

HACIA UNA ONTOLOGÍA PARA FÁBRICAS DE SOFTWARE

Kenyer Domínguez¹

María A. Pérez², Luis E. Mendoza² y Anna Grimán²

RESUMEN

Actualmente se está retomando el concepto de Fábricas de Software (FS) donde la reutilización juega un rol protagónico. Debido a la existencia de visiones diferentes en el área y a pesar de que el concepto de FS no es nuevo en la Ingeniería de Software, aún no se cuenta con la madurez conceptual necesaria como para identificar claramente el tratamiento de ciertas variables dentro del proceso. Por ello, en este artículo se realiza una revisión histórica del concepto FS y se propone una ontología basada en las definiciones más recientes. Este artículo forma parte de una investigación en progreso que pretende precisar la calidad sistémica en compañías desarrolladoras de software que decidan implantar una FS.

Palabras Claves: Fábricas de Software, ontología, reutilización, mejores prácticas.

1. INTRODUCCIÓN

Dado los exigentes requerimientos en el cumplimiento de tiempos y presupuestos que las empresas desarrolladoras de software viven constantemente, la eficiencia en el proceso de desarrollo se ha constituido en una necesidad creciente. Sin embargo, ésta es una variable que no necesariamente propicia su efectividad.

Es por ello que la reutilización en el proceso de producción de software ha sido foco de interés desde hace algún tiempo, sin obtener éxitos reales. Actualmente, los desarrolladores crean aplicaciones desde cero por medio del uso de herramientas genéricas, procesos adecuados y arquitecturas personalizadas; una práctica que consume mucho tiempo y que da lugar a errores.

Consciente de esta realidad, algunas organizaciones han establecido diversas estrategias de trabajo e incluso han agrupado las mejores prácticas en esta materia con el fin de industrializar el desarrollo de software. Es por ello que actualmente se está retomando el concepto de Fábricas de Software (FS) donde la reutilización juega un rol protagónico.

Debido a la existencia de visiones diferentes en el área

y a pesar de que el concepto de FS no es nada nuevo en la Ingeniería de Software; aún no se cuenta con la madurez conceptual necesaria, como para identificar claramente el tratamiento de ciertas variables dentro del proceso, una de ellas es la calidad.

En este artículo se pretende establecer una ontología que permita entender las diferentes acepciones del concepto FS, así como las características, innovaciones e implicaciones de las visiones más recientes de FS en el ámbito de la Ingeniería del Software. Este artículo forma parte de una investigación en progreso que busca estimar la calidad sistémica en compañías desarrolladoras de software que decidan implantar una FS.

A continuación se presentan los antecedentes de esta investigación, luego se describe la metodología, para luego dar una descripción de cada uno de los conceptos involucrados a medida que se muestra la representación gráfica de los elementos resaltantes, finalmente se muestra el modelo conceptual integrado y se proponen las conclusiones y recomendaciones.

2. ANTECEDENTES

En la Ingeniería del Software se han hecho muchos intentos para solventar la ausencia de un proceso de desarrollo bien definido y bien gerenciado que afecta, tanto a los desarrolladores como a usuarios y clientes, al no obtener el producto deseado dentro del tiempo y los costos estimados. La aplicación de un modelo de FS es uno de esos intentos, donde la reutilización del código, la especificación de procesos y el desarrollo dirigido por modelos juegan un papel protagónico.

¹ Decanato de Investigación y Desarrollo, Universidad Simón Bolívar. Apartado Postal 89000, Caracas 1080-A. Caracas - Venezuela. E-mail: kdoming@usb.ve

² Laboratorio de Investigación en Sistemas de Información (LISI), Departamento de Procesos y Sistemas, Universidad Simón Bolívar. Apartado Postal 89000, Caracas 1080-A. Caracas - Venezuela. E-mail: mvalles@usb.ve, lmendoza@usb.ve y agriman@usb.ve.

No obstante, el término FS ya ha sido utilizado en la Ingeniería de Software. Según [10], el término FS fue utilizado por primera vez entre 1950 y 1960 en Estados Unidos y Europa para mejorar la productividad en el desarrollo de software a través de herramientas estándar y reutilización de métodos y de componentes; luego, entre 1970 y 1980, los fabricantes de computadores en Japón adoptan de nuevo el concepto y mejoran considerablemente el desarrollo de software.

Aún en la década de 1990, según [10] se reportaron investigaciones sustanciales en relación al término FS. Posteriormente, [11] anuncia su visión del concepto atada a su producto “Visual Studio Team System 2005” y a su plataforma Microsoft .NET. Casi en la misma fecha, [3], presentan las técnicas que utilizan para el desarrollo de una FS en Brasil, donde incluso actualmente se está conformando una Fábrica de Software basada en Código Abierto [14].

Como antecedentes importantes en el tema de las FS se tienen las visiones más recientes encontradas en la literatura: [3] y [6]. Según [3], una compañía de software que no tenga las siguientes características no puede ser considerada una FS:

- Producción de software en masa y en gran escala,
- Estandarización de tareas y controles; y
- División de trabajo y automatización.

