Implementando objetos del estandar ANSI/ISA-95 con el patrón de diseño decorador "decorator"

VASQUEZ-SESTEAGA, Fabian†*, DURAN-DE LEÓN, Junin, GUTIÉRREZ-TORRES, Ludivina, CRUZ-RENTERIA, Jesus Raúl, ZÚÑIGA-FÉLIX, Ismael A.

Instituto Tecnológico de Nogales
Instituto Tecnológico de Agua Prieta

Recibido Septiembre 30, 2016; Aceptado Febrero 01, 2017

Resumen

El estándar ANSI/ISA-95, propone una serie de reglas y procedimientos para que los distintos niveles de las empresas de manufactura puedan comunicarse entre si (ISA, 2010). Este estándar provee clases, lo cual facilita enormemente la integración de cualquier plataforma de desarrollo de software. Para este trabajo se ha tomado un objeto especificado en la "Parte 2: Object Model Attributes" de dicho estandar. Por otra parte los patrones de diseño son soluciones a problemas recurrentes dentro del diseño de software. Y es posible usarlos para solucionar algunos de los problemas mas frecuentes dentro del diseño de software. En este trabajo se usaron objetos del modelo "Production Schedule" el cual forma parte del estándar ANSI/ISA-95, este nos proporciono las clases y los objetos con los cuales se ha usado el patrón de diseño "Decorador". El principal objetivo de esta propuesta es agregar responsabilidades adicionales a un objeto proveniente del estándar ANSI/ISA-95 de forma dinámica. El uso del patrón Decorador nos proporciona una alternativa flexible que permite que las subclases extiendan su funcionalidad. Lo anterior da como resultado que la reusabilidad mejora en los desarrollos de software.

ERP, MRP, MES, Estándar ANSI/ISA-95, Patrones de Diseño, Patron de diseño Decorador "Decorator"

Abstract

The ANSI / ISA-95 standard proposes a series of regulations and procedures so that different levels of manufacturing companies can communicate with one another (ISA, 2010). This standard provides classes, which greatly facilitates the integration of any software development platform. For this work we have taken an object specified in "Part 2: Object Model Attributes" of this standard. On the other hand design patterns are solutions to recurring problems within software design. Also, it is possible to use them to solve some of the most frequent problems within the software design. In this paper we used objects from the "Production Schedule" model which is part of the ANSI / ISA-95 standard, which provided us with the classes and objects with which the "Decorator" design pattern was used. The main objective of this proposal is to add additional responsibilities to an object coming from the ANSI / ISA-95 standard dynamic manner. Using the Decorator pattern gives us a flexible alternative that allows subclasses to extend their functionality. This results in reusability improving within software developments.

ERP, MRP, MES, Estándar ANSI/ISA-95, Patrones de Diseño, Patron de diseño Decorador "Decorator"

Citación: VASQUEZ-SESTEAGA, Fabian, DURAN-DE LEÓN, Junin, GUTIÉRREZ-TORRES, Ludivina, CRUZ-RENTERIA, Jesus Raúl, ZÚÑIGA-FÉLIX, Ismael A. Implementando objetos del estandar ANSI/ISA-95 con el patrón de diseño decorador "decorator". Revista de Prototipos Tecnológicos .2017, 3-7: 20-26

*Correspondencia al Autor (Correo Electrónico: Fabian.Vasquez12@gmail.com)
† Investigador contribuyendo como primer autor.

