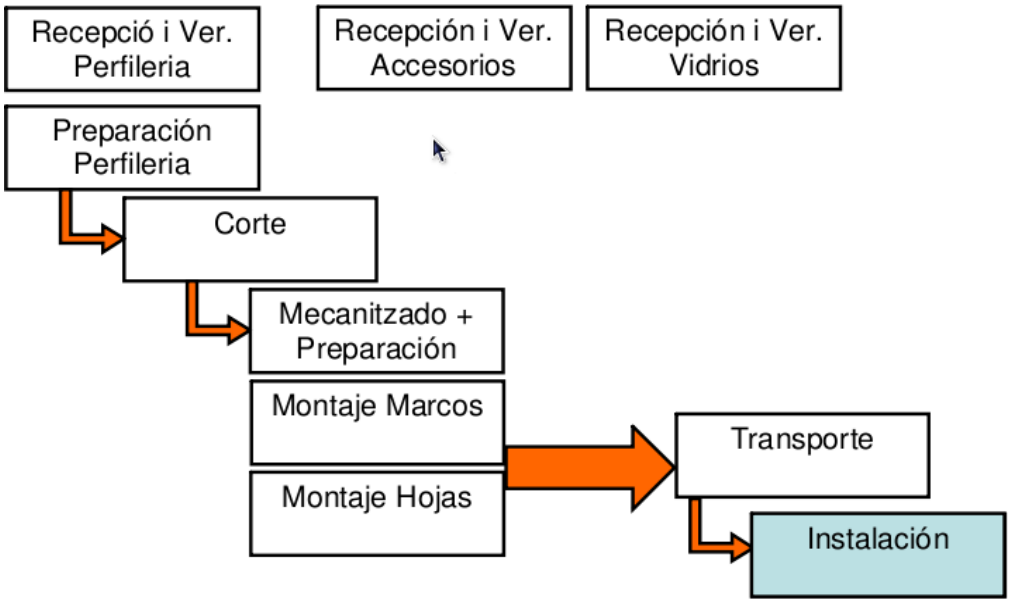


Figura 1: Modelo de línea de producción de aluminio (Lafuente,2007)

La forma en que abordan el problema de optimización de la línea de producción es alterando el orden de trabajo para lograr como resultado una mayor producción; a esta solución se llegó como conclusión del análisis de las estaciones de trabajo.

Como propuesta de solución, lo que se realizó fue una mejora en los tiempos de los procesos, que en este caso son realizados por maquinaria con operarios a cargo implementando un sistema de cronometraje BEDAUX con el cual controlaron los tiempos de los operarios.

Hay que destacar la forma en que realizaron la investigación de la línea de producción, donde se enfocaron en las estaciones de trabajo como corazón de la línea de producción, y dejando como una segunda instancia el método

de transporte de los materiales, es entonces que se podría tomar esta forma de abordar problemas de líneas de producción, enfocándose en las estaciones de trabajo donde se realizan los procesos, ya que son estas las que tienen una mayor probabilidad de optimización, sin dejar de lado el método de transporte, el cual también influye en los resultados de toda la línea de producción.

4.1.2 Modelo de programación jerárquica de la producción en un Job shop flexible con interrupciones y tiempos de alistamiento dependientes de la secuencia (Osorio,2008)

Dentro de este paper se da a conocer una propuesta usando Flexible Job Shop para la solución de problemas de programación de producción incluyendo aislamientos y tiempos independientes de la secuencia, dando como respuesta una solución NPHard donde están involucrados seis máquinas, seis trabajos y trece operaciones.

Dado las múltiples combinaciones de todos los elementos involucrados en el problema, hacen que la elección de flexible job shop sea útil, ya que este método enfoca en dos aspectos, máquinas y trabajo, donde el trabajo consta de varias operaciones secuenciales que realizan varias máquinas en un determinado tiempo y las máquinas solo pueden realizar una operación.

Al evaluar los factores de las máquinas y el trabajo se busca es minimizar los tiempos que gastan las máquinas en sus respectivas operaciones, donde la aplicación del job shop descompone los problemas en unos más pequeños.

Para el uso que se dara al proyecto, se tomara como base el modelo que se realiza para resolver el problema, el cual es un modelo lineal.

$$Z = Max \sum_{J=1}^{N} T_{R,J} X_{R,J} \qquad \forall R \in (1,\ldots,L) \qquad (1)$$

Ecuación 1: función propuesta para la solución de optimización con fjsp (Osorio,2008)

Con el que se describen los tiempos de trabajo por medio de T(r,j) donde se busca minimizar Z.

$$Min\, Z \qquad (2)$$

$$Z \geq \sum_{J=1}^{N} T_{R,J} X_{R,J} \qquad \forall R, R = 1 \ldots N \quad (3)$$

$$\sum_{R=1}^{L} X_{R,J} = 1 \qquad \forall J, = 1 \ldots N \quad (4)$$

$$X_{R,J} = \begin{cases} 1\, si\, el\, trabajo\, J\, se\, asigna\, al\, centro\, R \\ 0\, de\, lo\, contrario \end{cases}$$

Ecuación 2: Modelo lineal para la solución (Osorio,2008)

4.2 MARCO CONCEPTUAL

4.2.1 Línea de producción (Muther,1990)

Una línea de producción es un sistema que está compuesto por varios elementos que deben trabajar de forma sincronizada para tomar un producto base o materia prima y convertirlo en otro producto, las líneas de producción actuales son generalmente compuesta por máquinas que ofrecen beneficios como:

- Mínimo tiempo entre procesos.
- Mayor cantidad de producción.
- Mayor sincronización.
- Menor costo de transporte entre estaciones.

Las línea de producción se podrían agrupar en dos grupos, las líneas de producción balanceadas, que son las que dan la misma cantidad de trabajo para cada estación y las des balanceadas que dan diferentes cargas de trabajo.

Por otra parte las líneas de producción se pueden categorizar por métodos de producción, que pueden ser en cadena, urinaria o por proyecto e intermitente o por lotes.

4.2.1.1 Método de Producción en Cadena (SEDE BOGOTA,2009)

Este es uno de los métodos en serie, que tiene la capacidad de determinar la cantidad que hay que producir y almacenar para cada uno de los productos de la línea de producción, además de calcular el numero de personas responsables de cada máquina.

Este método es uno de los mas exitosos desde hace casi un siglo gracias a que implementa las siguientes características, las cuales también fueron la causa de una búsqueda de una mejor técnica:

- La cantidad de producción en muy alta, al compararse con la diversidad de productos.
- Debe tener mantenimiento constante.
- Las personas que intervienen no necesitan un alto grado de entrenamiento.
- La producción por empleado es alta.

En general este método permite la producción de todos los productos que se ofrecen mediante una sola línea de producción.

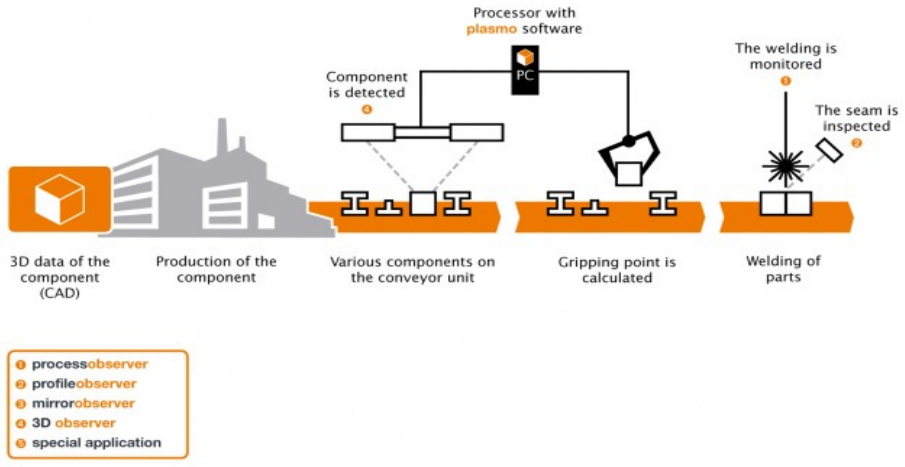


Figura 2: Ejemplo de una línea de producción en cadena (process chain,2009)

4.2.1.2 Método de Producción Unitaria o por Proyecto (SEDE BOGOTA,2009)

El método de producción unitaria realiza la producción por pedidos, así que no se empieza una producción hasta que no se haya concretado un pedido y por esta causa, la variedad de producción es muy pequeña.

Este tipo de producción usa generalmente el método de PERT para la planificación, el cual ayuda a determinar el tiempo pesimista, el mas probable y el optimista

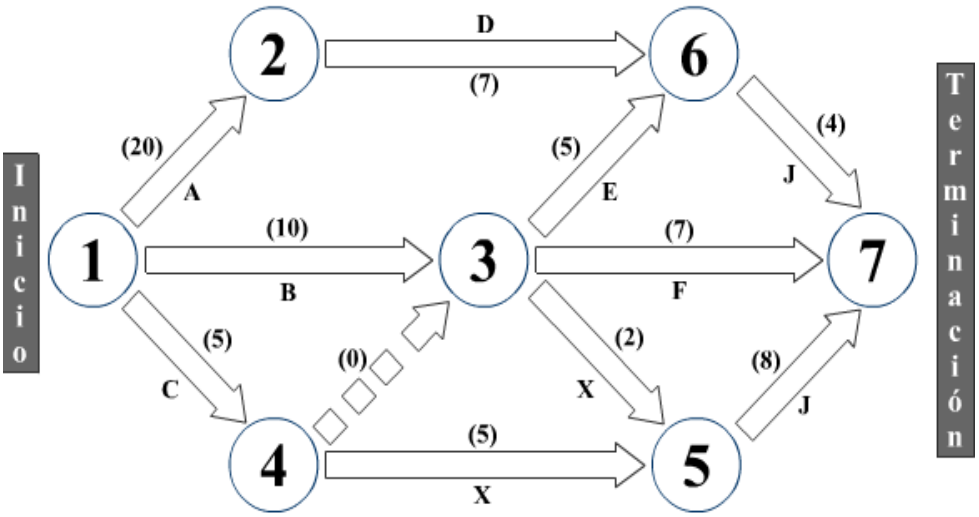


Figura 3: Ejemplo de producción unitaria o por proyecto(SEDE BOGOTA,2009)

Dentro de este sistemas, no existe un transporte automatizado de los productos, pero si existe un orden en el que debe ser realizado haciendo la labor prácticamente personalizada, adquiriendo altos costos.

4.2.1.3 Método de Producción Intermitente o por Lotes (SEDE BOGOTA,2009)

La producción por lotes está muy ligada a la demanda del producto, y esta es implantada cuando la producción supera la demanda, así se crean lotes completos del producto antes de empezar uno nuevo, a diferencia de los otros tipos de producción que son continuos.

Al momento de la fabricación, los productos se encuentran en tres estados diferentes:

- Esperando a ser procesado.
- En procesamiento.
- Esperando para la siguiente estación.

Ya que la producción por lotes depende de la demanda, es importante determinar el tamaño del lote ya que de esta forma se ahorran recursos y gastos innecesarios.

4.2.2 Distribución de una línea de Producción (Balanceo de Línea,2009)

Los productos ya sean materia prima o los estados por los que debe pasar el producto final, tienen la obligación de moverse, esto lo hacer por medio de procesos automatizados, haciendo esta la parte más importante de una línea de producción, además tienen las estaciones de trabajo, que se encargan de realizar una operación sobre la materia prima que llega.

En general, la forma en que se compone una línea de producción es por un método de transporte que alimenta el sistema con la materia prima, para pasar al sistema de transporte interno de la línea de producción que es el encargado de enviar el producto a las diferentes estaciones de trabajo para que sea procesado.