3. METODOLOGÍA UTILIZADA

Se observa que el término FS engloba una cantidad de conceptos que conviene manejarlos ordenadamente y este es justamente el fin de esta investigación en progreso. Según [12] una ontología es una descripción explícita y formal de conceptos en un dominio de discurso. Estos autores en su metodología para crear ontologías, proponen los siguientes siete pasos:

1. Determinar el dominio y el alcance.
2. Considerar la reutilización de ontologías existentes.
3. Enumerar los términos importantes.
4. Definir las clases y su jerarquía
5. Definir las propiedades de las clases
6. Definir las relaciones de las clases
7. Crear las instancias

Para efectos de presentación en este artículo, no se detallan cada uno de los pasos propuestos; sin embargo, a continuación se resaltan las actividades realizadas en cada uno de ellos: (1) para encontrar los términos relevantes se identificaron aquellos

conceptos comunes de cada definición, así como también los no comunes pero más importantes; (2) para restringir el alcance, sólo se tomaron en cuenta las versiones más recientes de FS; (3) no se encontraron ontologías en este tema por lo que no hubo reutilización de ontologías existentes; (4) los términos importantes son descritos iterativamente a lo largo de este artículo, especificando la jerarquía entre ellos así como también sus relaciones; (5) a este nivel no se especifican las propiedades ni se crean las instancias dado que aun no se aplica a ningún caso de estudio.

4. ONTOLOGÍA PROPUESTA PARA FS

De acuerdo con [3], una FS debe ser flexible, capaz de producir varios tipos de productos, implementar conceptos de ingeniería del software y debe ser capaz de analizar, proyectar, implementar, desarrollar y mejorar sistemas. Se observa que el término FS no es nada nuevo pero, en su mayoría, las diferentes versiones del concepto FS coinciden en la producción en masa de una Familia de Productos (perteneciente a un determinado Dominio de Solución), que cumplen con determinados requerimientos comunes (pertenecientes a un determinado Dominio del Problema), que son establecidos por un conjunto de personas involucradas. Estos involucrados no necesariamente son los desarrolladores, sino que también contempla a los clientes, usuarios, proveedores e incluso los arquitectos de software.

En la Figura 1 se presenta un primer nivel de conceptos en base a los antecedentes reseñados.

Por su parte, [6] definen el término FS como una ‘**Línea de Productos de Software**’ (LPS) que configura herramientas, procesos y contenido usando una **plantilla** basada en un **esquema** dado, con el fin de automatizar el desarrollo y mantenimiento de variaciones de un producto arquitectónico mediante adaptaciones, ensamblaje y configuración de componentes basados en *frameworks* soportados por un **ambiente**.

La definición de Greenfield se toma como punto de referencia para la ontología que se desea proponer, posteriormente se verá que la misma, en sus detalles, contiene las definiciones del resto de los autores mencionados. Para entender mejor este modelo, se realiza una segunda iteración para analizar por separado los conceptos más relevantes: Línea de Productos de Software (LPS), plantilla, esquema y ambiente.

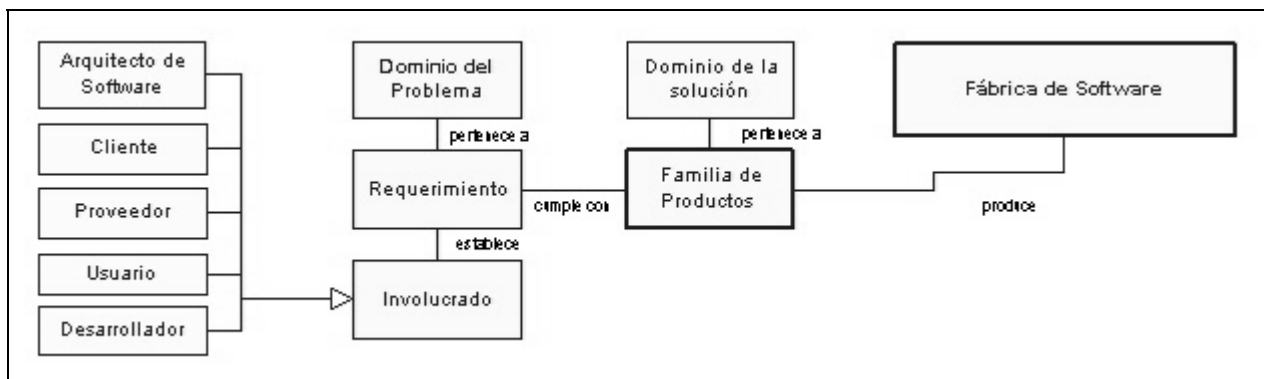


Figura 1. Elementos que intervienen en la definición de FS.

