Programación II. Rúbrica del proyecto

Marzo de 2019

Descripción	Experto	En desarrollo	Incipiente	Insuficiente
Aplicas los llamados patrones generales de asignación de responsabilidades; la aplicación del conjunto de estos patrones, simultáneamente con los principios mencionados a continuación, da como resultado programas orientados a objetos con las características deseadas, es decir, con una distribución razonable de responsabilidades. Es observable en la medida que cada decisión de diseño puede ser explicada en términos del patrón.	Aplicas correctamente en tus programas cada vez que hay oportunidad los siguientes patrones: Expert, Polimorfism, Low Coupling, High Cohesion, Don't Talk to Strangers y Creator; lo haces explícito consistentemente -mediante comentarios en el código-; y argumentas correctamente sobre la aplicación	Aplicas correctamente solo algunos de esos patrones o no siempre los aplicas cuando hay oportunidad; no siempre lo haces explícito; o no argumentas correctamente sobre su aplicación	No aplicas correctamente varios de esos patrones o casi nunca los aplicas cuando hay oportunidad; no siempre lo haces explícito; o no puedes argumentar correctamente sobre los patrones que sí aplicas	No aplicas correctamente ninguno de estos patrones; casi nunca lo haces explícito; o no puedes argumentar correctamente sobre los patrones que sí aplicas
Respetas los llamados principios sólidos, un acrónimo proveniente del inglés formado por los nombres de los principios; la aplicación del conjunto de estos principios, simultáneamente con los patrones antes mencionados, da como resultado programas orientados a objetos con las características deseadas, es decir, con una distribución razonable de	Aplicas correctamente en tus programas cada vez que hay oportunidad los siguientes principios: Single Responsibility Principle, Liskov Substitution Principle, Dependency Inversion, Interface Segregation, y Open/Closed; lo haces explícito -mediante comentarios en el código-; y argumentas correctamente sobre la aplicación	Aplicas correctamente solo algunos de esos principios o no siempre los aplicas cuando hay oportunidad; no siempre lo haces explícito; o no argumentas correctamente sobre su aplicación	No aplicas correctamente varios de esos principios o casi nunca los aplicas cuando hay oportunidad; no siempre lo haces explícito; o no puedes argumentar correctamente sobre los patrones que sí aplicas	No aplicas correctamente ninguno de estos principios; casi nunca lo haces explícito; o no puedes argumentar correctamente sobre los patrones que sí aplicas

responsabilidades. Es observable en la medida que cada decisión de diseño respeta o viola alguno de estos principios.				
Diseñas clases expresando sus responsabilidades y colaboraciones en las llamadas tarjetas CRC; las tarjetas no tienen detalles de implementación y son expresadas en lenguaje natural. Es observable a través del contenido de las tarjetas.	Identificas las clases con las que crearás objetos para tus programas, las responsabilidades de hacer y conocer de esas clases, y las colaboraciones necesarias para implementar esas responsabilidades; y las expresas en la notación llamada tarjetas CRC	No siempre identificas las clases con las que crearás para tus programas, o las responsabilidades de esas clases, o las colaboraciones necesarias para implementarlas; o no siempre usas las tarjetas CRC	Casi nunca identificas las clases con las que crearás para tus programas, o las responsabilidades de esas clases, o las colaboraciones necesarias para implementarlas; o casi nunca usas las tarjetas CRC	No identificas correctamente las clases con las que crearás para tus programas, o las responsabilidades de esas clases, o las colaboraciones necesarias para implementarlas; o casi nunca usas las tarjetas CRC
Diseñas clases y objetos expresando las relaciones entre ellos utilizando un subconjunto de construcciones un lenguaje gráfico de modelado orientado a objetos; estas construcciones son: clase, interfaz, objeto, colaboración, mensaje, atributo, método, herencia, composición, cardinalidad. Es observable a través de los modelos gráficos.	Expresas en forma de modelos gráficos las clases con las que crearás objetos para tus programas, las responsabilidades de hacer y conocer de esas clases, y las colaboraciones necesarias para implementar esas responsabilidades, usando todas las construcciones	No siempre expresas en forma de modelo las clases con las que crearás para tus programas, o las responsabilidades de esas clases, o las colaboraciones necesarias para implementarlas; o usas solamente algunas de las construcciones	Casi nunca expresas en forma de modelo las clases con las que crearás para tus programas, o las responsabilidades de esas clases, o las colaboraciones necesarias para implementarlas; o usas pocas construcciones	No expresas ninguna de las clases con las que crearás para tus programas, o las responsabilidades de esas clases, o las colaboraciones necesarias para implementarlas; o usas muy pocas construcciones
Escribes un programa sintácticamente correcto utilizando un conjunto amplio de construcciones de un lenguaje de programación orientado a objetos. Es observable a través del código del programa y el resultado de la compilación.	Tus programas son sintácticamente correctos y utilizas al menos todas las construcciones disponibles en la versión 3.0 de C# -puedes usar construcciones disponibles en versiones posteriores-	Tus programas son sintácticamente correctos aunque no utilizas todas las construcciones disponibles	Tus programas son sintácticamente correctos pero utilizas pocas construcciones disponibles	Tus programas no son sintácticamente correctos o utilizas muy pocas construcciones disponibles
Traduces un conjunto de tarjetas CRC en lenguaje	Conviertes correctamente las clases con las que crearás los objetos para	No siempre conviertes correctamente las clases	Casi nunca conviertes correctamente las clases	No conviertes o conviertes incorrectamente las clases

