¿Qué dice su software sobre usted?

Artículo cedido por National Instruments

www.ni.com

Autor: Simon Hogg, director de producto senior de LabVIEW de NI

El mundo del software es a menudo un lugar divertido. Es muy raro que para cualquier tarea dada haya solo una solución. Para una tarea determinada existen miles de diferentes soluciones posibles y no es raro que varias implementaciones diferentes de esencialmente la misma solución compitan entre si. Mac OS X, Linux y Windows son todos unos sistemas operativos de escritorio muy capaces cada uno de hacer frente a la gran mayoría de las necesidades de la mayor parte de los usuarios de ordenadores. Sin embargo, la gente a menudo se identifica como usuaria de "Mac" o de "Windows" como si fuera parte de su identidad (tanto es así que Apple produjo una serie de cuñas comerciales tituladas "Soy un Mac", donde un joven a la moda se enfrentaba a un viejo y torpe "PC"). Hay una larga lista de editores de texto, navegadores Web, editores de fotografía, lenguajes de programación y otros, pero el software que se elige usar dice mucho acerca de uno mismo y lo que le importa.

Todos los sistemas de prueba y medida modernos tienen componentes de software y el software que se elige debe estar alineado con tus prioridades (o las de los proyectos/empresas). Los ingenieros más eficaces son capaces de utilizar diferentes softwares con el fin de elegir la mejor herramienta para el trabajo en ese momento, mientras que otros ingenieros tienen preferencia por una herramienta que, a pesar de que podría no ser la mejor para el trabajo, funciona bien debido a la maestría del usuario con la misma. Luego están los ingenieros que sufren al tratar de encajar clavijas cuadradas en agujeros redondos. Estos toman malas decisiones debido a la terquedad, la ignorancia, o la fe fuera de lugar y son a menudo la causa de incumplimiento de plazos, resultados pobres, y problemas de mantenimiento.

Puede ser útil pensar que las muchas opciones de software a las que los ingenieros se enfrentan están comprendidas dentro de estas tres grandes categorías:

- Las aplicaciones de software de función fija
- Los lenguajes de programación de propósito general
- El software que combina elementos de ambos

Aplicaciones de software de función Fija – Son rápidas y fáciles de usar, pero son una opción restrictiva

La principal y más popular categoría de software, con mucha diferencia sobre las demás, son las aplicaciones de software de función fija. Este tipo de software se vende o distribuye para hacer un trabajo específico y sólo puede funcionar de una manera determinada. El usuario evalúa las diferentes opciones y escoge una que tiene el conjunto de características más cercano a las características que desea y tiene que vivir con el hecho de que no podrá obtener todo lo que desearía. En el caso de paquetes de software de oficina, navegadores de web y editores de fotografía esta decisión tiene mucho sentido.

Millones de personas por ahí comparten esencialmente los mismos objetivos y un mismo producto puede satisfacer posiblemente la mayor parte de sus necesidades comunes.

La ventaja de comprar software comercial (o "fuera de la Internet", como es el caso más frecuente) es sustancial. Resulta mucho más rápido evaluar y comprar algo que ya existe en lugar de especificar y crear algo completamente nuevo. Como usuario final se está protegido de los riesgos de desarrollo del software, tales como la retirada de un sistema operativo antiguo o de la adopción de un nuevo estándar. Si se produce uno de estos cambios tecnológicos sólo hay que pagar para actualizar el software o cambiar al proveedor que mejor se adapte a las necesidades. El coste del cambio es mucho menor que si uno mismo hubiera desarrollado el software y por lo general, es seguro asumir que el proveedor puede distribuir los costes de mantenimiento sobre una gran base de usuarios. Siendo uno de los muchos usuarios de un software idéntico se tiene también la ventaja de que los materiales de formación existentes se pueden aprovechar para ponerse al día rápidamente uno mismo y el equipo.