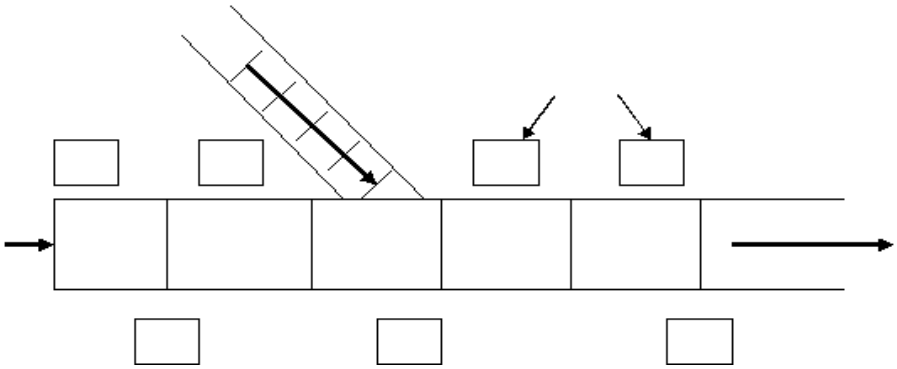


Figura 4: Ejemplo de una línea de producción (Balanceo de Línea,2009)

4.2.3 Job Shop Flexible

Este problema, es una subclase del problema jop shop, el cual se busca dar solución mediante el uso y adaptación de una o varias máquinas comprendidas dentro de un conjunto.

Dentro de la solución del flexible job shop se tienen en cuenta dos tipos de conjuntos, uno de máquinas y otro de trabajos, en el que se busca un conjunto de máquinas que cumplan un proceso secuencial, en el que se busca la máquina que represente un menor “costo” para la realizar un trabajo determinado.

El problema de flexible job shop esta dado por *nxm* donde *n* es el numero de máquinas y *m* el número de trabajos entonces este puede generar *(n!)^m* soluciones posibles ya que hay que asignar las tareas a las máquinas haciendo dos pasos para aplicar este método, 1) asignar las tareas a los recursos 2) secuenciar los recursos dependiendo del orden de las tareas, pero hay que tener en cuenta las premisas:

- Todos los trabajos están disponibles para iniciarse en t=0.
- Todas las máquinas están listas para funcionar en t=0.
- Los tiempos depende de la secuencia dada.

4.3 MARCO TEÓRICO

4.3.1 OMG (OMG,2009)

Esta es la organización internacional que establece estándares en el desarrollo de software los cuales dan una metodología enfocada hacia el paradigma de objetos desde el año de 1989 por varias de las compañías más influyentes del desarrollo de software para crear un estándar para la manipulación del paradigma de objetos dentro de la industria; de la investigación de estas empresas surge UML, MOF y MDA, entre otros, estos estándares se usan actualmente para la definición de nuevas tecnologías y frameworks que facilitan la construcción de aplicaciones de software.

4.3.2 MDA (OMG,2009)(MELLOR,2004)

Arquitectura guiada por el modelo, es un estándar de la OMG para la representación de modelos, metamodelos y la transformación de estos, este estándar surge del la estructuración y especificación de los acrchivos generados por diferentes herramientas basadas en modelos en el año 2001, el cual se basa en la propiedad de la abstracción como concepto de transformación pasando de un modelo con un nivel de abstracción alto a un nivel de abstracción mediante la definición los mapeos y las marcas que debe seguir un modelos para llegar a otro.

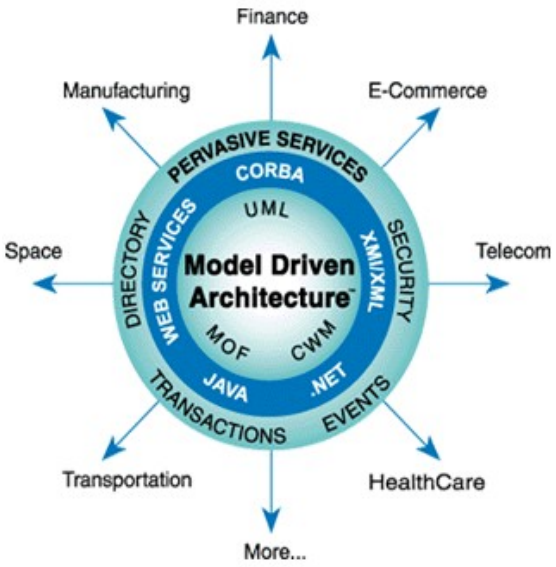


Figura 5: MDA core (OMG,2009)

MDA define una aproximación hacia el ambiente del negocio más que otros estándares, ya que se enfoca mucho más en el moldeamiento de la solución de un problema determinado sin tener en cuenta la tecnología especifica con la que se va a implementar, ya que separa totalmente la tecnología de la solucion del negocio.

4.3.3 Abstracción (Mellor,2004) (Miller,2003)

El concepto de abstracción es el nivel de información mostrado por el modelo, donde el nivel de abstracción es definido por la cantidad de datos expresados explícitamente dentro de un modelo, el nivel de abstracción alto es el que muestra pocos datos y esto permite que el usuario o cliente pueda interpretar y entender, mientras que el nivel de abstracción bajo muestra más datos que permiten plantear una mejor solución siendo más útil para las personas expertas.

4.3.4 Refinamiento (Mellor,2004)

El concepto de refinamiento es el grado de información suministrada por un modelo, lo cual nos indica la cantidad de datos que son mostrados explícitamente por un modelo, el refinamiento es lo inverso a la abstracción y esto quiere decir que si un modelo tiene un alto grado de abstracción, entonces va a tener un bajo grado de refinamiento y viceversa.

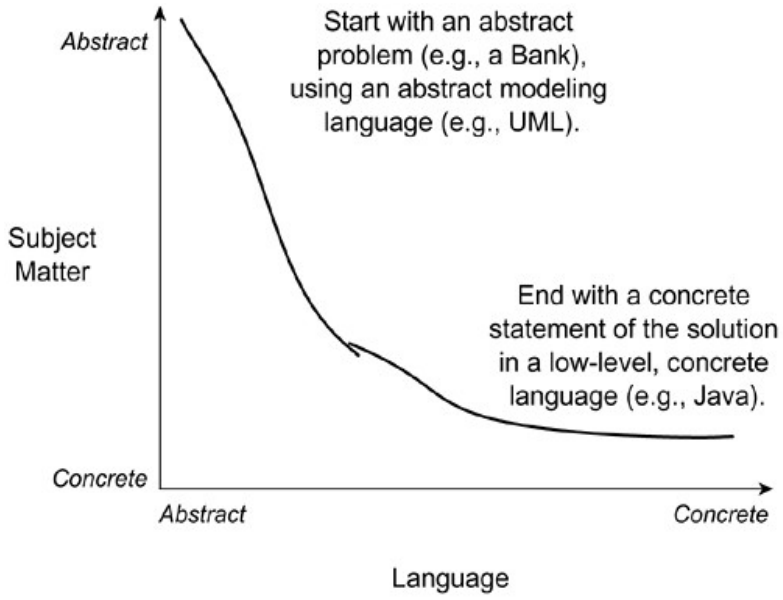


Figura 6: Abstracción vs refinamiento (Mellor,2004)

El refinamiento es también el proceso por el cual un modelo tiene más información y por lo tanto es mas concreto para un uso más específico.

4.3.5 Mapeo (Mellor,2004)

Los mapeos son funciones que tienen una lista de reglas que el modelo origen debe seguir para llegar al modelo final, o modelo siguiente en una cadena de modelos intermedios entre el inicial y el final, estas reglas definen la precondiciones que debe cumplir el modelo para poder ser transformado al siguiente; estos mapeos están definidos dentro del estándar MDA, convirtiéndose en un pilar para la implementación de transformaciones mediante MDA.

4.3.6 Marcas (Miller,2003)

Las marcas influyen en una gran proporción en la transformación de modelos, ya que estas son las que indican la forma y secuencia en que van a ser transformados como CIM, PIM y PSM, que son los modelos generales dentro de la secuencia de transformación. Las marcas son las encargadas de brindar información de las transformaciones de un modelo particular, sin necesidad de estar ligadas al modelo, ya que destruiría la capacidad de transformación a diferentes PSM's

4.3.7 QVT (OMG,2009) (SmartQVT,2009)

Query, view, transformation, es el lenguaje estándar usado para la transformación de modelos definido por la OMG en su rama MDA.

QVT se basa en la definición de reglas para la transformación de un modelo a otro, teniendo en cuenta que los modelos pueden ser representaciones de un mismo metamodelo o diferentes metamodelos; este se divide en tres partes, la primera es query o las consultas que se hacen dentro del dominio del metamodelo y que devuelven uno o varios valores resultado, se puede decir que es una expresión que es evaluada dentro de un metamodelo y que su resultado es un conjunto de instancias de los tipos definidos en el metamodelo, también se encuentra view que es un modelo basado en otro modelo(modelo base) el cual se comporta como una copia y por último, se encuentra transformation que es la encargada de generar un modelo destino.

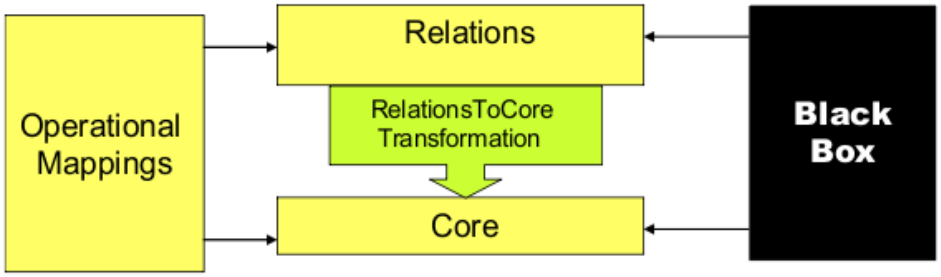


Figura 7: Arquitectura base de QVT (SmartQVT,2009)

Para tomar un ejemplo de QVT como lenguaje de transformación podríamos tomar la Figura 8: Ejemplo de transformación de modelos con QVT (SmartQVT,2009) donde se muestran dos metamodelos con sus componentes como BOOK que define un metamodelo para libros y otro para un PUB que define un metamodelo para una publicación.

```
metamodel BOOK {
  class Chapter;
  class Book {title: String; composes chapters: Sequence(Chapter);}
  class Chapter {title : String; nbPages : Integer;}
}

metamodel PUB {
  class Publication {title : String; nbPages : Integer;}
}

transformation Book2Publication(in bookModel:BOOK,out pubModel:PUB);

main() {
  bookModel.objectsOfType(Book)->map book_to_publication();
}

mapping Book::book_to_publication () : Publication {
  title := self.title;
  nbPages := self.chapters->nbPages->sum();
}
```

Figura 8: Ejemplo de transformación de modelos con QVT (SmartQVT,2009)

Luego de la definición de los metamodelos, solo se indica el modelo que debe entrar que en este caso debe ser un modelo de BOOK y el modelo que debe salir que es un PUB.

4.3.8 ATL (ATLAS *Transformation Language*) (ATL/UserGuide,2009) (ATLProject,2009)

Al igual que QVT, atlas es un lenguaje de transformación de modelos creado por ATLAS INRIA & LINA research group como respuesta al nuevo paradigma de programación MDA propuesto por OMG, donde su base para las transformaciones está compuesta por reglas que definen la comparación de los elementos del modelo origen hasta llevarlos al modelo destino, ya sea transformando los existentes y creando los que no existen.