natural a un modelo gráfico en un lenguaje de notación orientado a objetos; y viceversa. Es observable a través de la comparación entre el contenido de las tarjetas y la correspondencia con el modelo gráfico resultante.	tus programas de código C# a tarjetas CRC o modelos gráficos; y viceversa	con las que crearás los objetos para tus programas de código C# a tarjetas CRC o modelos gráficos; o viceversa	con las que crearás los objetos para tus programas de código C# a tarjetas CRC o modelos gráficos; o viceversa	con las que crearás los objetos para tus programas de código C# a tarjetas CRC o modelos gráficos; o viceversa
Traduces un modelo gráfico en un lenguaje de notación orientado a objetos a un programa en un lenguaje de programación orientado a objetos; y viceversa. Es observable a través de la comparación entre el modelo gráfico y la correspondencia con el código del programa resultante.	Expresas correctamente todas las siguientes construcciones del lenguaje de programación en modelos gráficos y viceversa: objeto, colaboración, mensaje, clase, interfaz, atributo, método, herencia, composición	No expresas todas las construcciones correctamente en ambos sentidos, o las expresas correctamente pero en un solo sentido; expresar en un sentido es ir de modelos gráficos a lenguaje de programación, o viceversa	Expresas solo unas pocas construcciones correctamente, en un sentido o en otro	No expresas ninguna construcción correctamente
Conoces al menos un catálogo de patrones de diseño orientado a objetos y sabes cómo utilizarlo. Es observable en la medida que puedan hacer referencia al origen de sus decisiones.	Puedes encontrar soluciones para problemas de diseño consistentemente buscando en uno o más catálogos de patrones; y haces referencia explícita al catálogo	Puedes encontrar a veces soluciones para problemas de diseño buscando en uno o más catálogos; no siempre haces referencia explícita al catálogo	Casi nunca puedes encontrar soluciones para problemas de diseño en catálogos de patrones; o casi nunca haces referencia explícita al catálogo	No conoces ningún catálogo de patrones; o no haces referencia explícita a los patrones de catálogos que conoces
Reconoces la oportunidad para, y eres capaz de, aplicar correctamente alguno de los patrones de diseño de los catálogos de patrones que conoces. Es observable en función de la correspondencia entre la estructura del código fuente del programa y la estructura definida en algún patrón de diseño. La redacción	Conoces uno o más catálogos de patrones y aplicas correctamente en tus programas cada vez que hay oportunidad dos o tres patrones del catálogo; lo haces explícito; y argumentas correctamente sobre la aplicación	Conoces uno o más catálogos de patrones pero no siempre aplicas patrones del catálogo cuando hay oportunidad; no siempre lo haces explícito; o no argumentas correctamente sobre su aplicación	Conoces al menos un catálogo de patrones pero no aplicas correctamente varios de los patrones del catálogo o casi nunca los aplicas cuando hay oportunidad; no siempre lo haces explícito; o no puedes argumentar correctamente sobre los patrones que sí aplicas	No conoces ningún catálogo de patrones o no aplicas correctamente ninguno de los patrones del catálogo que conoces; casi nunca lo haces explícito; o no puedes argumentar correctamente sobre los patrones que sí aplicas