La desventaja de esta categoría de software es que se está a menudo limitado a las características que están disponibles en el mercado. En el caso del software dirigido al mercado masivo, como los sistemas operativos y los paquetes de software de oficina, hay una serie de opciones con conjuntos de características ampliables y calidad impresionante (o por lo menos ofrecen parches y actualizaciones frecuentes).

Para las tareas más especializadas, la cantidad y calidad de estas opciones disminuye a medida que el tamaño del mercado se hace más pequeño. Los proveedores de software tienen que ganar suficiente dinero de sus clientes para cubrir el coste de diseñar, desarrollar, probar y distribuir el software, por lo que, naturalmente, las herramientas que pueden ser provechosamente vendidas a la mayor parte de la gente obtienen la mayoría de las inversiones, lo que resulta en una mayor calidad. Para tareas muy especializadas, puede ser que ningún software disponible en el comercio haga exactamente lo que se necesita y al final haya que rellenar las partes del proceso con trabajo manual.

Esto puede ser muy frustrante cuando se toma en cuenta el aumento de los costes, errores e incomodidad general que ocurre cuando se tiene una persona que hace lo que el software debería haber estado ayudándole a hacer en primer lugar.

La compensación de las deficiencias de un software inflexible o incompatible es uno de los mayores costes ocultos en el lugar de trabajo dominado por los ordenadores de hoy en día. ¿Cuántas horas-hombre de técnicos han sido perdidas en vano tratando de convertir datos entre formatos patentados o se-

	Fixed-Function Software Applications	General-Purpose Programming Languages	Software That Combines Elements of Both
Pros	▪ Quickest to implement ▪ Easiest to learn	▪ Ultimate flexibility ▪ Many options ▪ Low acquisition cost	▪ More flexibility ▪ Reusable components ▪ Industry-familiar vendors
Cons	▪ May not be flexible enough ▪ May not exist for your use case ▪ May lack features you need	▪ High development cost ▪ Excessive development time ▪ Use-case agnostic vendors	▪ Up-front cost ▪ Required learning

cuenciando manualmente las operaciones que deberían ser fácilmente automatizadas?

En comparación con los sistemas operativos y los paquetes de software de oficina, el mercado del software de prueba y medida es relativamente pequeño. Esto significa que no hay una gran cantidad de productos de software de alta calidad disponibles que pueden satisfacer todas las necesidades.

Añadiendo a esto la realidad de que muchos científicos e ingenieros están, por su propia naturaleza, investigando o creando cosas que nadie más está haciendo, se reduce aún más la probabilidad de que exista una opción de software comercial de alta calidad que se pueda salir corriendo a comprar para hacer el trabajo.

Lenguajes de programación de propósito general - Potente pero hay que dedicarle mucho tiempo

Cuando el software que se necesita no existe, muchas personas eligen el método del extremo opuesto del espectro y escriben su propio software. Esta es una manera segura de conseguir la funcionalidad exacta que se necesita sin depender de una entidad externa que la proporcione. El software personalizado adecuado puede proporcionar también una ventaja competitiva, lo que resulta en ventajas de costes y en última instancia, más ganancias a la larga.

Una vez que se conocen exactamente las características que el software necesita, el siguiente paso es seleccionar un lenguaje de programación y adquirir o desarrollar el conjunto de habilidades necesarias para utilizar ese lenguaje de programación.

Existen muchos lenguajes de programación diferentes que pueden ayudar en este esfuerzo, que van desde los lenguajes probados desde hace tiempo como C y FORTRAN, a los lenguajes populares más recientes, como C#, Java y Python. Posiblemente, cualquiera de estos lenguajes se podría utilizar para escribir cualquier programa; ya que, todos soportan las operaciones elementales necesarias para crear las funciones más avanzadas. Las características de la sintaxis de programación y del tiempo de ejecución, tales como la gestión automática de memoria son normalmente las características definitorias de un lenguaje de programación, pero sorprendentemente la mayoría de la gente elige un lenguaje basado en otro conjunto de criterios, por ejemplo: "¿Que cantidad de código tendré que escribir?", "¿Qué tan potentes son las herramientas que soportan ese lenguaje?" y "¿A que nivel conozco ese lenguaje y sus herramientas?".