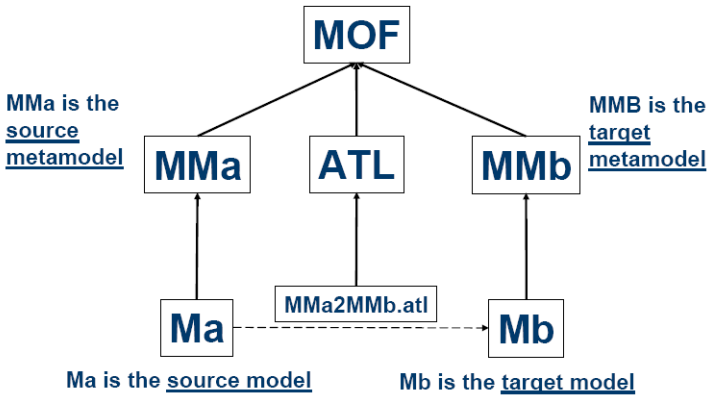


Figura 9: Diagrama de transformación ATL (ATLProject,2009)

```
package Families {

	class Family {
		attribute lastName : String;
		reference father container : Member oppositeOf familyFather;
		reference mother container : Member oppositeOf familyMother;
		reference sons[*] container : Member oppositeOf familySon;
		reference daughters[*] container : Member oppositeOf familyDaughter;
	}

	class Member {
		attribute firstName : String;
		reference familyFather[0-1] : Family oppositeOf father;
		reference familyMother[0-1] : Family oppositeOf mother;
		reference familySon[0-1] : Family oppositeOf sons;
		reference familyDaughter[0-1] : Family oppositeOf daughters;
	}

}

package PrimitiveTypes {
	datatype String;
}
```

Figura 10: Definición de modelos con ATL (ATL/UserGuide,2009)

Las reglas al ser el corazón de ATL son las encargadas de realizar la transformación por medio de dos restricciones, 1) Para que tipos de elementos origen, deben ser generado los elementos destino 2) la forma en que se deben inicializar los elementos destino, con el cumplimiento de estas restricciones es posible crear una regla capaz de realizar una transformación.

```
rule Member2Male {
      from
            s : Families!Member (not s.isFemale())
      to
            t : Persons!Male (
                  fullName <- s.firstName + ' ' + s.familyName
            )
```

Figura 11: Regla ATL (ATL/UserGuide,2009)

Parte importante del lenguaje ATL son los **helpers**, los cuales permiten la ejecución de código de partes diferentes del programa, así como lo hacen los métodos en el lenguaje Java

```
helper context Families!Member def: isFemale() : Boolean =
      if not self.familyMother.oclIsUndefined() then
            true
      else
            if not self.familyDaughter.oclIsUndefined() then
                  true
            else
                  false
            endif
      endif;
```

Figura 12: Helper ATL (ATL/UserGuide,2009)

4.3.9 M2M (M2M,2009)

Model to model o transformación de modelo a modelo, es un aspecto de el desarrollo de software que es guiado por modelos (MDD) se podría decir que es una implementación de QVT, lo que permite tener un framework para la IDE Eclipse, que permite transformar modelos a otros modelos cumpliendo con todas las reglas que pide el estándar para poder hacerlo, como es la definición de metamodelos y la creación de modelos a partir de los metamodelos con tres motores de transformación ya definidos con QVT, los cuáles son ATL, Procedual QVT y Declarative QVT.

4.3.10 M2T (M2T,2009)

Model to text o transformación de modelo a texto, es la ultima transformación que se puede realizar, ya que se pasa de un modelos de debe ser un PSM a código fuente o texto representativo para la plataforma especifica; M2T esta implementado en el IDE Eclipse, el cual tiene varios motores de transformación como JET y Acceleo entre otros, los cuáles permiten dependiendo de una plantilla UML2 pasar a código lo que se representa en el modelo UML.

4.3.11 CASE Tool (CASETOOL,2009)

Esta es una herramienta de ingeniería, la cual proporciona ayuda del computador para una labor determinada, generalmente se usan herramientas CASE para ingeniería de software.

La utilidad de estas herramientas es que el trabajo es en su mayoría realizado gráficamente, lo cual ahorra tiempo y conocimientos específicos de la tarea que el software realiza, este tipo de herramientas son drag and drop, donde se escoje un elemento, se arrastra hasta la posición que se desee y se deja allí para ser usado posteriormente.

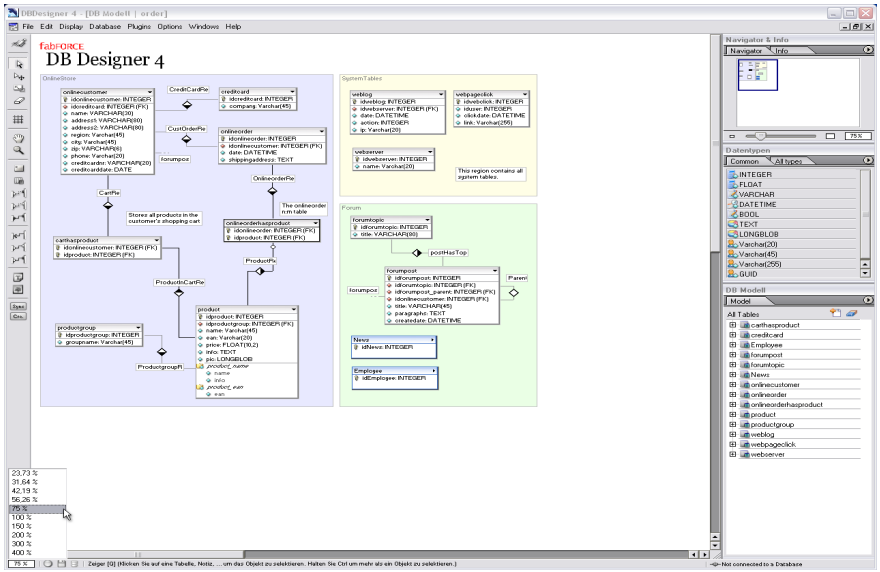


Figura 13: Herramienta CASE (CASETOOL,2009)

Uno de los aspectos más importantes de la herramienta tipo CASE es la capacidad de manipular los elementos del modelo, que se definen en el caso de una aplicación GMF como un metamodelo ecore.

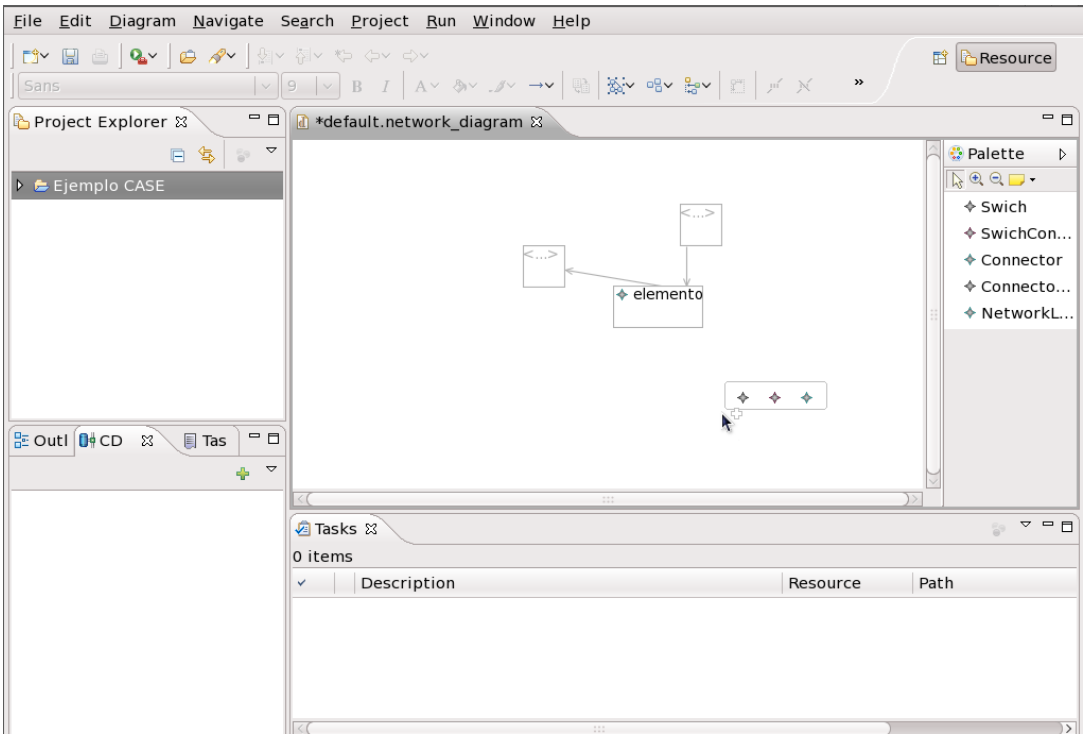


Figura 14: Ejemplo CASE desde eclipse

4.3.12 EMF (EMF,2009)

Es una framework desarrollado por la Eclipse Foundation para la creación de aplicaciones las cuáles generalmente son herramientas desde archivos XMI que tienen una representación del modelo, El framework EMF permite la creación de la aplicación final desde un editor gráfico, el cual es una representación del modelo o desde un editor de propiedades.

Este tipo de framework's permite dentro del proyecto hacer un desarrollo más estandarizado y más corto, en especial este permite realizar la lógica de la herramienta que permitirá crear en tiempo de ejecución las clases necesarias para hacer la representación del diagrama de líneas de

producción para poder trabajar sobre estas clases y hacer la transformación.

4.3.13 GMF (GMF,2009)

Este es un framework de edición gráfica desarrollado por la Eclipse Foundation, para la creación de herramientas y aplicación es que tengan la propiedad de hacer ediciones de forma gráfica de elementos que representan en el caso de este proyecto, una línea de producción, esta es la parte más importante dentro de la herramienta CASE.

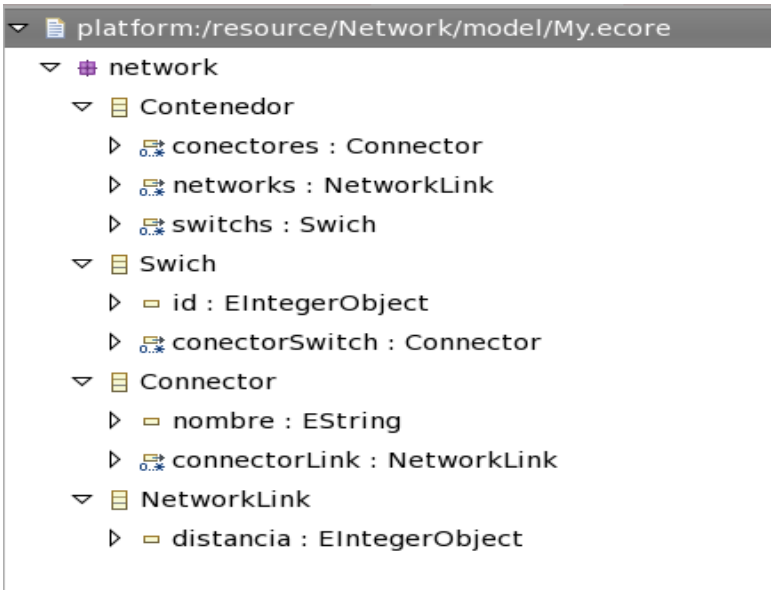


Figura 15: Definición de elementos desde GMF para la herramienta CASE

4.3.14 RCP (RCP Eclipsepedia,2009)(RCP,2009)

Rich Client Plataform, es la tecnología usada por el IDE eclipse para estandarizar los componentes usados para el desarrollo de aplicaciones, la cual se basa en reusar la arquitectura del IDE para crear un cliente personalizado que está conformado por un conjunto mínimo de plugin's con interfaz para el usuario.

Al lograr tener una aplicación basada en el IDE con al menos una interfaz para el usuario, es posible añadir otros plugin's que se deseen para

completar la herramienta, que funcionara independientemente al entorno de desarrollo pero de forma similar.

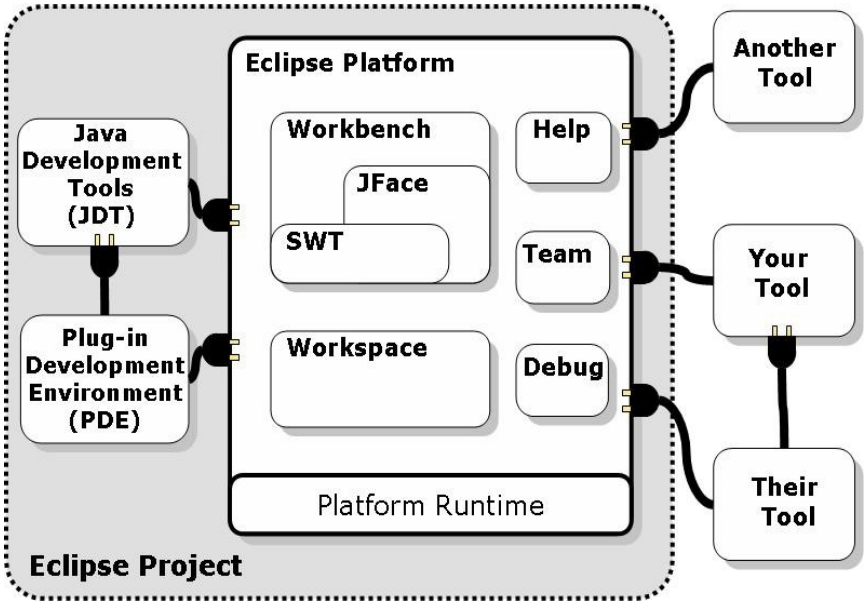


Figura 16: Arquitectura RCP(RCP,2009)

Ya que es una aplicación independiente de entorno de desarrollo integrado, debe funcionar por sí sola, esto conlleva a que sus componentes están conformados por:

- Núcleo.
- Framework de construcción estándar.
- Espacio de trabajo (editores, vistas, manejo de perspectivas, etc).
- Administrador de texto (editor, manejo de archivos).