Introducción

La gran complejidad en la manufactura de productos en los distintos ramos de la industria (automotriz, eléctrico, aéreo espacial, etc...), ha obligado a la industria a invertir en sistemas de software tales como:

- **ERP** ("Enterprise Resource Planning"). Se define como un método para la efectiva planeación y control para todos los recursos necesarios para tomar, hacer, embarcar y contabilizar ordenes de clientes en una organización de manufactura, distribución o servicio (Chen, 2003).
- **MRP** ("Manufacturing resource planning"). Es un sistema de planificación de componentes de fabricación que mediante un conjunto de procedimientos lógicamente relacionados, traduce un Programa Maestro de Producción, PMP, en necesidades reales de componentes, con fechas y cantidades. (Dominguez Machuca, 1995).
- **MES** ("Manufacturing Execution System"). Es una colección de sistemas integrados a través de una base de datos común, que monitorean y controlan toda la producción y sus actividades relacionadas al nivel de manufactura. (Nagalingam , 2007)

Estos sistemas como se describe anteriormente son para poder mantener y controlar los procesos de producción de una manera sencilla y maximizando el rendimiento y utilidades de las empresas.

Por otra parte la complejidad de la manufactura no desaparece, solo se traspasa al software el cual realiza una "digestión" del proceso para mostrarlo de manera amigable al usuario final. Para facilitar el desarrollo de software de sistemas complejos, ha surgido la necesidad de nuevas herramientas y estandares.

ISA decidió en los años noventa desarrollar la norma ISA-95 para la integración de la empresa y los sistemas de control reduciendo asi los riesgos, costos y errores que van de la mano con la implementación de sistemas de control de manufactura y la integración de estos con sistemas ERP. (Scholten, 2007) La aparición del estándar ANSI/ISA95, ha ayudado a resolver muchos problemas a grandes empresas. Desde el año 2000, empresas como Nestle, Arla Foods, Bat Manufacturing y MasterFoods han empleado el estándar ANSI/ISA95 para estandarizar el intercambio de información entre sus paquetes ERP y sus sistemas MES. La compañía Bat Manufacturing en Zevenaar, Holanda, incluso ha desarrollado sus propios modulos MES basados en modelos de datos ISA95, y MasterFoods en Veghel, Holanda ha implementado un paquete de programación compatible con ISA95. (Scholten, 2007).

En el 2011 se desarrollo un prototipo de Software MES que integra islas de automatización en manufactura utilizando ISA-95. En este prototipo se integraron los datos de multiples "islas de automatización "para una área de producción simulada, con dos líneas de producción compuestas de dos estaciones de trabajo. Se usaron indicadores de desempeño clave (KPIs). Los resultados mostraron que es factible el desarrollo de software MES basado en ISA-95 (Cruz Renteria, Zaragoza Peñuñuri, & Garcia Alva, 2011) El estándar ISA-95 tambien es posible aplicarlo en otros dominios diferentes al de la manufactura como muestra P. Vasilev en su trabajo "Capacidad final para la programación en la Industria del software" en este trabajo se utiliza el estándar para el problema de la programación en el desarrollo de software el cual se ha abordado desde muchos enfoques pero no con un estándar en este caso ISA-95 en este trabajo se usaron los modelos "Capability, Scheduling y Execution" los cuales están firmemente conectados con los recursos necesarios y su disponibilidad. Obteniendose la capacidad final de la programación de tareas y la gestión de calendarios. (Vasilev, 2015)

ISSN 2444-4995

Stuart Thiel, 2010, en su trabajo de tesis dice que: Por lo general, se entiende que las aplicaciones empresariales (EA) se basan en las aplicaciones que están destinadas a ser accedidas por múltiples usuarios, dentro de la misma organización. Las Aplicaciones Empresariales Basadas en la Web (WEAs) implican EAs disponibles a través de Internet. Sitios (como Amazon [AMAZON] y eBay [EBAY]), sitios bancarios, webmail, casinos en línea son ejemplos de la importancia de las WEAs. Mantenerse al día con el avance de WEAs requiere más que solo desarrolladores individuales.