4.1. Línea de Productos de Software

Según [4], una línea de productos se puede ver como un conjunto de productos que están estrechamente relacionados (por su funcionalidad), que son vendidos a los mismos grupos de compradores, que son comercializados a través del mismo tipo de distribución, o que caen en el mismo rango de precios. Este concepto también tiene su origen en la industria tradicional, pero es aplicable al desarrollo de software y su utilización no resulta muy novedosa. La conocida utilización del concepto de LPS quizá se deba a su alto nivel de generalidad, pudiéndose aplicar tanto en organizaciones grandes como en pequeñas.

La mayoría de las organizaciones producen familias de sistemas similares que se diferencian por ciertas características; pero a nivel general, según [2] se han detectado tres actividades generales que pueden ser aplicadas en cualquier situación. Para estos autores, las tres actividades están altamente relacionadas: el avance en una de las actividades necesariamente implica el avance en las demás, por lo que en realidad no se puede decir que exista un orden entre ellas, pero por razones de espacio cada una será descrita individualmente y de forma lineal.

- **Desarrollo de Activos Centrales (Core Asset Development)** Esta actividad se enfoca en la producción de recursos generales para cualquier producto. Aquí se define el alcance de la línea de productos, el plan de producción, los componentes, la arquitectura, los requerimientos, así como también las restricciones, los estilos, patrones y frameworks, entre otros. Se configura también el inventario de activos (*assets*) preexistentes, lo que más tarde FS denomina Biblioteca de Objetos.
- **Desarrollo del Producto (Product Development):** Esta actividad se encarga de ensamblar los activos desarrollados o almacenados en la actividad anterior para crear productos acordes con los requerimientos

del mercado; sin embargo, este proceso raramente es lineal. La creación de productos implica una continua relación con el alcance, el plan de producción y los activos centrales (*core assets*).

- **Gestión (Management):** La dirección juega un rol crítico en el éxito de la línea de producción. Las actividades deben tener unos recursos asignados, coordinados y supervisados. La dirección, tanto a nivel técnico como organizacional debe estar fuertemente asociada al esfuerzo de la LPS. Pero la dirección no sólo está dirigida hacia adentro, también se debe tener en cuenta la relación con los proveedores y consumidores. Este aspecto posteriormente es llamado Cadena de Suministro por FS.

Por su parte, [3], aunque basado en el modelo de FS de [1], propone un modelo que separa la producción de activos centrales o componentes, del proceso de producción de software, y el proceso de gestión lo llama "Organización del trabajo". Este modelo está compuesto por dos grandes actividades: Producción de Software y Producción de Componentes.

La concepción de FS de [6] es un poco más amplia ya que abarca el modelado del dominio de aplicaciones. FS está fielmente basado en LPS y comparte actividades con el modelo de [3], tales como Análisis, Diseño e Implementación, pero en el modelo de [6]. éstas están orientadas para una línea de productos. Los activos en el modelo de [6] son los que [3] denomina componentes. Este último modelo también contempla la posibilidad de seleccionar las herramientas más adecuadas para cada dominio, con la opción de personalizarlas.

Es importante resaltar la similitud que posee el modelo de FS de [3] y la plantilla de FS propuesta por [6] dado que ambos modelos fueron publicados en fechas similares pero utilizando referencias bibliográficas distintas, lo que refleja el interés actual en mejorar los procesos de desarrollo para producir sistemas de mayor calidad. Una vez conocido el término LPS y su rol

dentro de FS (ver Figura 2), se pueden definir el resto de los elementos que la conforman: esquema, plantilla

y ambiente.

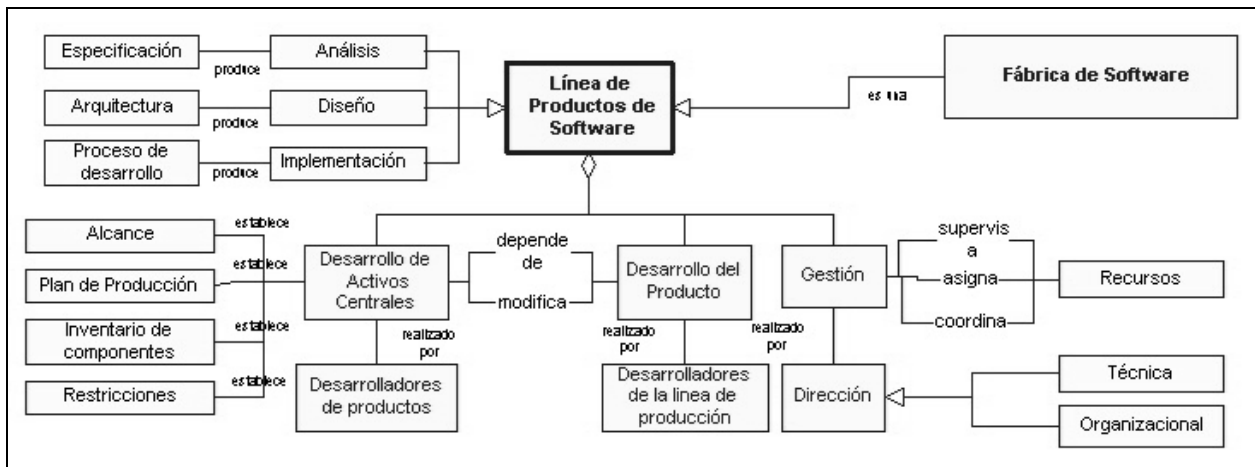


Figura 2. Conceptos de la Línea de Productos de Software.