4.3.15 Perspectivas

Las perspectivas son contenedores visuales que permiten agrupar componentes usados para el desarrollo o de la misma aplicación por medio de una de sus características, ayudando a aprovechar el espacio de trabajo, y permitiendo una obtención de los componentes mas fácilmente, ya que ordena el espacio de trabajo especialmente para una tarea en particular.

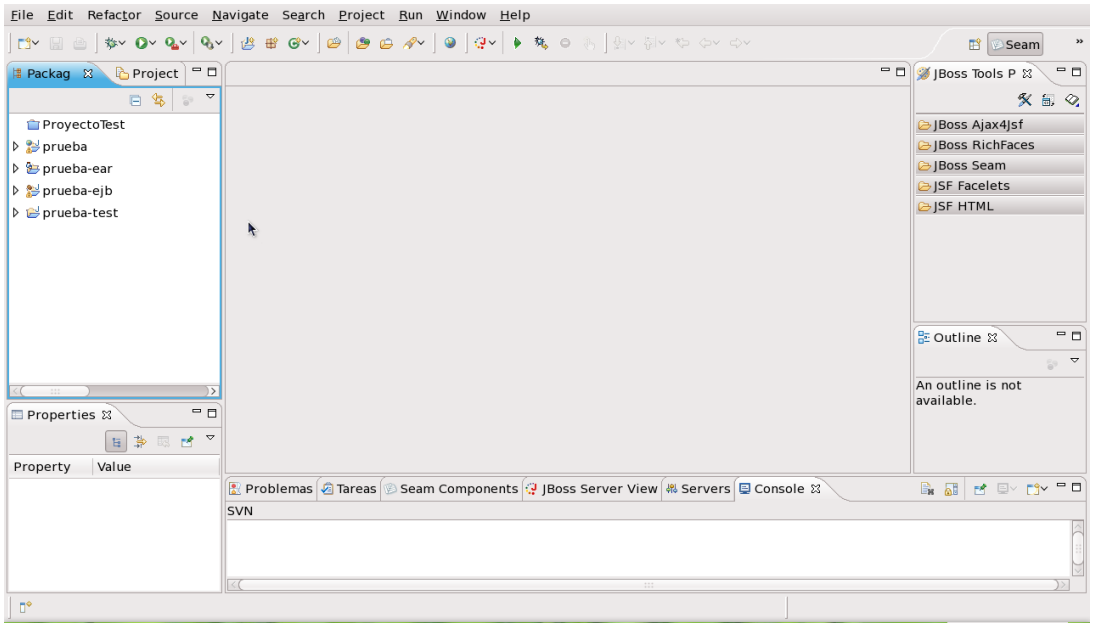


Figura 17: Perspectiva del IDE Eclipse para SEAM

4.3.16 Plugin(eclispePlugin,2009)

Los plugin's o complementos son componentes usados dentro del entorno de desarrollo integrado para aumentar la funcionalidad o añadir nuevas funcionalidades, estos están basados en los mismos componentes que hacen parte de un RCP, ya que un plugin puede añadir las mismas funcionalidades al IDE que tendría un RCP, la única diferencia es que el RCP puede ejecutarse sin la necesidad de el IDE mientras que el plugin si lo necesita para funcionar.

4.3.17 Control de Versionamiento (Collins-Sussman,2008)

Para poder manipular una versión del proyecto (código fuente, requerimientos,documentación, etc) en un momento del tiempo, hay que recurrir a un controlador de versiones, el cual llevara un control automático y la total administración de uno o varios archivos.

Gran parte de la funcionalidad del controlador de versiones es administrar los cambios hechos a un archivo, permitiendo que varios usuarios puedan manipular una versión local del archivo y que al momento de integrar los cambios de diferentes usuarios, estos se vean reflejados en la versión que se encuentra en el controlador de versiones.

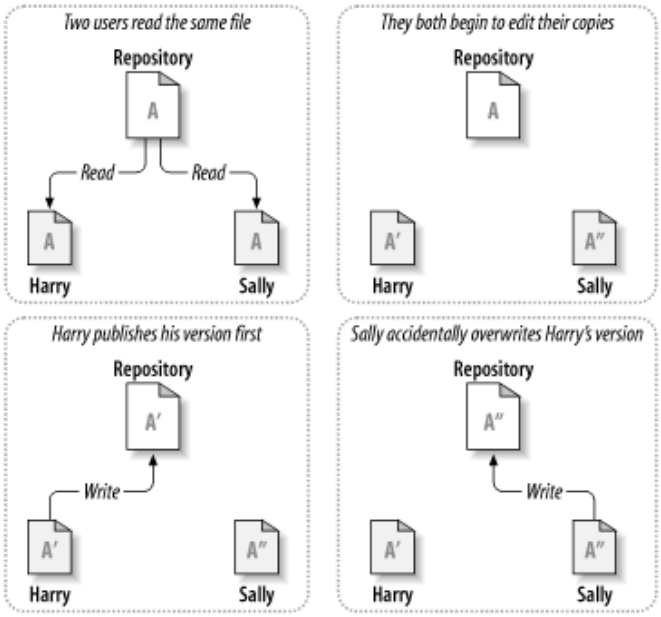


Figura 18: Manejo de múltiples usuarios con subversion(Collins-Sussman,2008)

Subversion es un controlador de versiones que en la actualidad es el mas usado dentro de los proyectos de software gracias a su administración de estados del repositorio, los cambios son tomados como atómicos y gracias a esto una versión en el repositorio es guardada solo como el cambio y no como el archivo completo.

4.3.18 Arquitectura MDA (OMG,2009) (Mellor,2004)

La guia propuesta por la arquitectura MDA, esta indica el uso de metamodelos, modelos y sus transformaciones como son CIM, PIM y PSM para la correcta utilización de esta tecnología con la cual se puede llegar a reemplazar el desarrollo tradicional de software, ya que la idea de este es reemplazar los diagramas hechos en el diseño y la codificación exclusiva en el desarrollo.

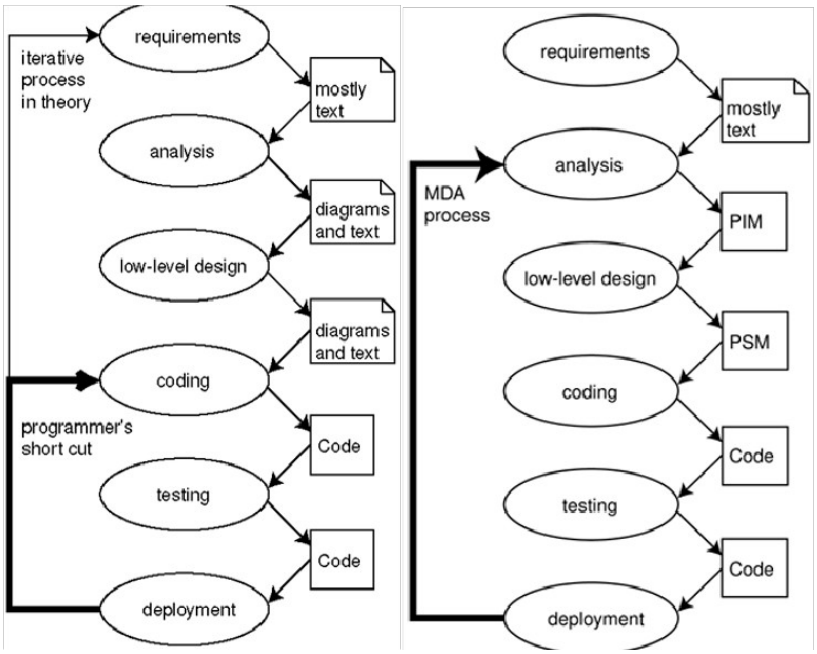


Figura 19: Desarrollo tradicional vs MDA (Mellor,2004)

4.3.19 CIM (Mellor,2004)(Miller,2003)

Modelo independiente de la computación, este es la versión más abstracta de un modelo dentro de MDA, ya que provee una representación del negocio totalmente independiente de cualquier herramienta o arquitectura usada dentro del diseño o desarrollo de aplicaciones de software. Es una representación en ocasiones dada por el interesado y si no es dada por el, es muy fácil de entender por personas que no están relacionadas con el campo de software.

4.3.20 PIM (Miller,2003)(Mellor,2004)

Modelo independiente de la plataforma, este es el modelo que sigue en nivel de refinamiento dentro de la arquitectura MDA después del modelo CIM el cual ya posee más información. Este modelo se puede tomar como un modelo genérico dentro del desarrollo, ya que no está conectado a ninguna arquitectura o tecnología en particular pero posee información más detallada y específica para el desarrollo de software como un diagrama de clases, la información dada por este tipo de modelos la puede entender una persona relacionada con la construcción de software pero no una persona a como los interesados.

4.3.21 PSM (Mellor,2004)(Miller,2003)

Modelo especifico para la plataforma, este modelo es el más refinado de todos, ya que expresa la información suministrada por uno o más PIM's a una arquitectura o tecnología específica, haciendo que sirva únicamente para esta, se podría decir que un PSM es un PIM con la adición de información especifica para la plataforma que se usara para el desarrollo.

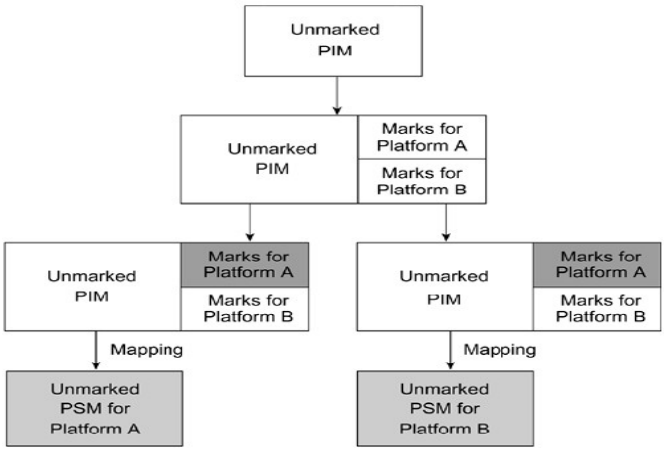


Figura 20: Transformación de PIM a PSM (ModelWare,2009)

4.3.22 XMI

XML Metadata Interchange, es un estándar creado por la OMG para la representación de modelos y poder utilizar este archivo en diferentes herramientas para diseño de software a través de XML, donde se añade metainformación del modelo para representarlo de una forma genérica para un uso correcto de UML, además de ser el estándar entre herramientas para el diseño UML es también el estándar para entornos distribuidos.

XMI al permitir añadir no solo la información del modelo sino la metainformación de este, es el formato adecuado para la manipulación de las diferentes versiones de modelos descritos por MDA y de su nivel de abstracción ayudando a la tarea de intercambiar información en el caso del proyecto, entre la herramienta CASE y el optimizador de modelos matemáticos al momento de enviar la información del diagrama de líneas de producción hecho por la herramienta CASE al optimizador y cuando se quiere obtener la información ya optimizada para que sea mostrada gráficamente al usuario.

4.3.23 MOF (JOT,2009) (Jablonski,2009)

Meta Object Facility, es el estándar propuesto por OMG para creación de lenguajes de modelamiento o metamodelos como UML, se puede decir que MOF es un meta-metamodelo, con el cual se definen y crean metamodelos y también meta-metamodelos ya que este se contiene a sí mismo como aparece en la Figura 21: Arquitectura MOF (JOT,2009).

Este estándar es muy importante al momento de crear metamodelos, ya que nos indica las reglas y elementos que deben ser tenidos en cuenta.