Grandes equipos que comprenden muchas funciones son ahora bastante normales. Con el aumento de mano de obra y separación de roles viene una mayor necesidad de comunicación y rendición de cuentas. Herramientas más estandarizadas, lenguajes y enfoques deben ser desarrollados y adoptados para asegurar resultados más confiables. Uno de las claves para mejorar el lenguaje de comunicación utilizado en este campo es el uso de patrones de diseño. (Thiel, 2010)

Estándar ANSI/ISA-95

ISA-95 es el estándar internacional para la integración de empresa y sistemas de control. Este consiste en modelos y terminología. Estos se pueden utilizar para determinar información, la cual debe intercambiarse entre los sistemas de ventas, finanzas y logística y sistemas de producción, mantenimiento y calidad. Esta información esta estructurada en modelos UML, que son la base para el desarrollo de interfaces estándar, entre sistemas ERP y MES. La norma ISA-95 puede utilizarse para varios fines, por ejemplo como guía para la definición de los requisitos de los usuarios, para los proveedores, el personal o el equipamiento. (ISA, 2010)

En este escrito tomaremos un objeto especificado en la parte 2 del estándar "Object Model Attributes". En esta parte del estándar se definen los objetos de los modelos y terminologías vistos en la parte uno del estándar "Models and Terminology". (ISA, 2010)

En específico tomaremos el objeto"Production Schedule", el cual pertenece al modelo con el mismo nombre mostrado en la Figura 1 Diagrama del modelo "Production Schedule".

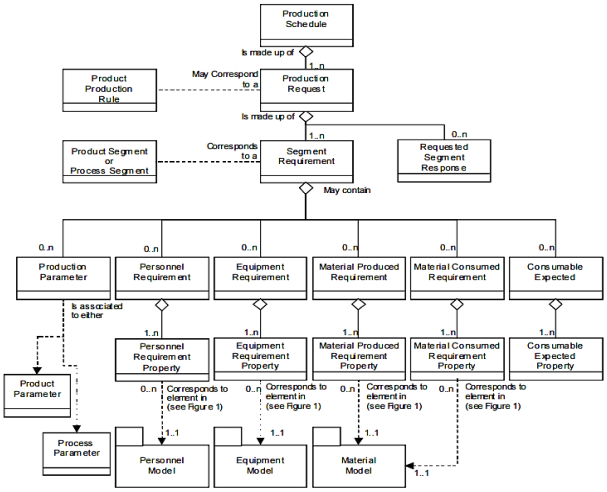

Figura 1 Diagrama del modelo "Production Schedule". Elaboración propia

Como se puede observar el estándar cuenta con una gran cantidad de modelos y objetos (ISA, 2010). En éste caso solo se necesita uno para mostrar la utilidad del estándar junto con las buenas prácticas de los patrones de diseño.

Los modelos definidos en el estándar cuentan solamente con los atributos y carecen de funciones o métodos. Esto es certado, ya que solo son prototipos y depende de las necesidades de las empresas y sus implementaciones el realizar las adiciones. Estas pueden ser procedimientos, funciones o inclusive nuevos atributos no definidos en el estándar. Lo cual nos lleva analizar la mejor manera de realizar estas adiciones.

Patrones de diseño

Los patrones de diseño son soluciones a problemas recurrentes dentro del diseño de software. En el año de 1990 se publicó el libro "Design Patterns" escrito por el grupo Gang of Four, a partir de la publicación de este libro fue que se comenzaron a utilizar y conocer exhaustivamente los patrones de diseño. La publicación cuenta con 23 de los patrones más comunes, detallando cada uno de los problemas y su solución por medio de un patrón (Gama, Helm, & Johnson, 1997).

Los patrones de diseño hacen más fácil la reutilización de diseños y arquitecturas exitosas. Expresando técnicas comprobadas como patrones de diseño, se hacen más accesibles para desarrolladores de nuevos sistemas. Los patrones de diseño ayudan a escoger alternativas de diseño que hacen a un sistema reusable y evitan alternativas que comprometen la reusabilidad. (Gama, Helm, & Johnson, 1997)