4.2. Esquema, Plantilla y Ambiente

Según [5] una FS es una LPS que configura herramientas de desarrollo con guías y paquetes de contenido, cuidadosamente diseñados para construir tipos específicos de aplicaciones. Estos mismos autores establecen que una FS contiene tres ideas claves: un esquema, una plantilla y un ambiente de desarrollo.

- **Un esquema** se puede pensar como una receta. Es una lista de ingredientes como proyectos anteriores, directorios de código fuente, archivos SQL y archivos de configuración. Además, el esquema explica cómo deben ser combinados estos ingredientes para crear el producto. El esquema establece cuáles lenguajes específicos del dominio (DSL del inglés *Domain Specific Language*) deben ser usados y describe cómo los modelos basados en esos DSL pueden ser transformados en código o en otros artefactos. El esquema describe la arquitectura de la línea de productos y las relaciones entre componentes y los *frameworks* involucrados.
- **Una plantilla** es cómo un paquete o bolsa de supermercado que contiene todos los ingredientes listados en el esquema. La plantilla provee los patrones, guías, ejemplos, herramientas específicas como editores gráficos de DSL, *scripts*, hojas de estilo (*style sheets*) y otros ingredientes necesarios para construir el producto.

- **Un ambiente** de desarrollo es como la cocina donde es elaborado el alimento. Herramientas de alto nivel como *Microsoft Visual Studio Team System* hacen posible un ambiente de trabajo adecuado para construir una familia de productos.

Según [5], esta analogía puede ser ejemplificada un poco más teniendo entonces que los productos son los platos servidos en un restaurant. Los involucrados (*stakeholders*) son como los usuarios que ordenan platos del menú. Una especificación de producto es como una orden de un plato con algunos extras.

Los desarrolladores de productos (*product developers*) son como los cocineros que preparan los platos descritos en las órdenes y quienes pueden modificar ciertos ingredientes. Los desarrolladores de la línea de productos (*product line developers*) son como los chefs que deciden qué aparecerá en el menú y qué ingredientes, procesos y equipos de cocina deben ser usados.

Estos tres elementos (plantilla, esquema y ambiente) se muestran en la Figura 3 y conforman las bases tecnológicas de una FS según la concepción de [6].

El término FS no sólo está compuesto por aspectos tecnológicos, a continuación se definen ciertos conceptos de economía aplicados al desarrollo de software y que también deben estar presentes en toda empresa que decida adoptar esta tendencia.

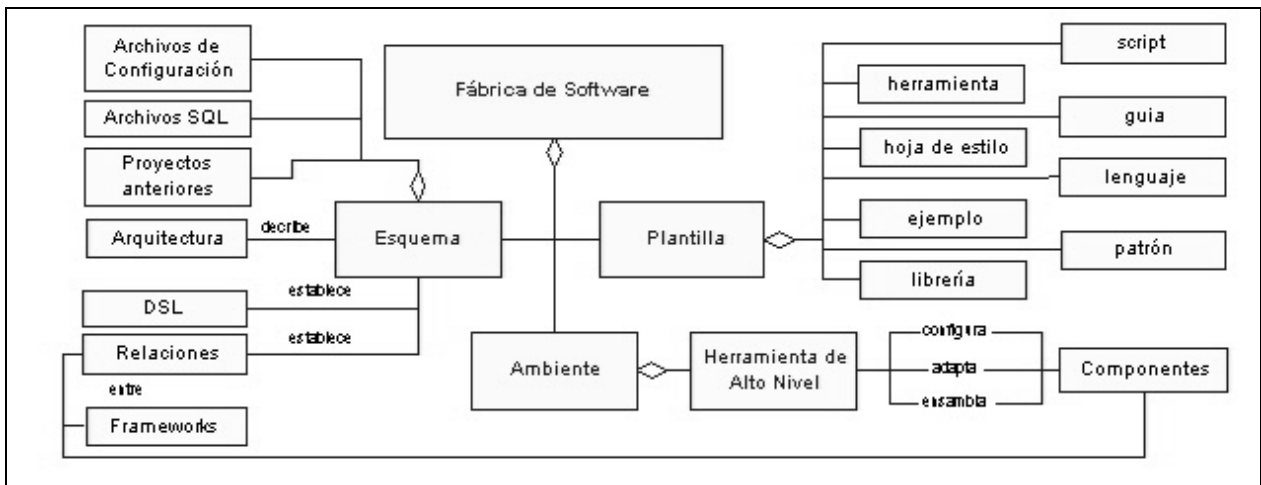


Figura 3. Elementos tecnológicos que conforman una FS.