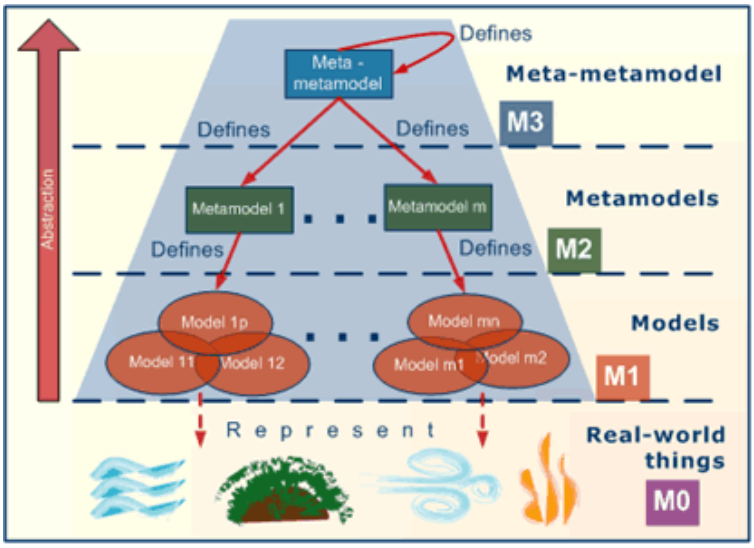


Figura 21: Arquitectura MOF (JOT,2009)

MOF forma parte esencial del proyecto ya que es el corazón de MDA, a partir de MOF se definirá los metamodelos empleados para la representación de líneas de producción y uno diferente para la representación de modelos matemáticos.

4.3.24 Acceleo

Acceleo es una herramienta para la generación de código fuente a partir de modelos desarrollados bajo el estándar propuesto por MDA. Acceleo es la herramienta usada para transformar a partir de un PIM (Modelo), una plantilla especifica del la plataforma para dar como resultado código fuente de la plantilla escogida permitiendo transformaciones M2T(Modelo a texto).

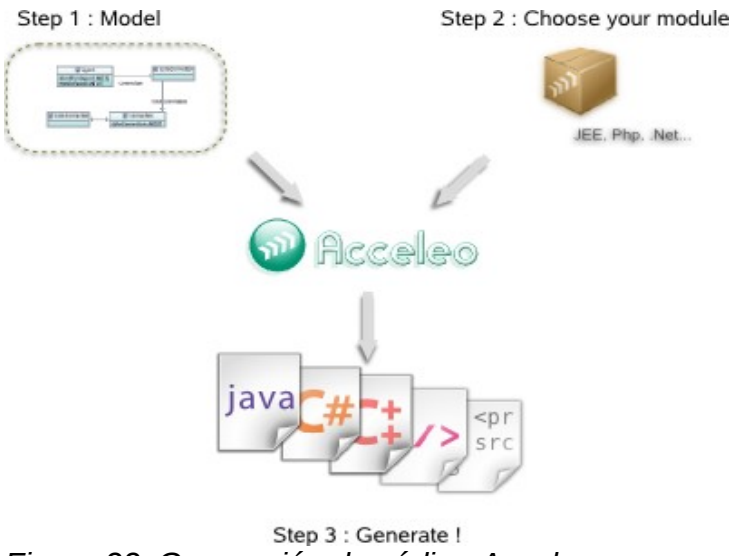


Figura 22: Generación de código Acceleo

Acceleo está basado en el uso de plantillas, las cuales recurren al API de acceleo para la generación de código, esta plantillas tienen un lenguaje en particular definido por acceleo, el cual es el que permite la identificación de la sintaxis particular, las plantillas generalmente son archivos .mt que son asociados a un modelo y desde la información de este, realiza lo expuesto en el lenguaje acceleo que hay en la plantilla.

```
<%
metamodel http://www.eclipse.org/uml2/2.0.0/UML
import utilidades.ClassesServices
%>

<%script type="uml.Class" name="Clases" file="<%name%>.java"%>

package chains;

public class <%name%> {

    <%for (attribute){%>
        private <%type.name%>  <%name%>;
    <%}%>


    <%for (attribute){%>
        public  <%type.name%> get<%name%>(){ return this.<%name%>;}
        public void set<%name%>(<%type.name%> <%name%> ){this.<%name%>=<%name%>;}
    <%}%>


    public <%name%> (<%for (attribute){%><%if (i()==0){%><%type.name%> <%name%><%}else{%>, <%type.name%> <%name%><%}%> <%}%>){
    <%for (attribute){%>this.<%name%>=<%name%>;<%}%>

    }


    <%if (hasStereotype("switch")){%>
        public void insertar(Conector con){
            this.<%for (attribute){%><%if (type.name.equalsIgnoreCase("ArrayList")){%><%name%><%}%><%}%>.add(con);
        }
```

Figura 23: Ejemplo de plantilla Acceleo

Como se puede notar en la Figura 23: Ejemplo de plantilla Acceleo el lenguaje de acceleo esta rodeado por los caracteres <% %> que indican que lo que hay dentro de ellos debe ser ejecutado usando datos del modelo como “type” , “attribute” y “name” entre otros.

4.3.25 Modelos Matemáticos (Aris,1994)

Un modelo matemático se puede describir como un lenguaje que permite hacer una representación abstracta de la realidad para observar su comportamiento en diferentes situaciones planteadas por la combinación de sus partes, las cuales son:

- **Variables**: Son las que representan la parte del modelo que irán cambiando por la toma de decisiones y que al final cambiaran el valor de la función objetivo, por ejemplo en el caso de una línea de producción podría ser “la cantidad de materia prima”.

- **Restricciones**: Las restricciones son las forman conjuntos y tienen la propiedad de acotar el valor de las variables, por ejemplo en una línea de producción seria “El número mínimo de máquinas”

- **Índices**: Son valores que indican una situación determinada dentro del sistema, generalmente no varían son definidos antes de empezar a resolver la ecuación, por ejemplo “El tiempo”.

- **Ecuaciones (Generalmente es una y es llamada función objetivo)**: Es la que nos permitirá ver el funcionamiento desde cierta perspectiva de un sistema, esta función esta compuesta por los elementos anteriores, por ejemplo en una línea de producción seria “La maximización de unidades de producto al finalizar la línea de producción”.

Al momento de definir un modelo matemático es importante la definición de las variables, las cuales representan por ejemplo el número de de máquinas en una línea de producción o el tiempo que tarda una máquina en realizar un proceso, estas variables están asociadas a una ecuación, el cual describe el funcionamiento, pero con ciertas restricciones.

En la Ecuación 4: Restricciones al modelo matemático (Linares,2001) se puede observar a las variables la variable *y* con su incide *i.*

$$x = \sum_{i=0}^{N} 2^i y_i$$

Ecuación 3: Función Objetivo (Linares,2001)

$$0 \leqslant x \leq u$$
$$2^N \leqslant u \leqslant 2^{(n+1)}$$

Ecuación 4: Restricciones al modelo matemático (Linares,2001)

5 ESTADO DEL ARTE

5.1 PROMODEL (PROMODEL,2009) (MORAGAS,S.F)

La compañía Promodel fundada en 1988 centra su industria en la optimización de procesos de negocio especialmente en desarrollos farmacéuticos, de salud y manufactura, la herramienta homónima con la capacidad de optimizar y modelar procesos de manufactura está basada en la tecnología VAO, permitiendo visualizar, analizar y optimizar el proceso a lo largo de la simulación y experimentando con escenarios virtuales.

5.1.1 VAO (MORAGAS,S.F)

Es la tecnología usada por las herramientas de ProModel para la construcción de herramientas más eficaces al momento de realizar las simulaciones pensando en múltiples hipótesis como variabilidad de datos, contención de recursos e independencias complejas; sus siglas vienen de visualize, analize y optimize.

Visualize se enfoca en tomar decisiones por medio de la visualización de los datos que están siendo simulados en tiempo real. Analize está por otra parte mira el impacto de los cambios realizados a lo largo de la simulación, mientras que Optimize se basa en la posibilidad del usuario para tomar decisiones mientras la simulación se está ejecutando.

Figura 24: Simulación ProModel (Promodel,2009)

ProModel es una herramienta cuyo campo de soluciones de manufactura es capaz de dar solución a líneas de producción como se muestra en la simulación de la Figura 24: Simulación ProModel (Promodel,2009). Al momento de crear el modelo para la simulación con la herramienta es muy similar a la creación de un modelo matemático, ya que los datos de entrada son similares a los de se necesitan para aplicar un modelo matemático estos son:

- Generalidades: se define por la existencia de dos reglas que cada elemento creado dentro de la simulación debe cumplir, las cuales son:
  - 1)Los elementos deben estar correctamente definidos antes de la ejecución de la simulación.
  - 2) Dentro del modelo deben existir como mínimo localizaciones, entidades, arribos y el proceso para poder realizar la simulación.

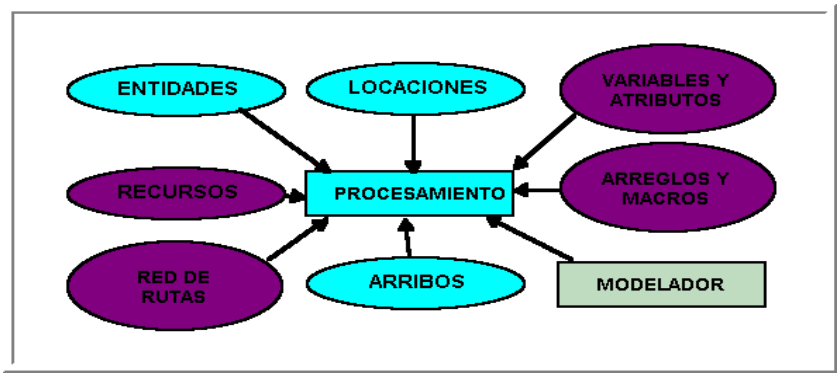


Figura 25: Elementos de la herramienta ProModel (Promodel,2009)

- Locaciones: Estas están dadas por la posición fija dentro del modelo en el que se dirigen las entidades para ser procesadas, las cuales son editables de forma gráfica.
- Entidades: Se llama de esta forma a todos los elementos dentro del modelo que son procesados como la materia prima necesaria, loes elementos de papeleo o productos en general; este también tiene un editor gráfico para este propósito.
- Path Networks: Estas son las rutas que existen dentro del modelo por los cuales se transportan las entidades de las cuales hay tres tipos, el primer tipo se refiere a las rutas donde las entidades pueden ser intercambiadas de orden o ruta, el siguiente tipo de ruta es la que no permite que las entidades cambien de orden o de ruta y por ultimo existen las rutas que son las que representan grúas.
- Recursos: Estos son los elementos que recolectan entidades para realizan una operación especifica con ellas, existen dos tipos de recursos, los dinámicos que pueden moverse a lo largo de la simulación como los operarios para reparar elementos dañados o escoger cierto producto y los estáticos que permanecen quitos y solo acceden a las entidades que llega a ellos por medio de las rutas.
- Proceso: Es el encargado de definir las conexiones entre las rutas y los recursos para realizar operaciones a las entidades a lo largo de la simulación con los que cambia el estado de la entidad como una cortadora y que serán representados en la herramienta como diagramas de procesos.
- Arribos: Son las nuevas entidades que entran al proceso a lo largo de la simulación, como materiales, personas, información, etc y su edición es en forma de tabla.

- Atributos: son características de los elementos que hacen parte del modelo a los cuales se les puede asignar un valor, los cuales generalmente son los que se optimizan.
- Variables: Como su nombre lo indica son valores que varían a lo largo de la simulación, estas pueden ser de dos tipos, globales y locales y son útiles para almacenar información de los cambios que van surgiendo en un atributo a lo largo de la simulación.
- Lógica: Son comandos definidos por la herramienta para suministrar lógica dentro del proceso, como restricciones.

Un ejemplo para la aplicación de la herramienta es la simulación de una línea de producción de tacos la cual quería analizar porque la producción de tacos no era capaz de satisfacer la demanda de los clientes y querían encontrar una solución factible dentro de los recursos que existían en ese momento.

