PROYECTO ATM SIMULATOR

Autores

Jerónimo Álvarez, jeronimo.alvarez@proyectos.com.uy
Alberto Canabal, aacl@adinet.com.uy
Arya Baher, aryab@adinet.com.uy

Instituto de Computación, Facultad de Ingeniería, *Universidad de la República Oriental del Uruguay,* www.fing.edu.uy/inco

Abstract

This paper is based on the career thesis of the authors, which provides the implementation of a mechanism to test financial devices, lowering significantly the costs comparing with the verification mechanisms available nowadays. It's based in the creation of a solution strongly oriented on protocols and developments promoted by the Open Source community and other open research communities.
It includes a flexible environment for developing applications for financial devices and a simulation environment for automated teller machines (ATM). These are based on the open communication protocol IFX (Interactive Financial Exchange), and the open standard XFS (Extensions for Financial Services) for the administration of hardware for financial devices, with its multiplatform variant J/XFS, which is based on the Java platform.
It also abstracts how these types of devices work as states machines, which reacts to events and execute an ordered sequence of actions. Finally, shows how this model can be used for the development of other types of devices as Point-Of-Sale (POS) terminals, soda machines, and applications in other areas like instant messaging programs and schedulers.

Keywords: Financial device, test, open communication protocol, Standard for devices administration, ATM, IFX, XFS, J/XFS, Open Source.

Resumen

El presente trabajo, basado en la tesis de grado de los autores, provee la implementación de un mecanismo de prueba de dispositivos financieros con una base de costos considerablemente inferior a la de los mecanismos actualmente disponibles de verificación, basado en la construcción de una solución fuertemente sustentada en una plataforma base de protocolos y desarrollos promovidos tanto por la comunidad Open Source como por comunidades abiertas de investigadores.

El mismo incorpora un entorno flexible de desarrollo de aplicaciones para dispositivos financieros e incluye además un entorno de simulación de ATMs (o cajeros automáticos) basados en el protocolo abierto de comunicación IFX, y el estándar abierto XFS de administración de dispositivos de hardware financieros, con su variante multiplataforma J/XFS basado en la plataforma Java.

Abstrae el funcionamiento de esta clase de dispositivos como máquinas de estado que reaccionan a eventos y realizan secuencias ordenadas de acciones; y muestra cómo es posible aplicar éste modelo al desarrollo de otros tipos de dispositivos tales como terminales POS, expendedoras de refrescos, así como en otras áreas de aplicaciones tales como programas de mensajería instantánea o planificador de tareas.

Palabras clave: Dispositivo financiero, Verificación, Protocolo abierto de comunicación, Estándar de administración de dispositivos, ATM, IFX, XFS, J/XFS, Open Source.

Introducción

El principal objetivo de la herramienta es liberar al desarrollador de la ardua tarea de probar casos de uso relacionados a un dispositivo financiero en un dispositivo real. Especialmente en el caso de cajeros automáticos, existe una diversa gama de modelos, cada uno con comportamientos específicos, por lo que en reiteradas ocasiones un desarrollador debe movilizarse físicamente a diversas instalaciones financieras para poder realizar un juego completo de pruebas funcionales que minimice los riesgos de un pasaje a producción. Esto sin contar el costo operativo que involucra para las organizaciones la dificultosa tarea de realizar, inclusive, pruebas unitarias y de integración, es decir, en etapas tempranas del desarrollo de cualquier proyecto.

Objetivos

A continuación se enumeran los objetivos planteados para la aplicación ATM Simulator:

- desarrollar un simulador de ATM con los siguientes requerimientos básicos:
  - proveer una interfaz gráfica que simule el funcionamiento real de un ATM
  - desarrollar simuladores para dispositivos de hardware de un ATM, como ser lectora de banda magnética, impresora de ticket o teclado
  - desarrollar un módulo de comunicación para conexión del simulador con un servidor central
  - el simulador de ATM debe funcionar en base a una máquina de estados configurable, modificable de forma dinámica, que actúe en base a eventos de dispositivos financieros, eventos de comunicaciones, eventos de componentes de las pantallas, o bien por eventos adicionales configurados (por ejemplo, un bloqueo temporal)

Estado del Arte

La presente sección explora ciertas características esenciales que hacen a la arquitectura de los ATM en la actualidad, y analiza a fondo las alternativas implementadas hasta el momento en cada una de ellas, que han experimentado mayor aceptación y desarrollo en el mercado financiero. Estos factores son analizados con la intención de brindar soporte a las decisiones de diseño adoptadas en el presente proyecto.