4.3. Conceptos de Economía

El concepto de **canales de mercadeo o cadena de suministro** es utilizado por cualquier organización que produzca algún bien o servicio. La industria del software no escapa de esta realidad pero hasta el momento se ha limitado a utilizar este concepto sólo para mercadear el producto ya desarrollado, pero no como parte del proceso de desarrollo. Según [8], un canal de mercadeo realiza la labor de llevar los bienes de los productores a los consumidores, superando las brechas del tiempo, plaza y posesión. Según [6], una FS puede ser segmentada tanto vertical como horizontalmente para delegar responsabilidades a proveedores externos, creando de esta forma cadenas de suministros para el desarrollo de un producto de software.

Otro concepto utilizado por FS es la **economía de reutilización**. Según [7] citado por [6], el hecho de reutilizar soluciones de un sub-problema común en un dominio dado puede reducir el costo total de resolver múltiples problemas en el mismo dominio. Además, la reutilización puede reducir el tiempo total de lanzamiento al mercado (*time to market*) y mejorar la calidad del producto. Es importante señalar la nueva connotación que se le otorga a este concepto dentro de una FS, donde se toma en cuenta no sólo la reducción del tiempo sino del costo del proyecto e incluso se mejora el retorno de la inversión ya que al invertir en patrones arquitectónicos, componentes y posibles eslabones en la cadena de suministro, la reutilización hace posible ver los resultados financieros de forma más rápida, favoreciendo **las economías de escala y de alcance**.

Según [6], estos dos conceptos con frecuencia son confundidos y aunque ambos reducen el tiempo y el costo del producto y mejoran su calidad produciendo múltiples productos en lugar de copias individuales, existen grandes diferencias entre ellos. La economía de escala establece la posibilidad de realizar múltiples instancias idénticas partiendo de un diseño simple. Por su parte, la economía de alcance se plantea cuando múltiples pero similares diseños y prototipos son producidos colectivamente más que individualmente.

La diferencia principal entre la economía de escala y la economía de alcance, en términos de FS, está en el momento en que se produce la reutilización. En el primer tipo de economía, la reutilización se produce después de tener el producto desarrollado, mientras que en la economía de alcance, la reutilización se produce durante el desarrollo del producto. La relación de estos conceptos se puede ver en la Figura 4.

Tanto la economía de escala como la de alcance, dentro de la concepción de Fábrica de Software, aumentan la eficiencia en el proceso de desarrollo de software dado que no se limitan al desarrollo de un producto a la vez, sino que transitan hacia el desarrollo en masa, sustentado fuertemente por el concepto de Línea de Productos de Software. Al mejorar la eficiencia, por ende, se mejora la calidad en el proceso de desarrollo.

Según [3], el desarrollo de una FS implica que las mejores prácticas de la ingeniería del software sean aplicadas sistemáticamente. A continuación se muestran brevemente algunas de las mejores prácticas que esta tendencia ofrece.

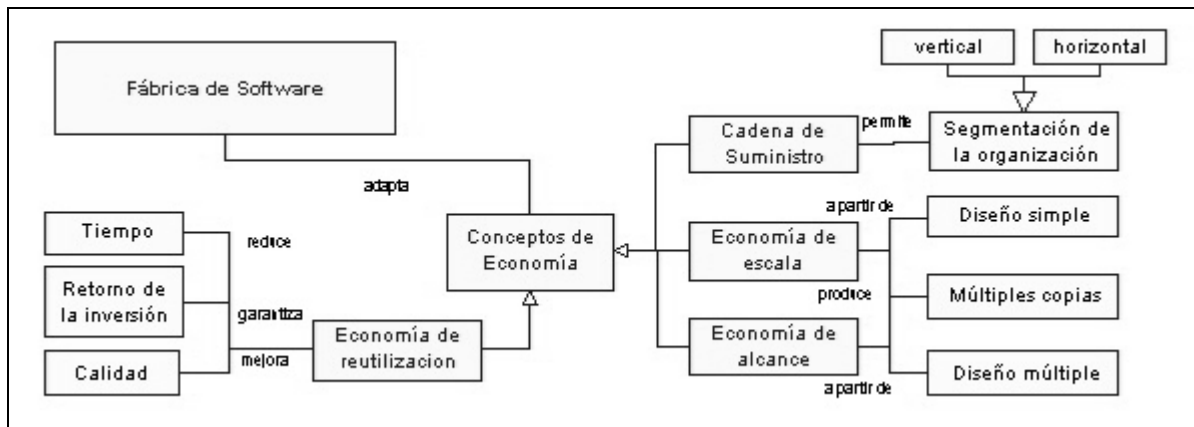


Figura 4. Conceptos de Economía en una FS.

4.4. Mejores Prácticas

Según [6], FS está basado en la convergencia de ideas claves en desarrollo dirigido por modelos y reutilización sistemática. Para los autores citados, el **Desarrollo Dirigido por Modelos** (MDD del inglés *Model-Driven Development*) reduce el problema de la abstracción por medio del uso de modelos que capturan información de tal manera que pueda ser fácilmente procesada, en lugar de crear modelos sólo como documentación. Esta es una de las principales prácticas propuestas por FS ya que, según estos autores, los modelos deben ser utilizados para desarrollar, más que para documentar el producto.