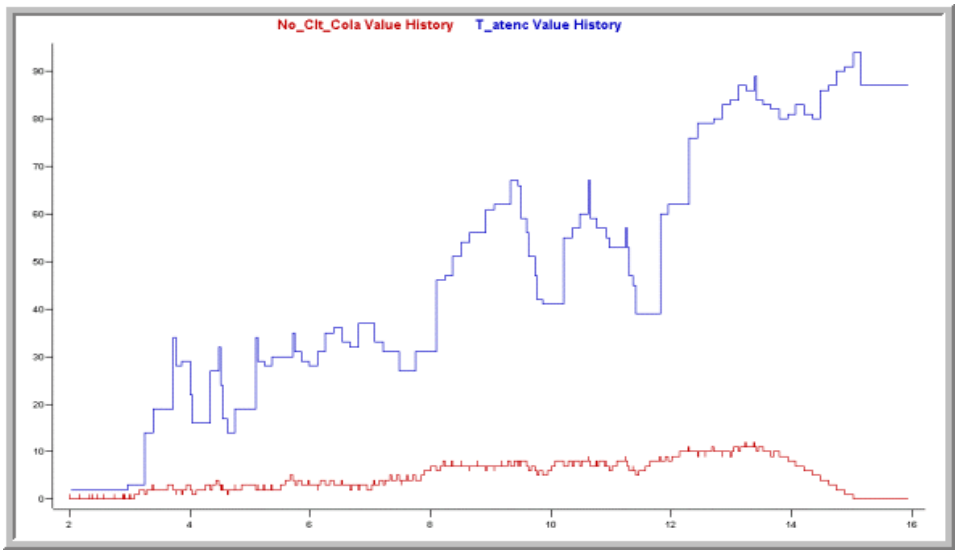


Figura 27: Resultados se simulación de la línea de producción de tacos (tiempo de atención, número de clientes en cola) (Arroyo,2009)

5.2 Flexible Line Balancing

Flexible line balancing es una herramienta creada en conjunto por LG Electrinics y multinacionales de manufactura con el propósito de mejorar las líneas de producción balanceando su carga de procesos dependiendo de unas restricciones definidas por el usuario.

La herramienta para el balanceo de líneas de producción usa el procedimiento COMSOAL para encontrar una solución al problema que se propone, que en este caso es el balanceo de una línea de producción.

5.2.1 COMSOAL (DePuy,2000)

Este es un algoritmo heurístico que proviene de las siglas del ingles “The computer method of sequencing operations for assembly lines”, el cual fue creado por Argus.

Este método genera un gran número de soluciones a partir de simulaciones que se basan en dos condiciones 1) Todos sus predecesores inmediatos ya esta asignados y 2) El tiempo de procesamiento es menor o igual al UACT (Unassigned cycle time); estas condiciones son aplicadas a una lista construida por este en la que se encuentran los procesos que están disponibles para ser programados.

Para dar solución al balanceo de líneas de producción el algoritmo elige el proceso siguiente aleatoriamente que cumpla con las condiciones mencionadas anteriormente con esto el algoritmo es capaz de resuelve el problema de asignación de recursos en cada iteración de la heurística gracias al empleo de Meta-RaPS el cual usa reglas de prioridad, aleatoriedad y Muestreo.

Como herramienta de balanceo de líneas de producción flexible line balancing es muy poderosa al momento de realizar análisis y simulaciones con ella; al momento de realizar el balanceo, esta pide los datos iniciales y dibuja de manera inmediata un diagrama de precedencia como el de la Figura 28: Diagrama de una línea de producción realizado por la herramienta Flexible Line Balancing (DePuy,2000)

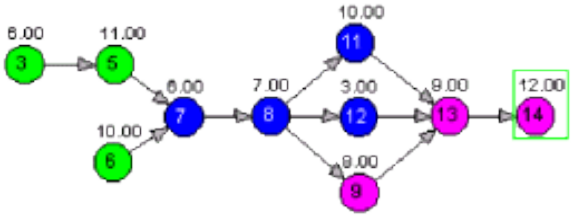


Figura 28: Diagrama de una línea de producción realizado por la herramienta Flexible Line Balancing (DePuy,2000)

después de esto la herramienta hace el balanceo y cambia de color los nodos que deben ser alterados y muestra los resultados de forma gráfica estadística para visualizar como se comporto la línea de producción mientras se hacían los cambios y que ofrece el análisis “What if?” el cual permite al usuario arrastrar con el mouse los elementos analizados dentro de la gráfica de resultado y mostrando como tendría que ser la línea de producción para dar ese resultado.

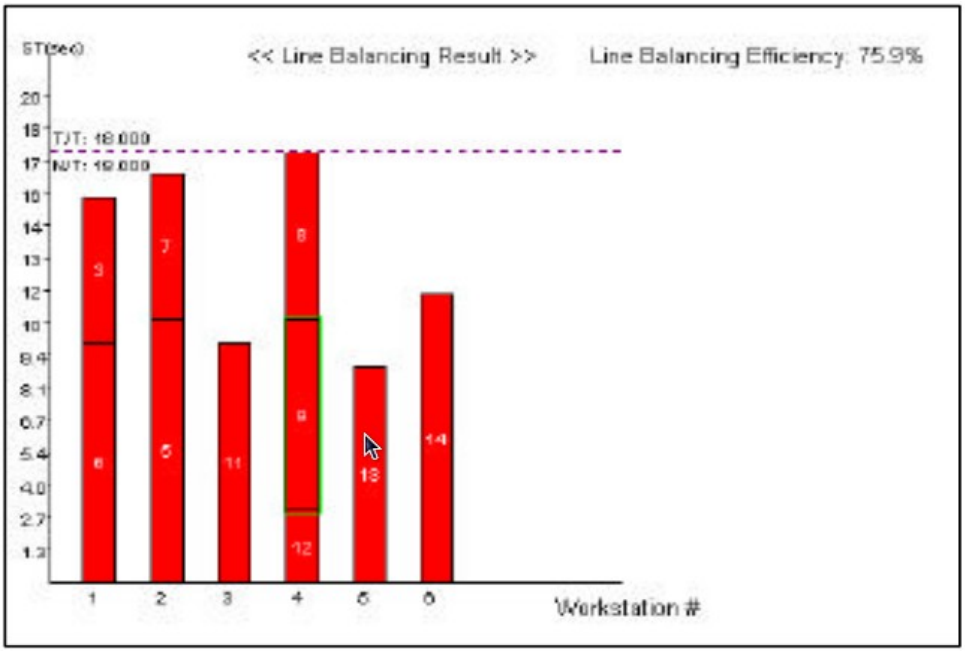


Figura 29: Resultado del balanceo de una línea de producción por la herramienta Flexible Line Balancing (DePuy,2000)

6 LIMITACIONES Y ALCANCES

El alcance de este proyecto está definido por el objetivo general y los objetivos específicos que están mencionados en este mismo documento, el alcance real de este documento está basado en la realización de una herramienta de software tipo CASE que permitirá realizar un diagrama de una línea de producción de la forma más natural posible tan solo arrastrando sus componentes a un área de trabajo dentro de la herramienta, para después tomar la información que se ha recolectado de esta forma y convertirla en un modelo basado en un metamodelo de líneas de producción realizado con las tecnologías mencionadas en el marco

teórico, después de esto el modelo tendrá que ser transformado por medio de marcas y mapeos para que concuerde con el metamodelo de modelos matemáticos, el cual tiene que ser transformado a código java de forma que sea la entrada de información para la herramienta AMM independiente a este proyecto que tendrá la función de optimizarlo, para luego retornar en el resultado de la optimización la cual debe ser mostrada a el usuario por un visualizador

Que no se realizara en este proyecto:

- No se diagramaran cosas diferentes a una línea de producción.
- No se optimizara el modelo de línea de producción, para ello se hara uso de una herramienta externa.
- Se usara un único método matemático para la representación de la línea de producción.
- Se transformara a código java únicamente.

Por otra parte, las limitaciones que afronta este proyecto están dadas por.

- El no conocimiento de las tecnologías a usar, lo cual puede generar atrasos, ya que puede existir una herramienta más adecuada.
- La limitación de tiempo, en el cual se debe ser muy estricto con el cronograma para cumplir las metas en las fechas estimadas.
- La generación de código se limitara al lenguaje de programación Java.
- Para la creación del modelo, solo se podrán modelar y optimizar modelos compatibles con el método matemático de Flexible Job Shop.

7 DISEÑO METODOLÓGICO (OPENUP NUTSHELL,2009)(OPENUP,2009)

La metodología que sera usada en este proyecto, es OpenUP “Open Unified Process”, que esta basado en micro incrementos personales que deben ser cumplidos por roles específicos, que al final harán parte de un incremento mayor llamado iteración cuyo objetivo es mostrar al cliente un “demo” del estado del proyecto, de esta forma se irá construyendo el proyecto a lo largo de sus cuatro etapas.

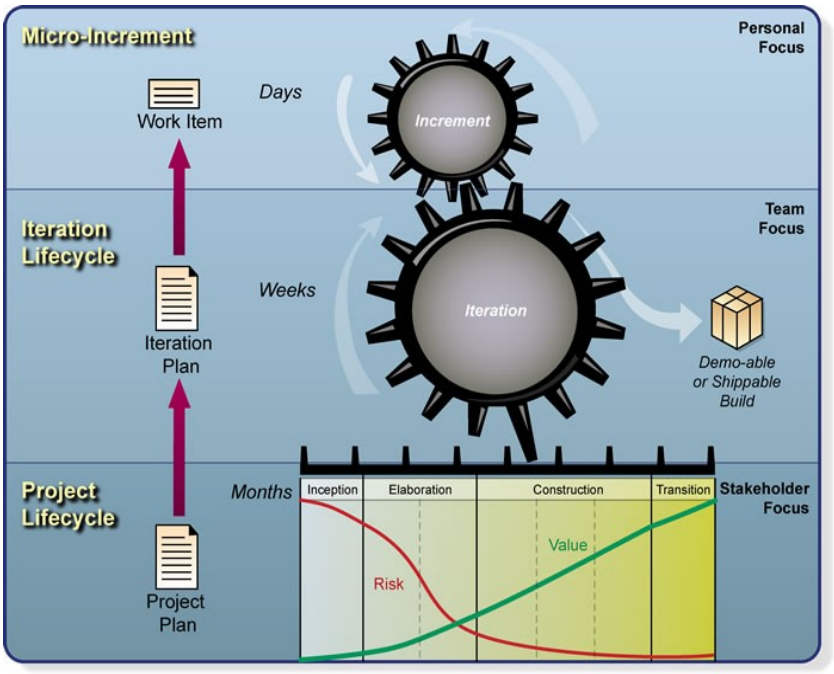


Figura 30: Capas de la metodología OpenUP (OpenUP,2009)

Otra parte de la metodología usada sera una parte de RUP (Rational Unified Process) que una de las más solidas para el desarrollo de proyectos de software y gracias a que OpenUP tiene las mismas características de RUP como un desarrollo iterativo, casos de uso y un enfoque centrado en la arquitectura entre otros, fue elegido para este proyecto la versión de OpenUP ya que solo se realizara lo que se requiera en el proyecto cumpliendo con las practicas propuestas:

- Desarrollo iterativo.

- Colaboración del grupo de trabajo.
- Integración y pruebas continuas.
- Entregables de software frecuentes.
- Adaptación a los cambios.

La metodología también propone la implantación de varios roles para el desarrollo del proyecto que trabajaran en un equipo y en el cual cada uno tiene responsabilidades y “Work products” o entregables del desarrollo como parte del proyecto, estos roles son:

- Analista: Es el encargado de dar a conocer los requerimientos del cliente y reunirse con el para acordarlos y darles prioridad.

Figura 31: Work products y responsabilidades del analista (OpenUP,2009)

- Arquitecto: Es el responsable de la arquitectura del software, tanto de la documentación como el diseño tomando decisiones como las tecnologías a usar y la forma de usarlas.

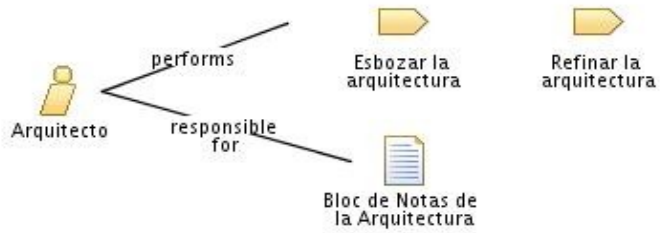


Figura 32: Work products y responsabilidades del arquitecto(OpenUP,2009)

- Desarrollador: El desarrollador es el responsable de codificar el sistema haciendo que sea acorde con lo diseñado por el arquitecto, también es el encargado de realizar las pruebas unitarias y prototipos para mostrar al cliente.

Figura 33: Work products y responsabilidades del desarrollador(OpenUP,2009)

- Administrador del proyecto(Líder): El director del proyecto tiene como función principal la planeación de este, también la coordinación con los interesados.

Figura 34: Work products y reponsabilidades del administrador del proyecto(OpenUP,2009)

- Stakeholder: Estas son las personas que están interesadas en el proyecto y pueden dar aportes para este y la realización lo afectara de alguna manera.

