Tendencias Pasadas, Presentes y Futuras en Patrones de Software

Durante más de una década, los patrones han influido en cómo los arquitectos de software y los desarrolladores crean sistemas informáticos. Los patrones centrados en el diseño proporcionan un vocabulario para expresar visiones arquitectónicas y diseños representativos claros, concisos e implementaciones detalladas. La presentación de piezas de software en términos de sus patrones constituyentes también permite a los desarrolladores comunicarse más eficazmente, con mayor concisión y menos ambigüedad.

Desde mediados de los años 90, muchos sistemas de software -incluyendo partes importantes de las bibliotecas Java y .NET y muchas plataformas middleware- han sido desarrollados con la conciencia de los patrones. A veces los desarrolladores aplicaron estos patrones de forma selectiva para abordar desafíos y problemas específicos. Otras veces, usaron patrones de forma holística para ayudar a construir sistemas de software, desde la definición inicial de arquitecturas de línea de base hasta la realización de detalles finos. El conocimiento y la aplicación consciente de los patrones se ha convertido en un producto valioso para los profesionales del software.

Mucho ha cambiado desde que Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides ("la Gang of Four" o "GoF") publicaron Design Patterns, el libro más popular sobre patrones.1 El panorama tecnológico ha cambiado,

Los enfoques de diseño de software han evolucionado y ampliado, nuestra comprensión de los procesos de desarrollo ha madurado, y sabemos más acerca de documentar y aplicar patrones al desarrollo de software. Irónicamente, Design Patterns sigue siendo tan popular e influyente que muchos desarrolladores de software no son conscientes de cuánto ha madurado el campo o dónde encontrar publicaciones de patrones que cubran una gama más amplia de dominios y tecnologías.

La comunidad de patrones ha buscado documentar y promover buenas prácticas de ingeniería de software. En este artículo se resume la amplitud y la profundidad de los patrones en la práctica para ayudar a los desarrolladores de software y los administradores de entender dónde ha estado el campo y hacia dónde se dirige, para que pueda utilizar patrones en sus propios proyectos.

Una breve historia de patrones de software Los patrones no siempre han sido tan populares o

Tan penetrante como lo son ahora. Aunque a finales de los 80 y principios de los noventa se vieron investigaciones aisladas sobre patrones de software, los patrones no entraron en la corriente principal hasta que se publicó Patrones de Diseño. El libro describió 23 patrones derivados en gran parte de las experiencias de los autores desarrollando un solo hilo, orientado a objetos, frameworks de interfaz de usuario en Smalltalk y C ++.

Patrones de diseño sigue siendo el trabajo de patrón más popular e influyente. Numerosos libros y artículos han abordado los patrones GoF de varias maneras. Algunas publicaciones tratan de implementar los patrones en otros lenguajes de programación, como C # y Java. Otras publicaciones reelaboran e integran los patrones GoF para contextos específicos de aplicaciones, como la informática distribuida, la seguridad y los sistemas embebidos en tiempo real.7-9

En la conferencia OOPSLA de 1999 (Programación Orientada a Objetos, Sistemas, Lenguajes y Aplicaciones), James Burke señaló que la historia rara vez ocurre en el orden correcto o en el momento adecuado -el trabajo del historiador es hacer que parezca como si lo hiciera . Del mismo modo, aunque resumimos las tendencias clave de dos décadas de investigación de patrones, nuestra lista está lejos de ser exhaustiva. Para referencias más amplias, vea http: // hillside.net/patterns.

Patrones independientes y colecciones de patrones Inspirados por el éxito del GoF, gran parte de la

La investigación de patrones populares en los mediados a finales de los 90 se centró en patrones autónomos y colecciones de patrones. Los patrones autónomos son "soluciones puntuales", que abordan problemas relativamente limitados que surgen en contextos específicos. Ejemplos de patrones autónomos populares incluyen los siguientes:

■ Iterator ofrece trazado agregado sin exponer detalles de representación agregada a los llamantes.

■ La estrategia captura un comportamiento conectable.

■ Wrapper Facade encapsula las funciones y los datos de una API procedural existente en interfaces más concisas, robustas, portátiles y cohesivas orientadas a objetos.