Vale la pena mencionar que a la fecha no existe una solución Open Source que cumpla con los requerimientos especificados. Las aplicaciones de ATM actualmente existentes son netamente comerciales y no se tiene acceso a su código fuente.

Las principales características a analizar son:

- interacción cliente-servidor
- presentación gráfica
- comunicación con dispositivos periféricos

Interacción Cliente-Servidor

La comunicación entre dispositivos financieros y servidores centrales es un tema ampliamente discutido en la actualidad. Se encuentra hoy en día fuertemente influenciado por protocolos propietarios, que condicionan inclusive las prestaciones mismas de los dispositivos financieros.

A continuación se exploran las dos corrientes de desarrollo mas conocidas al momento:

- ATM tradicional
- ATM inteligente

ATM Tradicional

Un ATM tradicional es una Terminal tonta manipulada en base a *órdenes recibidas desde un servidor*, *la cual no es consciente de los procesos del negocio* ni de la semántica de la información de negocio que manipula. Actúa sobre sus dispositivos en base a respuestas recibidas del servidor central, quien le indica a que estado ir a continuación y que acciones tomar (imprimir ticket, dispensar dinero, retener tarjeta, etc.), utilizando para dicha comunicación protocolos propietarios.[1] Dado que en la actualidad no existen especificaciones abiertas de mensajería para esta clase de arquitecturas, se requiere desarrollar un nuevo protocolo con características similares a los ya existentes.

ATM Inteligente

Un ATM Inteligente es una aplicación cliente que *es consciente de los procesos del negocio y de la semántica de la información de negocio que manipula*. Conoce la secuencia de acciones a realizar sobre sus dispositivos periféricos en base a la información de negocio recibida del servidor, el cual, sólo tiene potestad para monitorear el estado del ATM y administrar los Modos del mismo (En línea, Fuera de línea, etc.).

El protocolo de mensajería más difundido en la actualidad para esta clase de ATM se basa en la especificación abierta IFX. Este estándar es una especificación abierta de mensajería de negocios, una plataforma diseñada para soportar interoperabilidad a gran escala, independiente de la arquitectura de los dispositivos financieros así como de la capa de transporte de datos. Facilita el intercambio de datos entre múltiples partes, garantizando la consistencia de la información, indicando en todo momento a quién pertenece la misma y qué es lo que representa.

Presentación gráfica

A continuación se analizan dos alternativas fuertemente difundidas en el mercado financiero que determinan la capacidad de presentación gráfica de los dispositivos financieros:

- Especificaciones propietarias
- Aplicaciones Web-Enabled

Especificación propietaria

Las implementaciones gráficas de los ATM tradicionales son propietarias y con un diseño particular de cada proveedor. Desarrollar una implementación propietaria similar a los ya existentes, puede permitir traducir de forma relativamente sencilla la configuración de las pantallas a especificaciones de otras marcas de modo de reutilizar las mismas pantallas en toda la red de ATM de una empresa. De todos modos esta traducción siempre estará limitada por la compatibilidad entre las diferentes especificaciones.

[1] Los protocolos mas extendidos son NDC+ o Diebold, sobre los que no se tiene libre acceso. Ambos protocolos son propietarios de las empresas de cajeros automáticos más grandes a nivel mundial, NCR y Diebold.

Aplicación Web-enabled

A diferencia de la rigidez impuesta por las clásicas pantallas de los ATM tradicionales, las nuevas generaciones de ATM plantean un cambio de paradigma en lo que refiere a la presentación visual de estos dispositivos.

Un Web-enabled ATM es un dispositivo que opera sobre tecnologías Web, facilitando la presentación de contenido dinámico. Permite, de este modo, mediante un mecanismo simple, abierto y ampliamente difundido, el despliegue visual de contenido multimedia adaptado a las necesidades de cada usuario [1].