Alan Shalloway & James R. Trott expresan en su libro "Design patterns explained "las razones para usar patrones de diseño son: Reutilizar soluciones existentes y de alta calidad para problemas, establecer terminología común para mejorar las comunicaciones dentro de los equipos de desarrollo, cambiar el nivel de pensamiento a una perspectiva más alta, decidir si se tiene el diseño correcto, no solo el que funciona, mejorar el aprendizaje individual y el aprendizaje en equipo, mejorar la modificabilidad del código y facilitar la adopción de alternativas de diseño mejoradas incluso. (Shalloway & Trott, 2000)

Patrón de diseño Decorador

El Propósito de éste patrón es agregar responsabilidades adicionales a un objeto dinámicamente. El patrón decorador proporciona una alternativa flexible a las subclases para extender su funcionalidad. (Gama, Helm, & Johnson, 1997)

Incentivación. En ocasiones queremos agregar responsabilidades a objetos individuales, y no a una clase entera. Una herramienta para interfaz de usuarios, por ejemplo, debe de permitir agregar propiedades como bordes o comportamientos como "scrolling" o barra de desplazamiento, a cualquier componente de la interfaz de usuario. (Gama, Helm, & Johnson, 1997)

Una manera de agregar responsabilidades es con herencia. Heredar un borde de otra clase pone un borde alrededor de todas las subclases de la instancia. Esto no es flexible, sin embargo, por que la elección del borde se hace estáticamente.

Un cliente no puede controlar como y cuando decorar el componente con un borde. (Gama, Helm, & Johnson, 1997)

Una manera más flexible de abordar esto es encapsular el componente en otro objeto que agrega el borde. El objeto encapsulado es llamado "decorator". El decorator se ajusta a la interface del componente y lo decora de tal manera, que su presencia es transparente al componente del cliente. (Gama, Helm, & Johnson, 1997) Se muestra la estructura del patrón en la Figura 2 Diagrama UML estructura de patrón "Decorator".

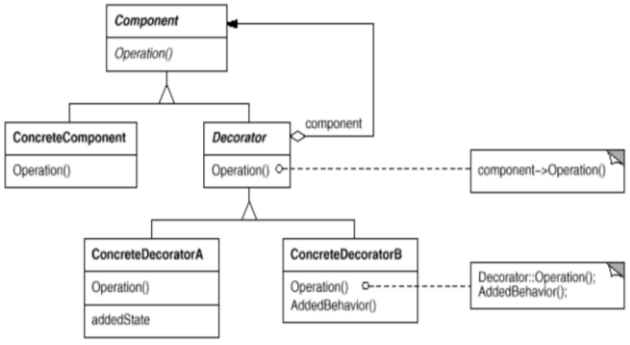

Figura 2 Diagrama UML estructura de patrón "Decorator". Elaboración propia

Componentes del diagram UML:

- "Component". Define la interfaz para objetos que se les puede agregar responsabilidades dinámicamente.

- "ConcreteComponent". Define un objeto al cual responsabilidades adicionales se le pueden ser agregadas.
- "Decorator". Mantiene una referencia a un objeto "Component" y define una interfaz que compone la interfaz del objeto "Component".
- "ConcreteDecorator". Agrega responsa-bilidades al componente

Implementación

Como se mencionó anteriormente, tomaremos una clase del modelo "Production Schedule" del estándar ANSI/ISA-95, e integraremos un procedimiento y un atributo mediante la utilización del patrón de diseño "Decorator". Para ejemplificar la implementación supondremos que contamos con las clases concretas de los modelos del estándar para un desarrollo de software y es necesario guardar la información de sus atributos en una base de datos, también se necesita contar con la posibilidad de controlar múltiples plantas, recordando que las clases de los modelos del estándar cuentan con solo los atributos definidos en el estándar. Lo anterior se resuelve mediante un procedimiento "Insertar" , el cual supondremos que reúne los atributos necesarios y los inserta en una base de datos, para la operación multi-planta, agregaremos un atributo "Planta", el cual contendrá el valor que identifica la planta para la cual corresponde la planeación ("Production Schedule"). La Figura 3 muestra el diagrama resultante de la implementación:

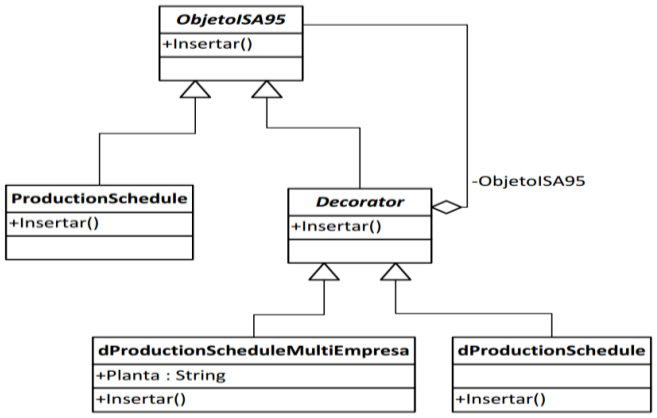

Figura 3 Diagrama UML Modelo de patrón "Decorator" para clase "Production Schedule".

Componentes del diagrama UML:

- ObjetoISA95 ("Component"). Define la clase abstracta o interfaz que contiene el procedimiento Insertar.
- Production Schedule ("ConcreteComponent"). Clase concreta del estándar ANSI/ISA-95, a la cual se le agregara o "decorara" el procedimiento Insertar por medio de la implementación de la interfaz o clase abstracta ObjetoISA95.
- "Decorator". Clase abstracta para mantener una referencia a ObjetoISA95.
- dProductionScheduleMultiEmpresa ("ConcreteDecorator"). Clase que contiene la "Decoracion". Esta clase contiene un atributo agregado "Planta" el cual no está definido dentro de la clase del estándar, pero que sería útil para una empresa que requiera de un sistema de software que controle varias plantas. Y por supuesto el procedimiento Insertar, el cual será compartido por todos los objetos que implementen ObjetoISA95.
- dProductionSchedule ("ConcreteDecorator"). Clase decorada únicamente con el procedimiento Insertar, para un ambiente de una sola planta.

Codigo fuente en Visual Basic.net para la implementación

Clase abstracta ObjetoISA95:

```
Public MustInherit Class ObjetoISA95
   Procedimiento para sobrecargar
   Public MustOverride Sub Insertar()
End Class
```

Clase Production Schedule:

```
Public Class ProductionSchedule : Inherits ObjetoISA95
   Public ID As String
   Public Description As String
   Public StartTime As Date
   Public EndTime As Date
   Public PublishedDate As Date
   Public Location As String
   Public ElementType As String
   Overrides Sub Insertar()
      Console.WriteLine("Informacion insertada para production schedule " & ID & ".")
   End Sub
End Class
```

Clase "Decorator":

```
Public MustInherit Class Decorator : Inherits ObjetoISA95
   Protected objISA95 As ObjetoISA95
   Sub New(ByVal _objISA95 As ObjetoISA95)
      Me.objISA95 = _objISA95
   End Sub
   Overrides Sub Insertar()
      If Not objISA95 Is Nothing Then
         'Llamado al procedimeinto Insertar del componente concreto
         objISA95.Insertar()
      End If
   End Sub
End Class
```

Clase dProductionScheduleMultiEmpresa:

```
Public Class dProductionScheduleMultiEmpresa : Inherits Decorator
   Public objProductionSchedule As New ProductionSchedule
   Sub New(ByVal objISA95 As ObjetoISA95, ByVal _Planta As String)
      MyBase.New(objISA95)
      objProductionSchedule = objISA95
      Planta = _Planta
   End Sub
   'Identificador de planta
   Private Planta As String
   'Sobre carga metodo insertar
   Overloads Sub Insertar()
      'Agrega referencia a planta
      InsertarReferenciaPlanta()
      'Llamado al procedimeinto Insertar del componente concreto
      MyBase.Insertar()
   End Sub
   'Procedimiento para insertar referencia a planta
   Sub InsertarReferenciaPlanta()
      Console.WriteLine("Referencia en planta " & Planta & ".")
   End Sub
End Class
```