Sin embargo, cualquier diseño significativo de software incluye inevitablemente muchos patrones, lo que hace que los patrones autónomos sean poco usuales en la práctica. Una colección de patrones es la presentación más obvia de patrones múltiples. La colección de patrones más ambiciosa hasta la fecha es el trabajo en curso de Grady Booch sobre el Manual de Arquitectura de Software (www.booch.com/architecture), que hace referencia a aproximadamente 2.000 patrones.

La mayoría de las colecciones de patrones son mucho más modestas en tamaño y ambición, a menudo numerando decenas de patrones y enfocándose en un tipo particular de problema, contexto o sistema.

Muchos patrones autónomos fueron creados inicialmente para -y perfeccionaron los talleres de los escritores en varias conferencias de lenguaje de programación (PLoP). En un taller de escritores, los autores leen los patrones de los demás y discuten sus fortalezas y debilidades para ayudar a mejorar el contenido y el estilo. Puede encontrar los patrones más maduros de estas conferencias en los libros de patrones de diseño de programas de Addison-Wesley o en los procedimientos de conferencias en línea de PLoP.

Relaciones de patrones

Los patrones que representan los focos para la discusión, las soluciones puntuales o las ideas de diseño localizadas se pueden utilizar de forma aislada con cierto éxito. Los patrones son generalmente gregarios, sin embargo, en que forman relaciones con otros patrones. Cualquier aplicación o biblioteca dada usará muchos patrones relacionados. Los tipos más comunes de relaciones de patrones incluyen complementos, compuestos y secuencias.

Complementos de patrones. En éstos, un modelo proporciona el ingrediente faltante para otro patrón o una solución alternativa a un problema relacionado. Los complementos cooperativos tienen como objetivo hacer el diseño resultante más completo y equilibrado. Por ejemplo, Método de eliminación complementa el Método de fábrica tratando la destrucción de objetos y la creación en el mismo diseño. Esta combinación complementaria apoya diseños que encapsulan las políticas del ciclo de vida de los recursos, como la agrupación. Los patrones también pueden competir entre sí. Por ejemplo, Batch Method, una alternativa a Iterator, puede acceder a los elementos de un objeto agregado sin exponer su implementación subyacente. Batch Method es más adecuado para entornos distribuidos (donde el acceso remoto hace que el uso de un iterador convencional sea prohibitivo) porque accede a los elementos de un agregado a granel, reduciendo los costos de red de ida y vuelta.

Compuestos patrón. Estos captan subconjuntos recurrentes de patrones que son comunes e identificables lo suficiente que los desarrolladores de software pueden tratarlos como una sola decisión en respuesta a un problema recurrente. Por ejemplo, aplicar dos patrones juntos, como un Comando implementado como un Compuesto, es tan común que puede nombrar los patrones como una sola entidad, como Comando Compuesto. Otro ejemplo es Batch Iterator, que une dos patrones complementarios, Iterator y Batch Method, para acceder de forma remota a los elementos de agregados con gran cantidad de elementos. Un iterador por lotes refina la localización basada en la posición de un iterador con un método por lotes para acceder a muchos elementos, pero no a todos.

Secuencias de patrones. Estos generalizan la progresión de patrones y muestran cómo establecer un diseño uniendo patrones predecesores para formar parte del contexto de cada patrón sucesivo. Por ejemplo, en nuestro trabajo anterior, presentamos una secuencia de patrones de middleware de comunicación que se une al Broker, Layers, Wrapper Facade, Reactor, Acceptor-Connector, Half-Sync / Half-Async, Monitor Object, Strategy, Abstract Factory y Component Patrones del configurador.2 Una secuencia de patrones captura cómo un diseño o situación se despliega patrón por patrón. También puede ilustrar tal progresión con una historia de patrón, que es un ejemplo concreto de aplicar una secuencia de patrones.