La pregunta de "¿cuánto código tendré que escribir?" es una gran pregunta que hay que hacerse al evaluar un lenguaje. No importa qué tan conciso sea un lenguaje y lo productivas que sean las herramientas, no hay nada mejor que ser capaz de usar el código que ya se ha escrito, probado y aceptado como válido para el trabajo.

Más de un programador de C ha malgastado el tiempo por una pérdida de memoria que no habría sucedido si se hubiera estado usando un lenguaje como Java o C #, que incluye un gestor de memoria como parte del tiempo de ejecución. Del mismo modo, hay usuarios de C# y Java por ahí que pasan semanas traduciendo el manual de programación de un instrumento a un driver reutilizable - una tarea que podría haber sido innecesaria si hubiera elegido otro lenguaje. Cuando se selecciona un lenguaje, el objetivo principal debería ser el de elegir uno que minimice la cantidad de tiempo que se dedica a la escritura de código que no agrega valor a la organización. Escoger un lenguaje con una gran cantidad del código en cuestión ya escrito podría ser la mejor opción.

Las herramientas que existen en torno a un lenguaje son también una consideración importante al elegir un lenguaje de desarrollo. Por ejemplo, un editor que ayude a evitar la escritura de código con errores proporcionando consejos en el momento de edición y mostrando más información sobre las funciones que se usan, reduce al mínimo la pérdida de tiempo en la depuración.

La integración con el control del código fuente y los marcos de pruebas facilitan los flujos de trabajo más rigurosos de un solo desarrollador o de varios.

Las herramientas de interfaces de usuario gráficas WYSIWYG ("lo que ves es lo que obtienes") pueden ahorrar mucho tiempo en la creación de interfaces de usuario. Además, tener un conjunto de herramientas que permitan ahorrar tiempo al realizar el mantenimiento en curso es otro motivo de preocupación.

Algunas herramientas son mejores que otras y soportan las últimas tecnologías tales como los nuevos sistemas operativos de móviles y web. La elección del lenguaje y de las herramientas incorrectas puede dejar al desarrollador en apuros para actualizar las características necesarias y trasladarlas al sistema operativo apropiado a medida que cambian las necesidades de su organización. En lugar de elevarse con la marea tecnológica, se puede

permanecer anclado a las decisiones de desarrollo realizadas en el pasado y continuar renunciando a un precioso tiempo de desarrollo para mantener el software que se ha convertido en una parte crítica de la infraestructura.

La familiaridad con el lenguaje es la principal razón citada por los ingenieros cuando se les preguntó por qué eligieron un software determinado para su aplicación más reciente. Si bien es comprensible que la gente trabaje con lo que conoce, es un poco preocupante que la idoneidad para el trabajo no sea la principal preocupación. La razón de esto es multifacética. Para muchos hay un elemento de miedo a lo desconocido.

Después de trabajar con un lenguaje durante un tiempo se obtiene una buena idea de sus puntos fuertes y débiles y se adquiere práctica sobre sus deficiencias. El cambio a otro lenguaje hace que uno quede expuesto a caer víctima de sus limitaciones. Para otros hay una aversión natural a tener que pasar de ser un experto relativo a un estado de vulnerabilidad al convertirse de nuevo en un "principiante". Quizás la causa más probable de preservar el status-quo es el impulso de la organización y de la política. El tener que convencer a la dirección de que existe una mejor solución para un problema exclusivo supone a menudo más trabajo que el que se sufre usando la herramienta equivocada para dicho trabajo.

Lo mejor de ambos mundos - Aprovechamiento y ampliación

La combinación de las ventajas del software escrito previamente con la flexibilidad del software personalizado en una organización que es lo suficientemente ágil como para dar cabida a ambos es el método más productivo para la mayoría de los científicos e ingenieros. Hay herramientas que facilitan este método.