Clase dProductionSchedule:

```
Public Class dProductionSchedule : Inherits Decorator
   Sub New(ByVal objISA95 As ObjetoISA95)
      MyBase.New(objISA95)
   End Sub
   'Identificador de planta
   Private Planta As String
   'Sobre carga metodo insertar
   Overloads Sub Insertar()
      'Llamado al procedimeinto Insertar del componente concreto
      MyBase.Insertar()
      'Procedimientos adicionales.
      Console.WriteLine("Procedimientos adicionales")
   End Sub
End Class
```

Resultados y Conclusiones

Al implementar el patrón de diseño "Decorator" a uno de los modelos del estándar ANSI/ISA-95, podemos observar la utilidad de extender el modelo agregando nuevos métodos, funciones o atributos sin necesidad de modificar la clase concreta desarrollada a partir del modelo en sí.

ISSN 2444-4995

Como se muestra en la Figura 4, Resultado de la implementación del patrón "Decorator", se le agrego una nueva función llamada Ïnsertar" a la cual se le dio la responsabilidad de: identificar la planta a la cual pertenece la planeación o sea ("Production Schedule") o simplemente realizar procedimientos adicionales.

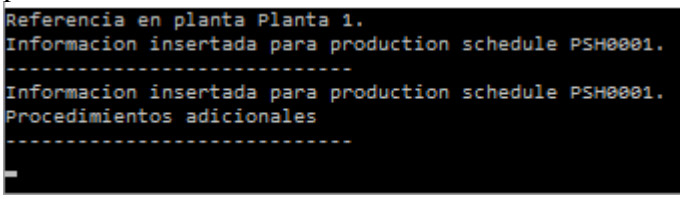

Figura 4 Resultado de la implementación del patrón "Decorator"

El utilizar patrones de diseño da una gran ventaja de escalabilidad en el desarrollo de software.

En conclusión, los patrones de diseño son herramientas eficaces y sencillas para resolver problemas persistentes en todos los desarrollos de software. Al combinar estos con el estándar ANSI/ISA-95 se conjugan dos herramientas poderosas para obtener desarrollos con buenas practicas de software, lo cual produce mejores resultados.

Referencias

Chen, C. (2003). *TAIWAN ENTERPRISE DATA OPERATION REQUIREMENT ANALYSIS: MANUFACTURING VERSION.*

Cruz Renteria, J. R., Zaragoza Peñuñuri, J. A., & Garcia Alva, S. (2011). Prototipo de Software MES que integra islas de automatizacion en manufatura utilizando ISA-95. *Coloquio de Investigacion Multidisciplinaria.* Orizaba Veracruz Mexico.

Dominguez Machuca, J. (1995). *DIRECCION DE OPERACIONES: ASPECTOS TACTICOS Y OPERATIVOS EN LA PRODUCCION Y SERVICIOS.* MADRID: McGraw-Hill.

Gama, E., Helm, R., & Johnson, R. (1997). *Design Patterns: Elements of Reusable Object-Oriented Software.* ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES.

ISA. (2010). *Enterprise-Control System Integration - Part 2: Object Model Atributes* (2010 ed.). Triangle Park North Carolina: ISA.

Nagalingam , G. (2007). CIM - Still the solution for manufacturing industry. *Elsiever Ltd*, 334 y 338.

Scholten, B. (2007). *The Road to integration: A guide to applying the ISA-95 Standar in Manufacturing.*

Shalloway , A., & Trott, J. R. (2000). *Design Patterns Explained.* Software Pattern Series.

Thiel, S. (2010). *Enterprise Application Design Patterns: Improved and Applied.* Montreal, Quebec, Canada: Concordia University.

Vasilev, P. (2015). ANSI/ISA 95 FINAL CAPACITY SCHEDULING FOR SOFTWARE INDUSTRY. *IFAC (International Federetation of Autoatic Control) Hosting by ELSEVIER Ltd.* .