Lenguajes de patrones

Ya en 1995, los principales autores en la comunidad de patrones comenzaron a documentar grupos de patrones para dominios de desarrollo de software específicos, particularmente sistemas de telecomunicaciones. Como demostraron los primeros libros de patrones de los libros de diseño de programas, estos patrones estaban más estrechamente relacionados que los patrones autónomos anteriores y las colecciones de patrones. De hecho, algunos estaban tan estrechamente relacionados que no existían aisladamente, por lo que los autores los organizaron como lenguajes de patrones en los que cada patrón se construyó y tejió otros patrones en el lenguaje. Podemos ver los lenguajes de patrones como la extrapolación lógica de las relaciones de patrón descritas anteriormente. Del mismo modo, vemos las relaciones de patrón como aspectos restringidos y simplificados y subconjuntos de lenguajes de patrones.

Los lenguajes de patrones tienen como objetivo proporcionar soporte holístico para el uso de patrones para desarrollar software para dominios técnicos o de aplicaciones específicos, tales como comercio electrónico o middleware de comunicación. Consiguientemente, alistan patrones múltiples para cada problema potencial y los tejen juntos para definir un proceso de desarrollo de software generativo, específico del dominio y del patrón-orientado.

Muchos lenguajes de patrones son elaboraciones y descomposiciones de patrones autónomos y secuencias de patrones. Es natural que estos lenguajes se desarrollen y se desarrollen recursivamente y fortalezcan las cualidades centrales de sus patrones constituyentes. Para un ejemplo, vea Remoting Patterns, 3 que descompone el patrón Broker10 en una descripción completa de lo moderno

Arquitecturas middleware de comunicación. Broker no podía ser tan expresivo cuando se describía como un patrón autónomo, pero el lenguaje del patrón Remoting reveló esta generatividad.

**Dominios y tecnologías documentados por patrones**

Desde el principio, los patrones y marcos han sido íntimamente conectados, demostrados por los muchos ejemplos de marco Patrones de diseño utilizados para motivar patrones. Un marco es un conjunto integrado de componentes que colaboran para proporcionar una arquitectura de software comoditizada para una familia de aplicaciones relacionadas. Los marcos maduros presentan una alta densidad de patrones, haciendo de los patrones una herramienta descriptiva ideal para el desarrollo, la evolución y la comprensión de los marcos. Más allá del interés en los frameworks, los siguientes dominios y tecnologías son focos populares para los autores de patrones.

Computación distribuída. Nuestro trabajo anterior presenta un extenso lenguaje de patrones para la construcción de sistemas de software distribuidos.2 El lenguaje de patrones conecta más de 250 patrones que abordan temas que van desde la definición y selección de una arquitectura de línea de base y una infraestructura de comunicación apropiadas hasta interfaces de componentes especificas y sus implementaciones e interacciones. El lenguaje de patrones también aborda aspectos técnicos clave tales como adaptación y extensión, concurrencia, acceso a bases de datos, manejo de eventos, sincronización y administración de recursos.

Idioma y dominios específicos del idioma.

Varios estilos de programación han evolucionado y surgido en la última década, incluyendo el desarrollo de software orientado a aspectos, el diseño impulsado por el dominio, el desarrollo de software impulsado por el modelo y la programación generativa. Cada uno tiene sus propios patrones y mejores prácticas que distinguen la programación en él de la programación en otros estilos. Los patrones y las colecciones de patrones existen para estos estilos. Las expresiones idiomáticas de programación han sido un enfoque común para patrones publicados, desde Smalltalk a Python y desde C ++ a C #.

Tolerancia a fallos y manejo. A medida que integramos cada vez más software en sistemas de seguridad de misión y seguridad, necesitamos técnicas robustas para satisfacer los requisitos de fiabilidad del usuario. Por lo tanto, la tolerancia a fallos y los patrones de gestión han sido un foco activo durante la última década. Varios libros recientes contienen patrones y lenguajes de patrones que abordan la tolerancia a fallos y la gestión de fallas para sistemas con estrictos requisitos operacionales. 4,7

Seguridad. Esto se ha convertido en un tema popular para patrones y sistemas de software en general. Por ejemplo, Security Patterns documenta una serie de patrones y lenguajes de patrones para áreas relacionadas con la seguridad, tales como autenticación, autorización, integridad y confidencialidad.