Los lenguajes de alto nivel específicos del dominio que se utilizan en LabVIEW y el software de MathWorks MATLAB® combinan una sintaxis de programación especializada con amplias librerías de funcionalidades útiles. Por ejemplo, el software de diseño de sistemas LabVIEW, es una herramienta de programación enfocada en las aplicaciones de medida y control que interactúan con el hardware de E/S. La sintaxis de la programación gráfica de LabVIEW es muy adecuada para el manejo del paralelismo y la temporización del mundo real inherente a este tipo de aplicación. Del mismo modo, las librerías de análisis integradas y los controladores de dispositivos de hardware están diseñados para ahorrar el tiempo del ingeniero en comparación con aquellos que tienen que implementar o juntar las piezas de una solución con herramientas de propósito general y fuentes desagregadas.

Estos lenguajes específicos de dominio respaldados por el mercado, le permiten darse cuenta de las ventajas del software personalizado sin tener que pagar el precio completo del desarrollo desde cero con un lenguaje de programación de propósito general.

El enfoque limitado de estas herramientas les permite proporcionar un valor más específico para sus usuarios, centrándose en el conjunto adecuado de librerías y tomando decisiones de diseño con sus usuarios específicos en mente. Aunque estas herramientas al principio pueden potencialmente costar más dinero en comparación con los sistemas operativos de fuente abierta y gratuitos, los proveedores que proporcionan herramientas tales como Microsoft Visual Studio y Apple Xcode tienen un lado positivo, hay una empresa al otro lado del producto que depende de que usted haga un uso continuado de su producto para ganar dinero. Esto significa que tienen un interés personal en su éxito.

Los proveedores de herramientas comerciales también están más motivados en ofrecer las últimas tecnologías de tal manera que se preserve la inversión existente que ya se ha realizado al adquirir la herramienta del proveedor. Normalmente, esto se traduce en menos pérdida de clientes y más aislamiento de los cambios potencialmente perjudiciales, tales como actualizaciones forzosas de sistemas operativos (en otras palabras: la reciente jubilación de Windows XP) y los cambios de hardware subyacentes (en otras palabras: los procesadores de 64 bits están eliminando gradualmente a los de 32 bits y la aparición de los procesadores ARM).

A veces, incluso un lenguaje de programación específico de dominio requiere demasiada inversión para el desarrollo y hay más métodos disponibles en el comercio. Por ejemplo, NI ofrece software de aplicación para la secuenciación de pruebas (NI TestStand), pruebas en tiempo real (NI VeriStand) y análisis de datos en línea (NI DIAdem).

Cada una de estas tareas proporciona suficiente funcionalidad común para que sea comercialmente viable para un proveedor el ofrecer un producto que satisfaga las necesidades más comunes del mercado.

Lo que separa a estas herramientas del software de función fija es que todos ellas pueden ser ampliadas y tienen una funcionalidad personalizada conectada a ellas. Este método basado en funcionalidad conectable es mediante la que se puede añadir el gran valor de una funcionalidad personalizada específica para las necesidades (es decir, el procedimiento de prueba para su dispositivo más reciente), al mismo tiempo que permite que se pueda volver a utilizar una función bien implementada como un secuenciador de pruebas.

¿Qué hacer a continuación?

Es importante recordar que no hay una sola herramienta de software que sea la respuesta correcta a cada tarea posible. Le corresponde a usted como ingeniero o gerente de ingeniería seleccionar el método de software más productivo, ya sea comprando software ya hecho, desarrollando una solución a partir de cero en un lenguaje específico, o mezclando y combinando componentes prefabricados con lenguajes de alto nivel y software de aplicación ampliable. Ser consciente de las opciones y abierto al cambio le permitirá aprovecharse de lo mejor que haya disponible y es de esperar que gane ventaja en un mercado cada vez más competitivo. Ω