Todos estos esfuerzos van encaminados a pasar, de la forma tradicional basada en el código fuente, a una nueva forma de desarrollar productos de software basada en modelos, donde se reutilicen cada vez más los intentos anteriores. Para [13], la **Ingeniería del Software Basada en Componentes** (ISBC) lucha por conseguir un conjunto de componentes de software preconstruidos y estandarizados para encajar en un estilo arquitectónico específico para algún dominio de aplicación. Este conjunto de componentes constituye el Desarrollo de Activos Centrales de LPS.

Así como [7] establece que un framework es un grupo de clases para un tipo específico de software, se puede intuir entonces que un framework de procesos es un grupo de sub-procesos para un determinado tipo de dominio. De acuerdo con [5] un **framework de procesos** es una estructura que organiza los micro-procesos usados para construir artefactos que componen los miembros de la familia de productos. Desarrollar un framework de procesos para una línea de productos de software tiene sentido porque el costo puede ser efectivamente amortizado con la producción de muchos productos.

Estos mismos autores afirman que un framework de procesos esencialmente descompone un proceso en micro-procesos adjuntados al desarrollo de varios tipos de artefactos, incluyendo la descripción de los requerimientos necesarios para tal fin. En estas descripciones se puede enumerar consideraciones tan variadas como puntos clave de decisiones, análisis de trade-off asociados con cada punto de decisión, actividades opcionales o requeridas y los resultados a ser producidos en cada actividad.

En este sentido, según [9], hasta ahora la arquitectura de un sistema permanecía siempre en una especie de caja negra siendo un secreto conocido sólo por sus autores. Una buena arquitectura del software fomenta la **reutilización** y permite obtener y mantener el control intelectual del desarrollo así como también gerenciar su complejidad y mantener la integridad del sistema, manejando de forma implícita la calidad en el desarrollo. La Figura 5 agrupa los conceptos involucrados en las mejores prácticas que reúne FS.

5. MODELO CON CONCEPTOS COMPARTIDOS

Una vez que se han descrito por separado los aspectos compartidos en torno a FS, conviene mostrarlos en un modelo unificado donde están presentadas nuevas relaciones no mostradas hasta el momento. Una de las principales relaciones está basada en que los aspectos económicos adaptados por FS al desarrollo de software, están soportados por los aspectos tecnológicos, de esta forma se tiene que la **economía de reutilización** fomenta la **reutilización sistemática**, las **economías de escala y alcance** fomentan el **Desarrollo Dirigido por Modelos**, y la **Cadena de Suministros** debe estar coordinada por la **Dirección** de la Línea de Productos de Software.

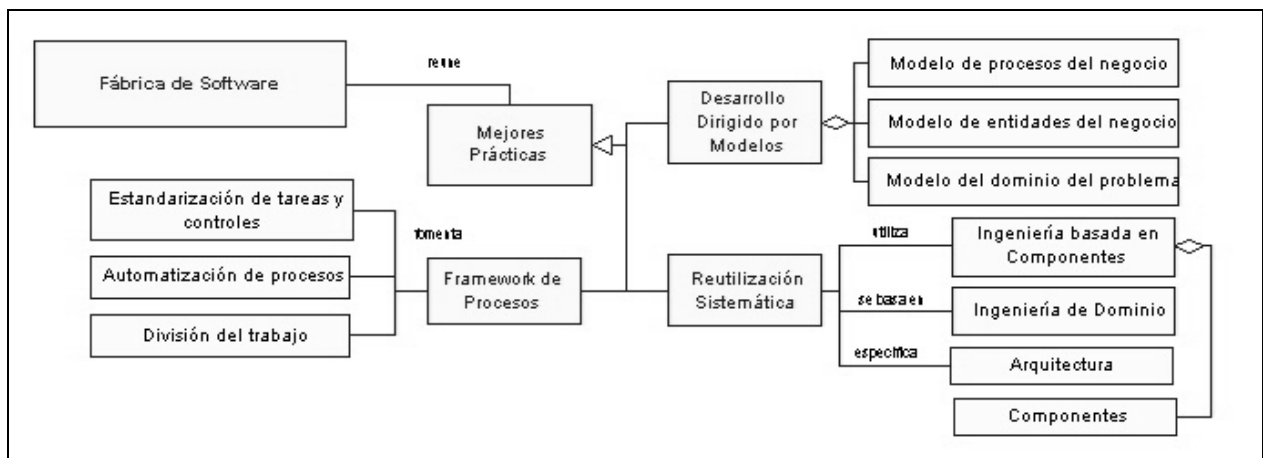


Figura 5. Mejores Prácticas en el Desarrollo de Software.