Esos patrones se documentaron en publicaciones de patrones anteriores, tales como los procedimientos de conferencias de PLoP. Otro texto reciente, Core Security Patterns, cubre la seguridad del software en red.6

Sistemas embebidos. Cada vez más, aplicaciones tales como marcapasos, controladores de centrales eléctricas y sistemas de aviónica de vuelo integran inteligencia en dispositivos y sistemas físicos. Debido a que estas aplicaciones están inextricablemente conectadas con el entorno físico, los desarrolladores deben diseñarlas para satisfacer las demandas y limitaciones físicas -como la dinámica, el ruido, el consumo de energía y el tamaño físico- de manera oportuna. La investigación en esta área documenta patrones para tratar estas restricciones sin perder los beneficios de la abstracción.7,8

Proceso y estructura organizativa. Muchas investigaciones lingüísticas de patrones se han centrado en mejorar los procesos de desarrollo de software y las organizaciones. Algunos libros recientes abordan ciertos tipos de procesos de desarrollo de software, tales como desarrollo de software distribuido, ágil, controlado por pruebas y dominio, así como refactorización de software y reingeniería.

Los patrones de desarrollo de software ágil integran patrones existentes en procesos y organizaciones de desarrollo de software en un lenguaje de patrones interconectado.13

Educación. Patrones y lenguajes patrón son vehículos populares para aprender el arte de la enseñanza de la programación y la ingeniería de software. Un libro de PLoPD compila varias publicaciones en esta área, 14 al igual que los Patrones Pedagógicos

Project (www.pedagogicalpatterns.org), que proporciona un foro para difundir y discutir los patrones de enseñanza, con un enfoque en la enseñanza de la práctica efectiva de software.

Donde están los patrones ahora

En un sentido, la comunidad de patrones ha cumplido su objetivo de documentar y promover buenas prácticas de ingeniería de software. Después de más de una década de experiencia en minería, documentación y aplicación de patrones, la comunidad ha establecido patrones en la práctica de desarrollo de software convencional. Muchos proyectos de software de producción y currículos universitarios utilizan patrones conscientemente. Además, hay mucha más experiencia con el formato utilizado para documentar patrones, como una apreciación más profunda de la importancia de capturar las fuerzas que configuran los diseños, 15 en comparación con el formato utilizado en los patrones de diseño. Esperamos que estas tendencias continúen mientras los desarrolladores comprenden cada vez más los patrones básicos y la riqueza de patrones y lenguajes de patrones en la literatura más amplia.

A pesar de que muchos sistemas de software han logrado aplicar patrones intencionalmente, también han ocurrido fallas debido a malentendidos comunes sobre patrones, incluyendo lo que son y no son sus propiedades, propósito, público objetivo, beneficios e inconvenientes. La comunidad de modelos ha buscado durante mucho tiempo comprender las teorías, las formas y las metodologías subyacentes de los patrones y los lenguajes de patrones (y sus conceptos asociados) para ayudar a codificar el conocimiento y la aplicación efectiva de patrones de software. El trabajo más grande en esta área, Arquitectura de Software Orientada a Modelos Volumen 5, integra muchas facetas del concepto de patrón en un todo coherente.15

En comparación con la onda inicial de usuarios de patrones después de la publicación de Patrones de Diseño, los desarrolladores de software parecen entender mejor los patrones que antes, en términos de su experiencia utilizando patrones de manera efectiva en proyectos de software y su comprensión de varios aspectos del concepto de patrón. Los patrones publicados y los lenguajes patrón también han aumentado generalmente en calidad. La mayoría de los patrones y patrones publicados en los últimos años son más expresivos, comprensivos, precisos y legibles que los publicados anteriormente.

Donde pueden ir los patrones

Muchos desarrolladores de software ahora están familiarizados con los patrones y los aplican en sus dominios, pidiendo la pregunta "¿Qué queda decir sobre los patrones?" Desde 1996, hemos estado observando tendencias que modelan la literatura de patrones y la comunidad, así como la predicción El futuro de los patrones como parte de nuestro trabajo en los libros de arquitectura de software orientado a patrones. Por lo tanto, hemos hecho conjeturas sobre las tendencias futuras dado el beneficio de la retrospectiva y nuestras experiencias de descubrir, documentar y aplicar patrones en la práctica.