es flexible respecto a qué patrones utilizar.				
Manejas correctamente excepciones; es decir, utilizas los bloques de control de excepciones de la forma adecuada, creas una jerarquía de excepciones adecuada para los errores que pueda encontrar tu programa. Es observable examinando el código fuente en búsqueda de bloques try…catch…finally y clases que herede de Exception.	En todos los casos en los que pueden ocurrir excepciones manejables usas correctamente bloques try…catch; en todos los casos en los que debes asegurar la ejecución de código ante excepciones usas correctamente bloques try…finally; y creas correctamente tus propias excepciones	En la mayoría de los casos en los que pueden ocurrir excepciones manejables usas correctamente bloques try…catch; en la mayoría los casos en los que debes asegurar la ejecución de código ante excepciones usas correctamente bloques try…finally; y habitualmente creas correctamente tus propias excepciones	En algunos casos en los que pueden ocurrir excepciones manejables usas correctamente bloques try…catch; en algunos casos en los que debes asegurar la ejecución de código ante excepciones usas correctamente bloques try…finally; y a veces creas correctamente tus propias excepciones	No usas bloques try…catch consistentemente en forma correcta cuando pueden ocurrir excepciones; o no usas bloques try…finally consistentemente en forma correcta cuando necesitas asegurar la ejecución de código ante excepciones; o no creas consistentemente tus propias excepciones
Creas programas que son fáciles de mantener. La facilidad de mantenimiento es el grado de efectividad y eficiencia con el cual un programa pude ser modificado por quienes lo mantienen [adaptado de ISO/IEC 25010:2011]	Puedes modificar los programas que creas efectivamente [sin efectos secundarios inadvertidos] y eficientemente [rápidamente] de forma consistente	Puedes modificar los programas que creas efectivamente [sin efectos secundarios inadvertidos] o eficientemente [rápidamente] la mayoría de las veces	Puedes modificar los programas que creas efectivamente [sin efectos secundarios inadvertidos] o eficientemente [rápidamente] algunas veces	No puedes modificar los programas que creas efectivamente [sin efectos secundarios inadvertidos] o eficientemente [rápidamente] la mayoría de las veces
Creas programas que son fáciles de entender porque respetan convenciones de estilo y nomenclatura en el código	Respetas consistentemente convenciones de estilo y nomenclatura en el código	La mayoría de las veces respetas convenciones de estilo y nomenclatura en el código	A veces respetas convenciones de estilo y nomenclatura en el código	No respetas casi nunca convenciones de estilo y nomenclatura en el código
Creas programas produciendo y consumiendo clases reutilizables; ver reutilización en [Measurement of Langue-Supported Reuse in Object-Oriented and Object-Based Software]	Creas clases que reutilizas o reutilizas clases existentes de forma consistente	La mayoría de las veces creas clases que reutilizas o reutilizas clases existentes	Algunas veces creas clases que reutilizas o reutilizas clases existentes	No creas clases que reutilizas o reutilizas clases existentes, o casi nunca lo haces

Puedes crear y utilizar efectivamente un repositorio de control de código fuente concurrente; esto incluye: inicializar el repositorio, clonarlo en tu espacio de trabajo local, mantener sincronizado tu espacio de trabajo local con el repositorio, crear ramas para mantener tu trabajo independiente del de los demás, consolidar tus modificaciones con las modificaciones de los demás. Es observable a partir de la traza de operaciones en el repositorio.	Todas las modificaciones al código están en ramas que consolidas de forma efectiva, nunca pierdes cambios al código propio o ajeno	La mayoría de las modificaciones al código están en ramas que consolidas de forma efectiva, aunque a veces pierdes cambios en el código propio o ajeno	Algunas modificaciones al código están en ramas que consolidas de forma efectiva, pero varias veces pierdes cambios en el código propio o ajeno	No usas ramas para las modificaciones, no las consolidas en forma efectiva, pierdes cambios en el código propio o ajeno
Puedes asociar modificaciones al código fuente en el repositorio con requerimientos o con errores, colaborando con otros programadores para validar modificaciones. Es observable a partir de la traza de operaciones en el repositorio y el vínculo con issues y bugs.	Todas las modificaciones al código están asociadas a requerimientos o errores; y pasaron por revisión de otros programadores	La mayoría de las modificaciones al código están asociadas a requerimientos o errores; y pasaron por revisión de otros programadores	Algunas de las modificaciones al código están asociadas a requerimientos o errores; y pasaron por revisión de otros programadores	No asocias modificaciones al código con requerimientos o errores; no pasas las modificaciones por revisión de otros programadores
Puedes escribir una especificación de cómo debería funcionar tu programa	Escribes casos de prueba que logran una amplia cobertura del código; los mantienes actualizados	Escribes algunos casos de prueba con cobertura parcial de código; no siempre los actualizas	Escribes pocos casos de prueba con poca cobertura del código; casi nunca los actualizas	No escribes casos de prueba, o los que escribes no tienen suficiente cobertura
Puedes poner un punto de ruptura para detener la ejecución de un programa. Es observable en el entorno integrado.	Pones puntos de ruptura que te permiten examinar la lógica del programa de forma consistente	La mayorías de las veces pones puntos de ruptura que te permiten examinar la lógica del programa	Algunas veces pones puntos de ruptura que te permiten examinar la lógica del programa	No pones puntos de ruptura que te permiten examinar la lógica del programa; o los pones pero no te lo permiten

Analizas el estado de un programa durante su ejecución; eventualmente puedes también modificarlo si el contexto de ejecución lo permite. Es observable en el entorno integrado.	Puedes ver y cambiar el valor de variables y argumentos de todos los tipos	Puedes ver el valor de variables y argumentos para la mayoría de los tipos; o cambiar el valor	Puedes ver el valor de variables y argumentos para algunos tipos; o cambiar el valor	No puedes ver ni cambiar el valor de variables y argumentos
Examinas la lógica de un programa durante su ejecución, ejecutando con mayor o menor granularidad según corresponda. Es observable en el entorno integrado.	Puedes ejecutar el programa paso a paso, entrando o no a métodos según sea necesario, de forma consistente	La mayoría de las veces puedes ejecutar el programa paso a paso, entrando o no a métodos según sea necesario	Algunas veces puedes ejecutar el programa paso a paso, entrando o no a métodos según sea necesario	No puedes ejecutar el programa paso a paso, o no puedes entrar o no a métodos