Por otro lado, al presentar todos los modelos de forma unificada, se destacan aquellos conceptos que se repiten y que a su vez sirven de integradores entre las partes involucradas. De esta forma se puede establecer que en una FS, la **Línea de Productos de Software** desarrolla un conjunto de productos con características similares conformando una **Familia de Productos**. Para desarrollar estos productos se sigue un **esquema** determinado que describe la **arquitectura** del sistema y a su vez establece las relaciones entre los **frameworks** utilizados y los **componentes** a ensamblar. Estos **componentes** se van registrando en el **inventario de componentes** fomentando la **reutilización sistemática**.

Como se ha mostrado hasta ahora, el término FS, además de poseer múltiples definiciones, reúne una gran cantidad de conceptos que conviene identificar para poder manejar una idea general de su alcance. En la Figura 6 se muestra una versión unificada del modelo conceptual propuesto, incluyendo las nuevas relaciones descritas anteriormente. Los recuadros no implican ningún tipo de orden ni relevancia entre conceptos, se utilizan sólo para agrupar conceptos similares y facilitar la lectura del modelo.

Si bien FS no aporta conceptos nuevos en el desarrollo de productos de software, la innovación consiste en la agrupación de conceptos anteriores (LPS, MDD, reutilización sistemática y framework de procesos), pero presentados en conjunto, como un nuevo intento para pasar de la producción tradicional a una automatizada.

6. CONCLUSIONES Y TRABAJOS FUTUROS

FS no sólo implica cambios tecnológicos sino también cambios en la economía; por lo tanto, cualquier empresa que decida adoptar FS como base para su

forma de producir software, debe tener en cuenta ciertos conceptos de economía que no estaban involucrados anteriormente en este sector de producción. Luego de desarrollar este trabajo, las conclusiones obtenidas pueden resumirse en dos aspectos fundamentales:

- Se dan los primeros pasos para organizar y documentar todo el conocimiento que gira en torno al concepto de Fábricas de Software.
- Hasta el momento no existe una forma de precisar la calidad de una empresa que decida implantar una FS como tendencia de trabajo.

Como corolario a la primera afirmación, se tiene que la reutilización y el desarrollo de varios productos en paralelo, deben ser las características fundamentales en una empresa que decida encaminarse hacia una Fábrica de Software. Otro aspecto de la problemática, es que **el tema de la calidad de los sistemas no es atacado profundamente** en todas las versiones de FS, aunque siempre se busca mejorar la eficiencia y mitigar, en la medida de lo posible, los errores y sus consecuencias, tratando siempre de medir la calidad del producto.

FS propone ciertos cambios en el proceso de desarrollo de software pero no establece una forma clara de medir su calidad. Por lo tanto, el resultado esperado se encamina a establecer un modelo de especificación de la calidad sistémica (calidad del producto y calidad del proceso) en empresas venezolanas que cumplan con las características de FS. Para ello además, se aplicará esta primera ontología a un caso de estudio para validar los conceptos. Luego se formulará el modelo y se hará su validación aplicando el método DESMET, el cual permite evaluar métodos y herramientas usados en el área de Ingeniería de Software. Se espera entonces que el modelo tenga al menos un 75% de aplicabilidad y de pertinencia.