Comunicación con dispositivos periféricos

Un tema de suma relevancia para el desarrollo de cualquier aplicación sobre dispositivos financieros, es la interacción de ésta con dispositivos periféricos particulares. Cada nuevo dispositivo de hardware presenta sus particularidades y la aplicación financiera debe ser capaz de lidiar con cada una de ellas. El problema crece cuando la aplicación debe contemplar una diversa gama de dispositivos, modelos, marcas, etc.

A continuación se analizan las tres corrientes mas conocidas en el mercado actualmente. La primera plantea la complejidad y la problemática actual al momento de desarrollar aplicaciones financieras, y las siguientes son alternativas que surgen de la evolución del pensamiento y la experiencia de las organizaciones:

- Librerías propietarias
- Librerías XFS
- Librerías J/XFS

Librerías propietarias

La mayoría de los dispositivos de hardware cuentan con librerías propietarias que permiten su interacción con aplicaciones de alto nivel. Tal como se mencionó anteriormente, una aplicación ATM debe tener la capacidad de interactuar con la diversidad de marcas y modelos de dispositivos con los que cuente la red de ATM. La complejidad crece entonces a medida que aparecen nuevos dispositivos financieros, cada uno con sus propias particularidades.

Librerías XFS

Si bien el desarrollo de cada ATM es actualmente propietario, las nuevas generaciones de ATM buscan independizarse del hardware, y para ello basan sus librerías en especificaciones abiertas, tales como la especificación XFS de comunicación con dispositivos periféricos. XFS especifica una colección de interfaces para la conexión de aplicaciones financieras con dispositivos periféricos. Abstrae la complejidad que involucra el soporte de interfaces de hardware específicas de cada fabricante, estableciendo tres componentes clave en su arquitectura, librerías de base, interfaces para las aplicaciones, e interfaces de servicios para los proveedores de hardware.

La existencia de un XFS Manager (ver figura) [3], permite la integración de los distintos actores del sistema, habilitando la interacción transparente de una aplicación con dispositivos periféricos instalados eventualmente en un nodo remoto de una red.

Partiendo de la base de que los fabricantes de hardware proveen una librería XFS para la manipulación de sus dispositivos, una aplicación ATM que implemente dicha especificación se verá liberada de la dependencia al hardware de base.

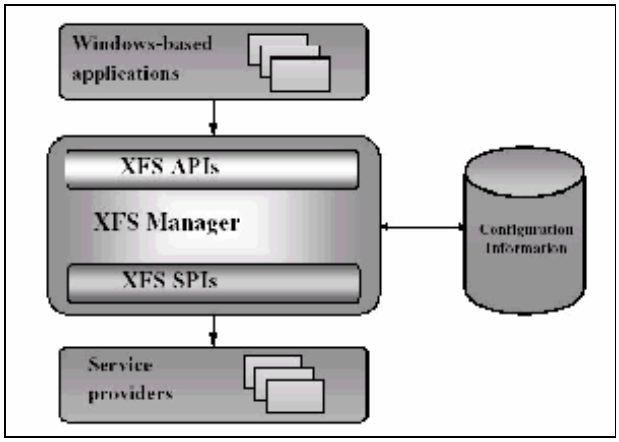

Ilustración 1 - Arquitectura XFS

En caso de no contar con hardware específico, o bien que no se provean librerías XFS, es posible cumplir con los requisitos de un simulador ATM basado en XFS utilizando stubs[2] desarrollados específicamente para simular los dispositivos, así como excepciones generadas por los mismos.

Librerías J/XFS

Fuertemente basada en el Estándar XFS, los desarrollos orientados a la especificación J/XFS pretenden eliminar la dependencia tecnológica que presenta XFS de los sistemas operativos Microsoft Windows®. Siguen una línea de desarrollo basada en la plataforma Java, la cual permite implementar soluciones para toda la variedad de sistemas operativos que soportan esta plataforma [4].