- Encargado de Pruebas (Tester): Uno de los roles mas importantes ya que comprueban la calidad del desarrollo, bajo su cargo se encuentran responsabilidades como identificar, definir, implementar y la realización de los test.

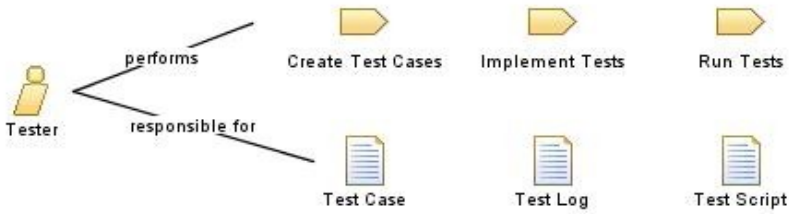


Figura 35: Work products y responsabilidades del tester (OpenUP,2009)

- Any Role: Es el encargado de realizar cualquier tarea que se le asigne.

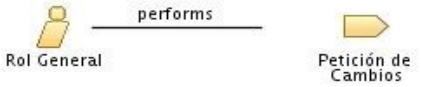


Figura 36: Work products y responsabilidades de any role (OpenUP,2009)

Los roles dentro del desarrollo del proyecto serán realizados por por dos personas, el asesor y el asesorado que estarán involucrados en cada uno de los roles mencionados anteriormente mas los interesados que solo tendrán el rol de stakeholder que proporcionaran información.

Las iteraciones como uno de los pilares de la metodología OpenUP ayudan a mantener a los stakeholders involucrados en el proyecto ya que generan un demo en cada finalización de la iteración donde cada “work product”

debe encajar como parte de los micro-incrementos que componen la iteración y que debe cumplir con un ciclo de vida.

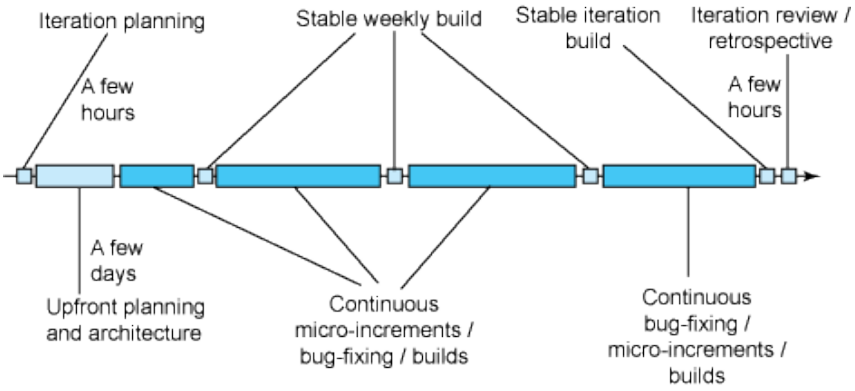


Figura 37: Ciclo de vida de una iteración (OpenUP Nutshell,2009)

El proyecto de dividirá en micro-incrementos que son paquetes compuestos por documentos y/o pequeños fragmentos de código que no duraran mas de un par de días en su realización y estos a su vez formaran parte de las iteraciones que se definirán por el numero de “demos” que se entregaran al cliente, estas iteraciones están comprendidas en las fases propuestas por la metodología y mencionadas en detalle a continuación.

Las etapas propuestas por la metodología OpenUP son cuatro y se aplicaran de la siguiente forma en el proyecto:

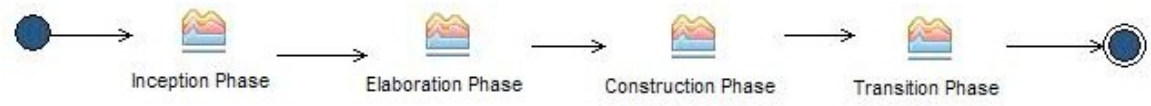


Figura 38: Ciclo de vida OpenUP (OpenUP,2009)

7.1 Initiate Project (Iniciación del Proyecto)

Como su nombre lo indica, esta es la fase en la que inicia el proyecto y se contempla la duración y el alcance en un nivel global así mismo el plan de desarrollo del proyecto. El primer ciclo dentro de la vida del proyecto es

realizar toda la planeación, la captura de información a los interesados y la documentación requerida como la visión, esto esta contemplado realizarse en una iteración que tendrá los micro-incrementos compuestos por documentos hechos a lo largo del proceso de seminario de proyecto de grado de esta forma.

ITERACION 1	
Micro-incrementos	**Fecha Esperada de Entrega**
M1	**10/08/09**
Documento de petición de Interesados	06/08/09
Documento de Lista Riesgos	08/08/09
Documento de Plan de Gestión de Riesgos	10/08/09
M2	**19/08/09**
Documento de Gestión de configuración	13/08/09
Documento de Plan de Gestión de Requerimientos	15/08/09
Documento de Requerimientos no Funcionales	17/08/09
M3	**26/08/09**
Documento de Visión	22/08/09
Documento Plan de Desarrollo de Software	26/08/09
M4	**02/09/09**

Glosario	31/08/09
Documento de solicitud de cambios	02/09/09
Plan de resolución de pruebas	10/08/09
M5	**16/09/09**
Documento de Anteproyecto	16/09/09

Tabla 1: Micro-incrementos fase de iniciación

7.2 Elaboration Phase (Fase de Elaboración)

En la fase de elaboración se empiezan a trabajar los requerimientos obtenidos de los interesados, sacando a partir del análisis los requerimientos funcionales y los no funcionales así como los documentos de casos de uso y casos de prueba para empezar con el diseño. En esta etapa se hará uso de una sola iteración que contendrá micro-incrementos como la documentación de diseño y los diagramas.

ITERACION 2	
Micro-incrementos	**Fecha Esperada de Entrega**
M1	**20/10/09**
Diagramas de Secuencia	01/10/09
Diagrama de Clases	16/10/09
Documento de Aseguramiento de Calidad	20/10/09
M2	**31/10/09**
Documento de Arquitectura	26/10/09
Documento de especificación de requerimientos	31/10/09
Plan maestro de pruebas	05/11/09
Plan de aseguramiento de calidad	07/11/09

Tabla 2: Micro-incrementos fase de Elaboración

7.3 Construction Phase (Fase de construcción)

Dentro de la fase de construcción se realiza todo lo relacionado a la codificación de la solución plateada dentro del diseño realizado en la fase de elaboración, para ello se hará uso de cuatro iteraciones las cuales generaran demos del estado del proyecto en su 30%, 60% 100% y uno mas

que sera usado inicialmente para preparación de los metamodelos y prerequisitos de las tecnologías.

ITERACIO 3	
Micro-incrementos	**Fecha Esperada de Entrega**
M1	**09/11/09**
Plugin estándar para eclipse en el cual se cambian las perspectivas	04/11/09
Rcp basado en eclipse en el cual se cambian las perspectivas	09/11/09
M2	**12/11/09**
Documento plan de aceptación del producto	12/11/09
M3	**22/11/09**
Metamodelo de líneas de producción	17/11/09
Metamodelo de modelos matemáticos	22/11/09
M4	**08/12/09**
Mapeos para la transformación	30/11/09
Marcas para la transformación	08/12/09
M5	**18/12/09**
Pruebas de modelos basados en el	13/12/09

metamodelo de líneas de producción.	
Pruebas de modelos basados en el metamodelo de modelos matemáticos	18/12/09

Tabla 3: Micro-incrementos para la iteración 1 de la fase de construcción

ITERACIO 4	
Micro-incrementos	**Fecha Esperada de Entrega**
M1	**07/01/10**
RCP para diagramación tipo CASE de líneas de producción	02/01/10
Pruebas de la herramienta RCP para líneas de producción.	07/01/10
M2	**27/01/10**
Transformación de modelo de líneas de producción a modelo matemático	22/01/10
Pruebas de transformación entre modelos.	27/01/10
M3	**20/02/10**
XMI para representación de modelos	04/02/10
Componentes para captura de información a partir de archivos	12/02/10
Componentes para captura de información a partir de base de datos	20/02/10

Tabla 4: Micro-incrementos para la iteración 2 de la fase de construcción

ITERACION 5	
Micro-incrementos	**Fecha Esperada de Entrega**
M1	**28/02/10**
Integración para transformaciones M2M con la herramienta CASE	25/02/10
Pruebas de transformación M2M a partir de la herramienta CASE	28/02/10
M2	**12/03/10**
Transformación M2T desde el modelo matemático haciendo uso de acceleo	05/03/10
Integración de M2T con la herramienta CASE	09/03/10
Pruebas de las transformaciones M2T haciendo uso de la herramienta CASE	12/03/10
M3	**21/03/10**
Generación de código fuente haciendo uso de acceleo y/o velocity	17/03/10
Pruebas de generación de código fuente haciendo uso de acceleo y/o velocity	21/03/10

Tabla 5: Micro-incrementos para la iteración 3 de la fase de construcción

ITERACION 6	
Micro-incrementos	**Fecha Esperada de Entrega**
M1	**30/03/10**
Componente para generar la base de código fuente que permita a otro programa su compilación y ejecución automática	26/03/10
Pruebas de compilación y ejecución automática de código fuente.	30/03/10
M2	**09/04/10**
Generación de código java para el uso de la herramienta AMM a través de su interfaz para la integración con la herramienta CASE	04/04/10
Captura e interpretación de resultados de la herramienta AMM.	09/04/10
M3	**17/04/10**
Visualizador de resultados para la respuesta de la herramienta AMM	14/04/10
Pruebas de la visualización de los resultado de la optimización	17/04/10
M4	**19/04/10**
Documento de ejecución de pruebas	19/04/10

Tabla 6: Micro-incrementos para la iteración 4 de la fase de construcción

7.4 Transition Phase (Fase de Transición)

En la fase de transición del proyecto, se realizara todo lo relacionado con la integración y entrega del producto final, los manuales de entrega para el usuario y documentación relacionada, para esta fase se hará uso de una sola iteración.

ITERACION 7	
Micro-incrementos	**Fecha Esperada de Entrega**
M1	**24/04/10**
Manual Técnico	24/04/10
M2	**29/04/10**
Manual de Usuario	29/04/10
M3	**14/05/10**
Monografía	14/05/10

Tabla 7: Micro-incrementos de la fase de transición

8 RECURSOS

Los recursos que serán usados dentro del proyecto se encuentran divididos en varios tipos, los recursos Humanos que son los relacionados a las personas que participaran de una manera activa dentro del proyecto, los recursos tecnológicos que involucran todo el aspecto de máquinas usadas para la realización de los entregables, recursos económicos que son los relacionados a el dinero invertido dentro del proyecto, recursos físicos, como el costo de la utilización de una oficina y sus servicios básicos, también los gasto de transporte están incluidos dentro de los recursos utilizados.