Esperamos ver a los desarrolladores de software documentar más tecnología y patrones específicos de dominio y lenguajes de patrones. Las tecnologías y los dominios de desarrollo de software no son todos abordados por los patrones todavía - y podría tomar décadas para cubrirlos a todos. En particular, los patrones de captura de experiencia, y para los nuevos dominios y tecnologías, los desarrolladores deben primero obtener, evaluar y codificar esta experiencia antes de que puedan documentar patrones efectivos. Aquí, predecimos qué dominios y desarrolladores de tecnologías abordarán primero.

Arquitectura orientada a Servicios

SOA es un estilo de organización y uso de capacidades distribuidas que diferentes organizaciones o grupos controlan y poseen. SOA no es un concepto nuevo, pero se ha convertido en una palabra de moda en los últimos años. El enfoque se basa en los principios y tecnologías de la computación distribuida y la integración de sistemas empresariales, extrayendo de un espectro de patrones existentes y lenguajes de patrones.16,17 Sin embargo, algunas tecnologías SOA, como el modelado de procesos empresariales, la orquestación de servicios, Sistemas todavía no están explorados y no están cubiertos por la literatura de patrones.

Tecnologías de software generativo

La mayoría de los patrones documentados desde los primeros años 90 han sido extraídos de software orientado a objetos escrito en lenguajes de programación de tercera generación. Los patrones y los lenguajes de patrones, sin embargo, ya han comenzado a influir y apoyar otros enfoques de desarrollo de software, particularmente el desarrollo de software orientado a aspectos y modelo. Medimos la madurez y la aceptación de un enfoque en términos del grado en que los autores publican sus patrones y mejores prácticas. Los desarrolladores han intentado capturar patrones de desarrollo de software impulsados ​​por modelos asociados con procesos y organización, modelado de dominios, arquitectura de herramientas y desarrollo de plataformas de aplicaciones.

Esperamos que los patrones y los lenguajes de patrones tengan la misma influencia en el desarrollo de software orientado a aspectos.

Sistemas distribuidos en tiempo real y embebidos Desarrollar sistemas distribuidos en tiempo real distribuidos de alta calidad

Y los sistemas embebidos es más difícil de hecho que el desarrollo de sistemas tradicionales en tiempo real y embebidos, y algo así como un "arte negro". Un próximo libro documenta patrones para desarrollar software distribuido en tiempo real y embebido basado en material de talleres de escritura de patrones. Esperamos ver más patrones para estos sistemas porque el progreso en este campo es esencial para desarrollar aplicaciones viables de misión crítica y de seguridad.

Calidad de servicio para sistemas distribuidos basados ​​en COTS

Para reducir el tiempo y el costo del ciclo de desarrollo, los proyectos desarrollan cada vez más sistemas distribuidos que utilizan múltiples capas de hardware COTS, sistemas operativos y componentes de middleware. Sin embargo, es difícil configurar sistemas basados ​​en COTS que puedan satisfacer simultáneamente varias propiedades de QoS, como seguridad, oportunidad y tolerancia a fallos. A medida que los desarrolladores e integradores continúan dominando las complejidades de proporcionar garantías de QoS de extremo a extremo, esperamos ver un aumento en los patrones documentados que ayudan a otros a configurar, monitorear y controlar sistemas distribuidos basados ​​en COTS que poseen una gama de propiedades de QoS interdependientes .

Sistemas móviles

Las redes inalámbricas se han vuelto omnipresentes, y los dispositivos informáticos integrados se están volviendo cada vez más pequeños, ligeros y capaces. Del mismo modo, ahora puede acceder a servicios de Internet, desde navegación Web a banca en línea, desde sistemas móviles. Sin embargo, los sistemas móviles presentan muchos desafíos, como la gestión de ancho de banda y potencia bajos y variables, la adaptación a frecuentes interrupciones en la conectividad y la calidad del servicio, los protocolos divergentes y el mantenimiento de la coherencia de la caché a través de nodos de red desconectados. Esperamos que los desarrolladores de sistemas móviles experimentados documenten sus conocimientos en forma de patrón para ayudar a satisfacer la creciente demanda de mejores prácticas en esta área.