Decisiones generales de diseño

A continuación se presentan las decisiones más importantes de diseño realizadas que inciden de forma significativa en la arquitectura definida.

Interacción Cliente-Servidor

A partir de las dos alternativas estudiadas (ATM Tradicional y ATM Inteligente), surge la necesidad de crear una nueva opción que conjugue los beneficios de ambas tendencias, al que se denominará de ahora en adelante ATM Híbrido.

Esta opción propone una solución intermedia a ambas alternativas: conocer los procesos de negocio y la semántica de la información de negocio que manipula, pero no tener control total sobre la secuencia de acciones a realizar sobre los dispositivos periféricos. La responsabilidad de establecer y transmitir dichas secuencias de acciones pasa a ser principalmente del servidor.

La especificación abierta IFX no contempla todas las funciones requeridas para implementar el modelo propuesto, por lo que resulta necesario especificar una semántica adicional para la representación de las secuencias de acciones sobre dispositivos periféricos.

La decisión tomada para remediar esta situación se basa en extender la gramática de la especificación IFX, estableciendo campos propietarios necesarios para representar los nuevos elementos requeridos en la mensajería.

[2] El término stub corresponde a una pieza de código utilizada en reemplazo de una pieza más compleja o hasta el momento no desarrollada. Un stub se utiliza especialmente en las etapas de desarrollo y verificación de software, cuando la concentración de trabajo se encuentra focalizada sobre otra pieza de código que presenta una dependencia con la reemplazada por el stub.

Presentación gráfica

Ante las dos posibles alternativas planteadas, referentes a la capacidad gráfica de dispositivos financieros (especificación propietaria y aplicación Web-enabled), la implementación se apoya especialmente en la potencia de la opción Web-enabled para la presentación de elementos multimedia (imágenes, animaciones, textos estáticos), y opta por seguir una especificación propietaria flexible para la presentación de elementos gráficos que interactúen con el simulador (botones, cuadros de texto, etiquetas).

Comunicación con dispositivos periféricos

En base a las alternativas estudiadas para la comunicación con dispositivos periféricos, se opta por utilizar librerías J/XFS, por su grado de flexibilidad, interoperabilidad y facilidad de integración al simulador, dado que ha sido desarrollado en el mismo lenguaje de programación (Java).

Configurabilidad - Interoperabilidad XML

La parametrización del sistema se ha estructurado en base a archivos de configuración en formato XML.

Las razones que han llevado a elegir esta opción son:

- Es un estándar ampliamente utilizado, para el que existen un número considerable de librerías y utilitarios, lo que permite la utilización de estas configuraciones desde distintas aplicaciones, no necesariamente desarrolladas en Java.
- Pueden editarse manualmente. Incluso pueden utilizarse diversos programas comerciales para su edición, lo cual resulta muy conveniente para representar estructuras complejas de datos.

Esencialmente los componentes del sistema son persistidos mediante una librería de codificación/decodificación de Java Beans a XML y viceversa. Por tal motivo se requiere que los elementos a persistir respeten las características de un Java Bean o bien provean componentes auxiliares que realicen los mapeos necesarios para cumplir con esta especificación.

Arquitectura de la solución

En las siguientes secciones se presenta una visión general de la arquitectura del sistema, así como una pequeña introducción a cada uno de los componentes que la conforman.

Esquema de la aplicación

La arquitectura del sistema presenta un diseño orientado a plug-ins. En la misma es posible categorizar dos clases de componentes:

- Kernel: componente central de la arquitectura. También conocido como núcleo, es la parte fundamental del sistema. Es el software responsable de gestionar los recursos y brindar los servicios básicos al resto de los componentes.
- Plug-in: un plug-in (o plugin) es un programa que interactúa con el kernel aportando a éste funcionalidades o utilidades específicas.

La figura 1 ilustra los principales componentes de la arquitectura desarrollada.