8.1 Recursos Humanos

Recurso	**Descripción**	**Costo**
Asesor Técnico	Andres Felipe Solarte	NA
Asesor Metodológico	Inez	NA
Asesorado	Jorge Iván Ramírez	NA
Director del Proyecto	Fabián Girando	NA

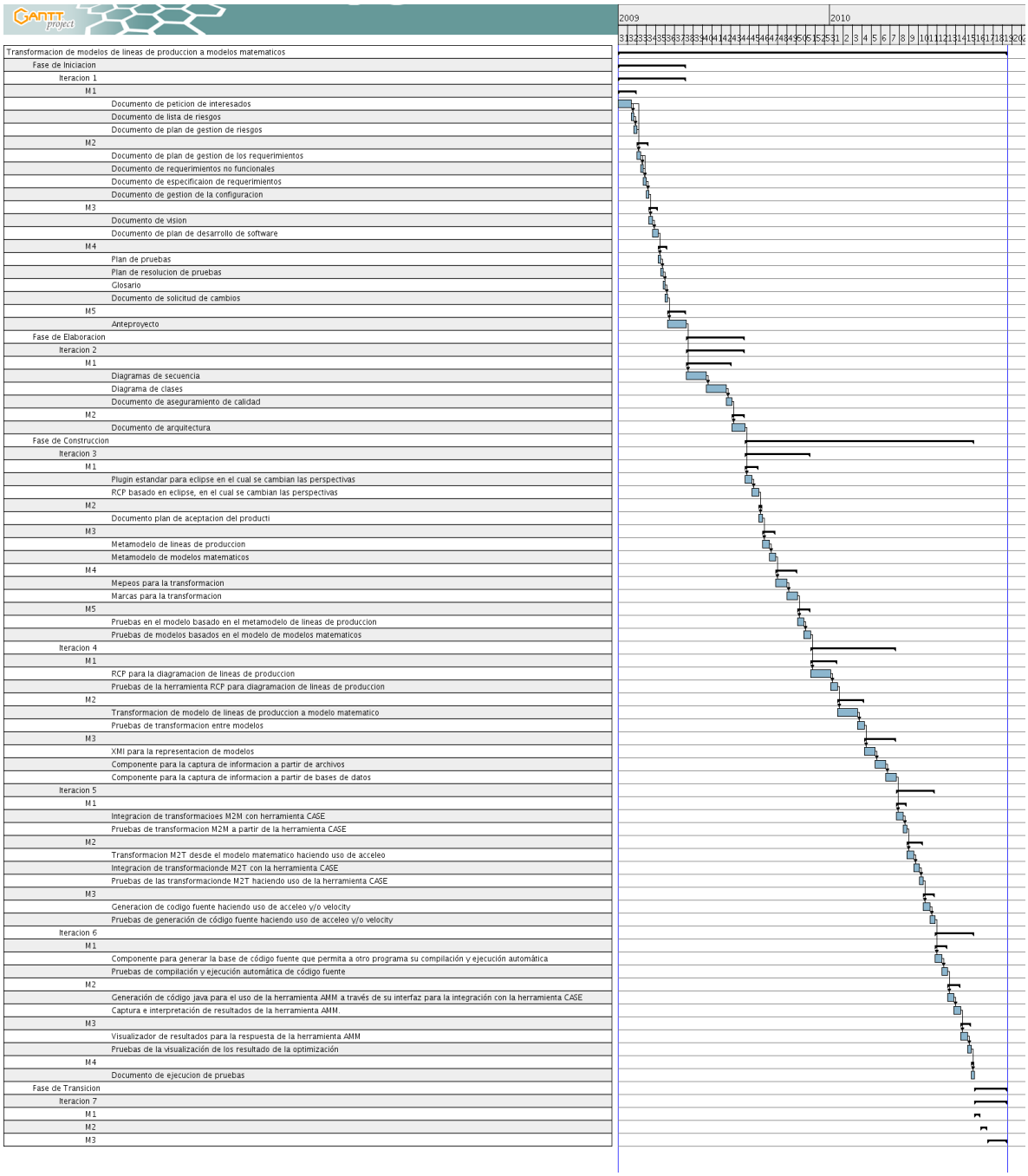
8.2 Recursos Tecnológicos

Recurso	**Descripción**	**Costo**
Computador Portátil	HP dv2000, 3Gb RAM, 250Gb DD	COP $ 1'900.000
Impresora	Epson	COP $ 200.000
Computador de Escritorio	Genérico 250Mb RAM, 120Gb DD	COP $ 1'000.000

8.3 Recursos de Oficina

Recurso	**Descripción**	**Costo**
Impresiones	Documentos que requieren ser presentados en papel	COP $100.000

9 CRONOGRAMA



GLOSARIO

- **MODELAMIENTO**

Abstracción: Nivel de abstracción se refiere a la cantidad de información del modelo, entre menos información, mas abstracto es el modelo.

ATL: Atlas Transformation Language, es el lenguaje definido para realizar transformaciones entre modelos que cumplan con los estandartes definidos por MDA.

CIM: Computer Independent Model, modelo independiente de la computación es un modelo de alto nivel de abstracción con la capacidad de ser entendido fácilmente.

Ecore:Es un metamodelo para la descripción de modelos EMF que es capas de dar soporte en tiempo de ejecución.

EMF: Eclipse Modeling Framework, es un framework para el modelamiento y generación de código basado en un modelo de datos estructurad.

GMF:Grafical Modeling Framework, es un framework para la realización de editores gráficos basados en EMF.

Helper: Es un componente de ATL que permite la ejecución de cogido reutilizable, de la misma forma que lo hace un método en el lenguaje Java.

M2M: Model to Model, es un subproyecto de Eclipse que contempla un framework para la transformación entre modelos basado en QVT y ATL.

M2T:Model to Text, es un subproyecto de Eclipse que contempla un framework para la transformación de un modelo a texto sin importar el formato, este framework esta basado en Jet y acceleo.

Mapeo: Es el proceso definido dentro de los conceptos de MDA en el que se hace una transformación ya sea entre modelos o de modelo a texto.

Marca: Es un concepto definido dentro de MDA, en el que se marca el modelo para guiar la transformación entre modelos, dependiendo de el modelo al que se quiera llegar.

MDA:Model Driven Architecture, Es un estándar y paradigma propuesto por la OMG, en el que a partir de los diagramas hechos en el diseño, se realiza el desarrollo del proyecto de forma automática, con muy poca perdida de información.

MDD:Model Driven Development.

Meta-Metamodelo:Es un lenguaje que crea y es capas de definir metamodelos.

Metamodelo:Es un lenguaje capas de definir modelos.

Modelo:Es un conjunto de elementos que describen una realidad física o hipotética.

MOF:Meta-Object Facility, es un meta-metamodelo creado por la OMG, el cual permite crear metamodelos como UML.

OMG:Object Management Group, Es un grupo de organizaciones y empresas dedicadas a la investigación de estándares y tecnologías de software.

PIM:Platform-Independent Model, es un modelo menos abstracto que un CIM, que es independiente de la plataforma o tecnología que se usara para la implementación.

PSM: Platform-Specific Models, Modelo especifico para la plataforma es un modelo con un nivel de abstracción alto en el que la información comprende datos de la plataforma o tecnología escogida para la implementación.

QVT:*Query/View/Transformatio, es un lenguaje de transformación de modelos basado en consultas, vistas y transformaciones.*

Realización: Nivel de abstracción se refiere a la cantidad de información del modelo, entre menos información, menos realizado es el modelo; es lo opuesto a la abstracción.

Rule: Es un componente de ATL, el cual es el encargado de realizar las transformaciones entre los modelos.

- **LÍNEAS DE PRODUCCIÓN**

línea de producción: Es un sistema de fabricación de productos, en el hay un conjunto de elementos secuenciales que procesan el materia prima y cada elemento tiene un proceso diferente para realizar con la salida del proceso anterior.

VAO:Visualize, Analize, Optimize, es el núcleo para la optimización de la herramienta promodel.

Optimización: Es la forma de mejoramiento de una línea de producción, por medio de la maximización o minimización de una de las características de esta, para lograr un mejor desempeño.

- **OTROS**

Acceleo:Es un generador de código que esta basado en MDA y trabaja con modelos y plantillas.

AMM: Herramienta de optimización de modelos matemáticos.

Componente: Se denomina componente a un elemento con una función especifica que hace parte de algo mas grande.

Demo: Demostración de un prototipo de una funcionalidad.

Drag and drop:Arrastrar y soltar, es una técnica muy útil e intuitiva para manejar interfaces gráficas de usuario.

Framework: Es una estructura de software con una función especifica, que puede ser unida a otros y que es capas de desarrollar piezas de software.

Herramienta CASE: Herramienta asistida por computados para tareas especificas que facilitan el uso al usuario.

IDE: *Integrated Development Environment,*Entorno de desarrollo Integrado es una herramienta de software compuesta por diferentes aplicaciones capaces de crear piezas de software.

Interface: El la forma que en que una componente de software expone sus funciones para que sean accedidos por otros.

Iteración: Dentro de la metodología OpenUP es el ciclo por que para el proyecto para mostrar un demo al cliente;

Micro-Incremento: Dentro de la metodología OpenUP es un paquete de entregables que no deben demorar mas de un par de días en su construcción.

Modelo Matemático:Es un modelo científico que permite la representación de una realidad por medio de una función matemática.

Plugin:Es un complemento que permite aumentar las funcionalidad de una aplicación dentro de un IDE.

RCP:Rich Client Plataform, Es una herramienta que tiene una funcionalidad especifica como la del plugin, pero funciona de manera independiente al IDE.

RUP:Rational Unified Process metodología propuesta por IBM para administración de proyectos de desarrollo de software.

Subversion: Sistema de versionamiento de archivos, que permite llevar un historial y fácil manejo de proyecto de desarrollo grupales.

Velocity: Asi como Acceleo, velocity es un generador de código a base de plantillas.

XMI: XML Metadata Interchange, es el estándar de la OMG para el transporte de modelos entre herramientas de diagramacion UML.

BIBLIOGRAFÍA

References

ATL Project (2009). . Retrieved from http://www.eclipse.org/m2m/atl/.

ATL/User Guide (2009). . Retrieved from http://wiki.eclipse.org/ATL/User_Guide.

Balanceo de Línea (2009). . Retrieved from http://www.elprisma.com/apuntes/ingenieria_industrial/balanceodelinea/.

Ben Collins-Sussman, B. (2008). Version Control with Subversion. : O'Reilly.

CASE tool index (2009). . Retrieved from http://www.unl.csi.cuny.edu/faqs/software-enginering/tools.html.

DePuy, G. W. & Whitehouse, G.E. (2000). Applying the COMSOAL computer heuristic to the constrained resource allocation problem. *Comput. Ind. Eng.*, *38*, pp. 413-422.

Eclipse Modeling - EMF (2009). . Retrieved from http://www.eclipse.org/modeling/emf/.

Eclipse Modeling M2T (2009). . Retrieved from http://www.eclipse.org/modeling/m2t/.

Graphical Modeling Framework (2009). . Retrieved from http://www.eclipse.org/modeling/gmf/.

Joaquin Miller, J. (2003). The Several Styles of Model Driven Architecture. , , p. 73.

Jose Arroyo, J. (2009). . Retrieved from http://www.unal.edu.co/salacam/tutorialpromodel/index.htm.

JOT: Journal of Object Technology (2009). . Retrieved from http://www.jot.fm/issues/issue_2006_11/article4/.

Juan carlos Osorio, J. (2008). Modelo de programacion jerarquica de la produccion de un job shop flexible con interrupciones y tiempos de alistamiento dependientes de la secuencia. , , .

Modelo To Model (M2M) (2009). . Retrieved from http://www.eclipse.org/m2m/.

ModelWare - Rich Media (2009). . Retrieved from http://www.modelware-ist.org/index.php?option=com_content&task=view&id=90&Itemid=104.

Object Management Group (2009). . Retrieved from http://www.omg.org/.

OpenUP (2009). . Retrieved from http://epf.eclipse.org/wikis/openup/.

OpenUP in a Nutshell (2009). . Retrieved from http://www.ibm.com/developerworks/rational/library/sep07/kroll/.

Pedro Linares, P. (2001). *Modelos matematicos de optimizacion*. (). :

process chain in automated (2009). . Retrieved from http://www.plasmo.eu/site/index.php/en/applications/process-chains/process-chain-in-automated-production.

Promodel (2009). . Retrieved from http://www.promodel.com/products/promodel/.

Reinaldo Moraga, R. (s.f). *Un enfoque de solucion eficaz para problemas combinatorios*. (). :

Rich Client Platform (2009). . Retrieved from http://www.eclipse.org/articles/Article-RCP-1/tutorial1.html.

Rich Client Platform - Eclipsepedia (2009). . Retrieved from http://wiki.eclipse.org/index.php/Rich_Client_Platform.

Richard Muther, R. (1990). Production-Line Techniqu. : McGraw-Hill book company.

Ruben Lafuente, R. (2007). *Optimizacionde una linea de produccion mediante el estudio de metodos y tiempos de mejora*. (). :

Rutherford Aris, R. (1994). mathematical modeling techniques. : Dover Publications Inc..

SEDE BOGOTA DNSAV (2009). . Retrieved from http://www.virtual.unal.edu.co/cursos/economicas/2006862/lecciones/capitulo%206/cap6_e.htm.

SmartQVT documentation (2009). . Retrieved from http://smartqvt.elibel.tm.fr/doc/fr.tm.elibel.smartqvt.doc/index.html.

Stefan Jablonski, S. (2004). Web application and platform architectures. : Springer Berlin Heidelberg.

Stephen Mellor, S. (2004). MDA Distilled: Principles of Model-Driven Architecture. : Addison Wesley.

(2009). . Retrieved from http://www.eclipseplugincentral.com/.