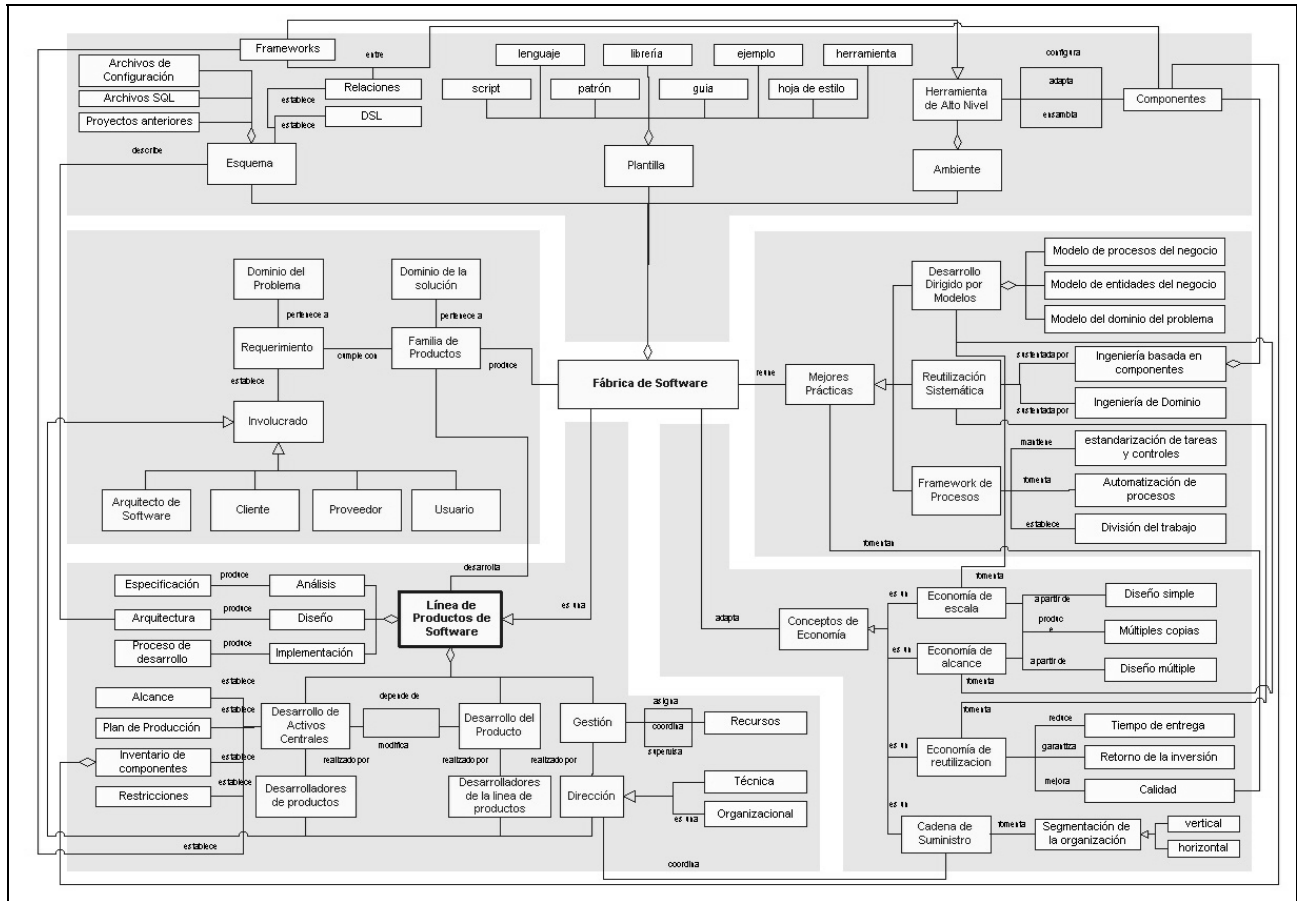


Figura 6. Modelo conceptual propuesto para Fábricas de Software.

7. REFERENCIAS BIBLIOGRÁFICAS

- [1] V. Basili, G. Caldiera y G. Cantone “A Reference Architecture for the Component Factory”. ACM Transaction on Software Engineering and Methodology. Vol 1. nº 1. pp 53-80. Enero, 1992.
- [2] P. Clements y L. Northrop. “Software Product Lines. Practices and Patterns”. SEI Series in Software Engineering. Addison Wesley. Segunda Edición. Boston, USA. pp. 29-31. 2001.
- [3] J. Fabri, A. Presende, L. Begosso, A. L’erário, F. Fujii y M. de Paula. “Techniques for the Development of a Software Factory: Case CEPEIN-FEMA”. 17th International Conference Software and Systems Engineering and their Applications. ICSSEA 2004. Paris, Noviembre 30 – Diciembre 3. 2004.
- [4] F. García, J. Barras, M. Laguna y J. Marqués. “Líneas de Productos, Componentes, Frameworks y Mecanos”. Informe Técnico, Departamento de Informática. Universidad de Salamanca. 2002.
- [5] J. Greenfield, y K. Short. “Moving to Software Factories”. White Paper, actualizado el 15 de Julio de 2004, tomado en Enero de 2005 de <http://www.softwarefactories.com/ScreenShots/M S-WP-04.pdf>
- [6] J. Greenfield, K. Short, S. Cook, y S. Kent. “Software Factories. Assembling Applications with Patterns, Models Frameworks and Tools”. Wiley. Primera Edición. 2004.
- [7] I. Jacobson, M. Griss y P. Jonsson. “Software Reuse : Architecture, Process and Organization for Business Success”. ACM Press, 1997.
- [8] P. Kotler. “Dirección de Marketing”. La edición del milenio. Prentice Hall. 2001.
- [9] P. Kruchten. “The Rational Unified Process. An Introduction”. Third Edition. Addison Wesley

Longman, Inc. 2003.

- [10] N. Lim, S. James, y F. Pavri. "Diffusing Software-based Technologies with a Software Factory Approach for Software Development. A Theoretical Framework". Proceeding of the 22nd Information Systems Research Seminar in Scandinavia (IRIS22): Enterprise Architectures for Virtual Organisations. 7-10 Agosto, Keuruu, Finland. 1999.
- [11] Microsoft Prensa. "Microsoft aumenta su ecosistema de socios alrededor de Visual Studio 2005 Team System". Tomado en Febrero de 2004 de <http://www.microsoft.com/latam/prensa/2004/Octubre/VisualStudio.asp>
- [12] N. Noy. y D. McGuinness. "Ontology Development 101: A Guide to Creating Your First Ontology". in Protege Documentation. Stanford University: Stanford, CA. 2001
- [13] R. Pressman. "Ingeniería del Software. Un enfoque práctico". Quinta Edición. Mc Graw Hill. 2002.
- [14] OXE Open Source Software Factory, "Open Experience Environment". Tomado en Julio de 2005 de <http://php.cin.ufpe.br/~oxe/>