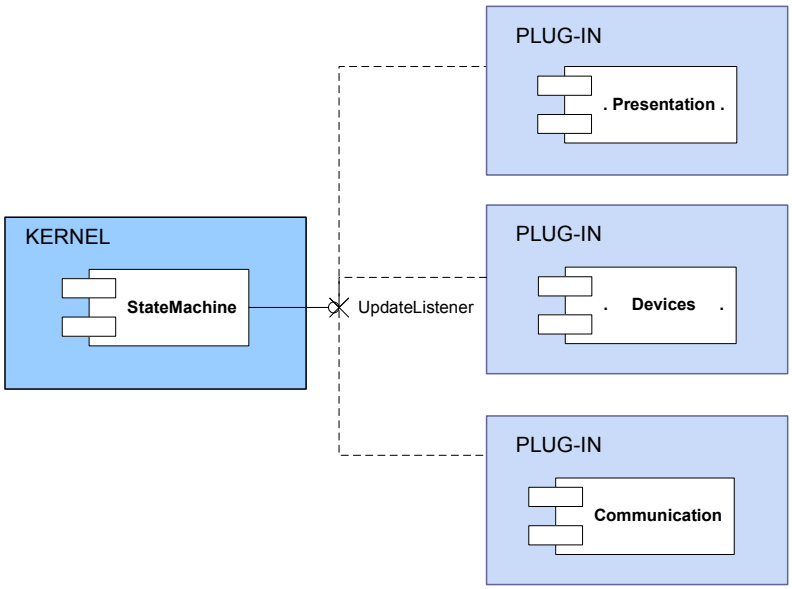

Figura 1 - Arquitectura de plug-ins

El diseño de la aplicación se encuentra dividido en cuatro componentes principales: subsistema de máquina de estados, subsistema de presentación, subsistema de comunicaciones y subsistema de dispositivos. De estos cuatro componentes, el de máquina de estados funciona como kernel de la aplicación, mientras que los restantes han sido desarrollados como plug-ins del mismo.

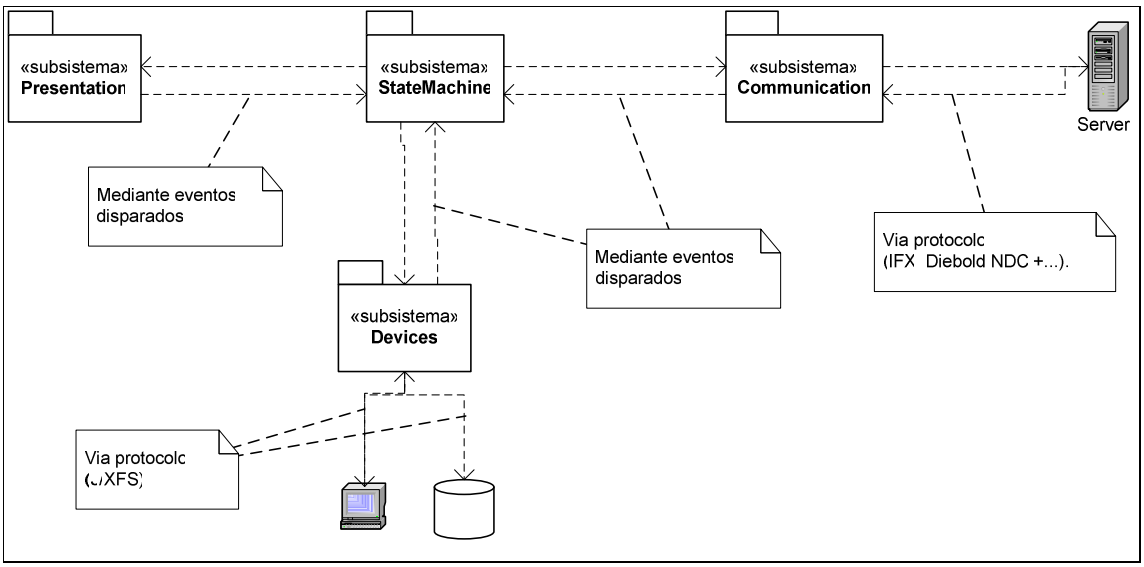

Figura 2 - Subsistemas

El subsistema de máquina de estados responde a eventos disparados por los otros subsistemas, y se comunica con éstos mediante acciones provistas para tal fin, como ilustra la figura 2. Cada uno de los demás subsistemas resuelve diferentes aspectos de la aplicación:

- El subsistema de presentación es el encargado de la presentación de los datos.
- El subsistema de comunicaciones provee los mecanismos necesarios para la comunicación del sistema con otras aplicaciones, mediante la implementación de protocolos de comunicaciones establecidos (por ejemplo, IFX, Diebold, NDC+).
- El subsistema de dispositivos gestiona la comunicación con los dispositivos conectados vía J/XFS.