Arquitectura de software

A pesar de un aumento en los lenguajes de patrones documentados, la industria del software no tiene paralelo con los manuales completos de otras disciplinas de diseño. Aunque la literatura de patrones existentes ha progresado constantemente hacia la creación de manuales para ingenieros de software, la meta final no se ha alcanzado. Como mencionamos anteriormente, el Manual de Arquitectura de Software contiene miles de patrones, exponiendo sus roles y relaciones esenciales y permitiendo comparaciones entre dominios y estilos arquitectónicos.

Interacción de grupo

Aunque muchos desarrolladores han publicado modelos de interacción hombre-computadora en los últimos años, todavía no tenemos una visión integrada de estos patrones. Además, los patrones no abarcan todavía todas las áreas de interacción grupal en un entorno electrónico, por ejemplo, mundos virtuales y juegos masivos multijugador. Dada la creciente demanda de apoyo de colaboración electrónica, esperamos ver más patrones y lenguajes de patrones en HCI en un futuro próximo.

web 2.0

La Web proporciona cada vez más el contexto para modelos empresariales más dinámicos y abiertos, donde el aprovechamiento de la inteligencia colectiva se convierte en el medio de los negocios y la próxima generación de Web (es decir, la Web 2.0) se convierte en el medio. El conjunto inicial de patrones Web 2.0 de Tim O'Reilly (www.oreillynet.com/pub/a/oreilly/tim/ news / 2005/09/30 / what-is-web-20.html) recoge algunas prácticas clave emergentes y comprobadas .

Sistemas de transacciones comerciales y comercio electrónico

Muchos sistemas de información de negocios, como los de contabilidad, nómina, inventario y facturación, se basan en transacciones. Las reglas para el procesamiento de transacciones son complejas y deben ser flexibles para reflejar nuevas prácticas de negocios y fusiones. Los sistemas empresariales también deben manejar volúmenes cada vez más grandes de transacciones en línea. El crecimiento meteórico del comercio electrónico en la Web ha expuesto muchos sistemas de negocio a negocio directamente a los consumidores. A pesar de la importancia de estos sistemas, se ha escrito relativamente poco acerca de su análisis robusto y seguro, arquitectura o patrones.

Estructura del proceso y de la organización

La creciente adopción de procesos de desarrollo ágil sugiere que la literatura de patrones correspondiente continuará creciendo. Algunos trabajos se centrarán en aspectos de macroprocesos, como el ciclo de vida general y la interacción comercial, y algunos se centrarán en aspectos de microprocesos, como el desarrollo de pruebas de prueba, la refactorización y el uso de herramientas.

La cuadrilla de cuatro

Muchos libros y artículos se refieren a los patrones GoF, sin fin a la vista. Los autores continuarán satisfaciendo la demanda de reflexiones sobre el trabajo del GoF, desde ensayos sobre ingredientes faltantes, soluciones alternativas y el alcance apropiado de los patrones hasta discusiones sobre si un determinado patrón está obsoleto o no es un patrón en absoluto. También se habla de una segunda edición de Patterns de diseño, aunque una fecha de publicación no se ha finalizado.

Teoría de patrones

El trabajo sobre la teoría subyacente a los conceptos de patrón continuará, centrándose en una mejor comprensión de las facetas conocidas, tales como secuencias de patrones, y explorando nuevos puntos de vista, como los marcos de problemas (que captan, describen y nombran tipos recurrentes de problemas). Aunque todavía hay mucho margen para consolidar, aclarar y comunicar conceptos teóricos, creemos que las tendencias más interesantes se asociarán con los dominios en los que se documentan los patrones, más que en teoría.

Sin duda, los autores publicarán patrones y lenguajes de patrones para áreas distintas a las que hemos mencionado, consideramos estos temas como los más prometedores. Naturalmente, nuestras predicciones incluyen una dosis saludable de incertidumbre y están basadas en nuestro conocimiento de la comunidad de patrones, sus intereses y actividades de investigación en curso y planeados (en la medida en que los conocemos) y las direcciones futuras con las que estamos involucrados.