Subsistema de máquina de estados

El subsistema de máquina de estados, componente central de la arquitectura, es el responsable de la creación y operación de las máquinas de estados, las cuales definen la base del comportamiento de cada instancia del sistema.

Subsistema de presentación

El subsistema de presentación permite la creación y despliegue de pantallas, las cuales tienen como fondo una página HTML y componentes de entrada y salida.

Subsistema de dispositivos

El subsistema de dispositivos provee las funcionalidades necesarias para interactuar con cualquier tipo de dispositivo financiero, en particular los dispositivos periféricos de un ATM.

Su diseño, si bien está influenciado por la arquitectura de base provista por la librería J/XFS, se ha desarrollado independientemente de toda implementación particular, de modo de poder incorporar en un futuro, de forma flexible y dinámica, otras implementaciones de librerías de dispositivos periféricos alternativas a J/XFS.

Subsistema de comunicaciones

El subsistema de comunicaciones brinda una capa de abstracción sobre la forma de comunicación de la máquina de estados con otros sistemas (por ejemplo, servidores u otras máquinas de estados), utilizando el protocolo de comunicaciones correspondiente.

ATM Simulator en funcionamiento

A continuación se presenta la simulación de una aplicación ATM diseñada utilizando las herramientas desarrolladas en el contexto del proyecto. La misma incluye una serie de capturas de pantalla que muestran la realización de una operación de retiro en efectivo. En dichas capturas pueden observarse las pantallas diseñadas utilizando el editor de pantallas, así como los simuladores provistos para cada uno de los dispositivos de hardware configurados (en este caso, un teclado, una lectora de tarjeta y una impresora).

Al ejecutar el simulador, se despliega una pantalla de solicitud de datos de la tarjeta del usuario (Figura 3).

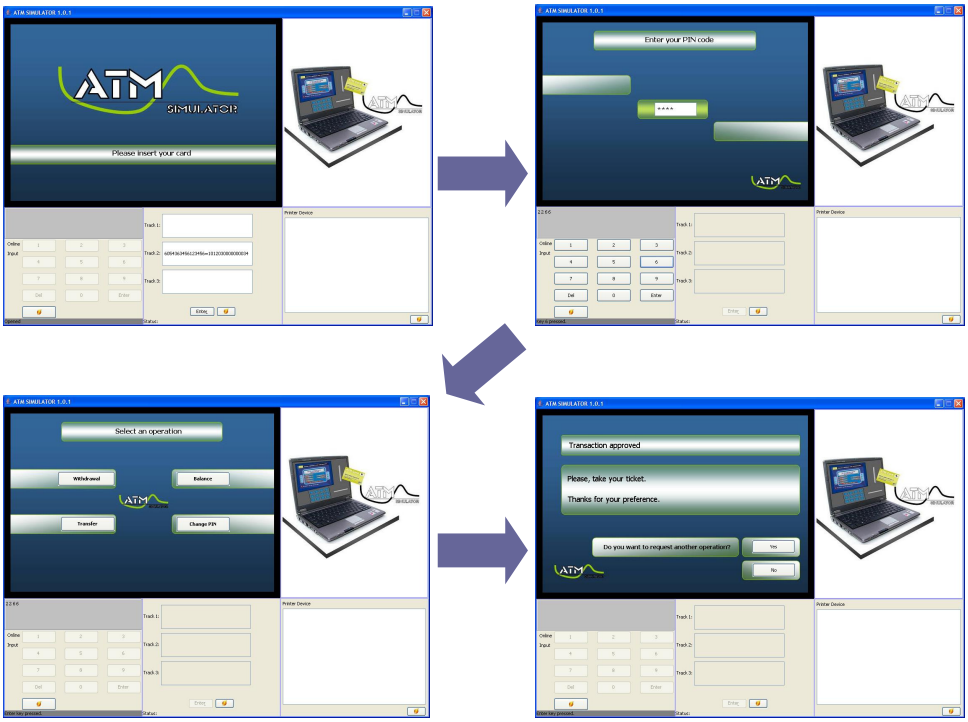

Figura 3 – Ejemplo de simulación de ATM

Luego de ingresar y confirmar los datos solicitados en la lectora de banda magnética, el simulador despliega una pantalla de solicitud de PIN (clave de identificación personal). A continuación se visualiza una pantalla de selección del tipo de operación a realizar, en este caso, el usuario selecciona la opción Withdrawal (retiro), pasando entonces a la siguiente pantalla La secuencia continúa a través de una serie de pantallas de solicitud de datos de la transacción y finalmente envía la misma a una aplicación externa, espera una respuesta, y notifica en una pantalla final el resultado obtenido.

Desarrollo de la aplicación

En esta sección se presenta una breve introducción a las aplicaciones desarrolladas que componen la simulación presentada anteriormente, sus principales características y algunas métricas interesantes del esfuerzo invertido en su desarrollo.

La simulación se compone básicamente de dos aplicaciones centrales que interactúan entre sí:

- una instancia de ATM
- una instancia de Servidor Central

El ATM opera como una máquina de 15 estados, con 121 acciones y 38 transiciones de estado configuradas. Además, soporta la ejecución de acciones enviadas desde el Servidor Central, funcionando así como ATM híbrido. Sú modulo de comunicación utiliza 2 canales de comunicación IFX para la interacción con el Servidor Central, y su módulo de dispositivos soporta la comunicación con 3 tipos de dispositivos J/XFS (lectora de banda magnética, teclado, impresora de tickets). La configuración e instanciación de la arquitectura es, en este caso, completa: es necesario configurar el kernel para que opere la máquina de estados, el plug-in de presentación para que despliegue las diversas pantallas, el plug-in de comunicación para que levante y administre los canales IFX, y el plug-in de dispositivos para permitir la comunicación con los periféricos.

Por otro lado, se ha desarrollado una aplicación de servidor muy simple, denominada Servidor Central, el cual atiende los pedidos que llegan desde un ATM y solicita al usuario mediante pantallas informativas la aprobación o rechazo de las transacciones, permitiendo modificar de este modo la respuesta enviada al ATM en cada caso. Además, presenta funcionalidades para administrar los modos de operación del ATM (en servicio, fuera de servicio, etc.) o apagarlo en caso que sea necesario. Si se analiza un poco más a fondo la especificación del Servidor Central, es una aplicación que debe responder a eventos generados desde canales de comunicación IFX, presentar pantallas informativas a los usuarios, y permitir a éstos solicitar la ejecución de determinadas acciones sobre el ATM. Esta aplicación, entonces, puede modelarse perfectamente como una máquina de estados, y es precisamente lo que se ha hecho: el Servidor Central opera como una máquina de 4 estados, con 60 acciones configuradas y utiliza 2 canales de comunicación IFX para el envío de órdenes y mensajes de respuesta al cliente. Dado que el Servidor Central no posee dispositivos J/XFS, no ha sido necesario configurar el plug-in de dispositivos para la instanciación del sistema. El kernel y los demás plug-in se configuran de manera similar a los correspondientes del lado del ATM.

Con respecto al esfuerzo de desarrollo de la simulación, la configuración de la aplicación ATM ha consumido alrededor de 12 horas de trabajo, y la del Servidor Central alrededor de 4. En resumen, en menos de dos días de trabajo puede contarse con un ambiente de pruebas básico para ATMs, el cual brinda, además, una serie de elementos reusables para futuras configuraciones. Por ejemplo, si se agregan nuevos dispositivos J/XFS, simplemente debe configurarse el plug-in de dispositivos para que los soporte, configurar en el kernel las acciones correspondientes, y ya se puede contar con un nuevo ambiente listo para realizar pruebas. Comparando esta alternativa con el tiempo y esfuerzo que insume actualmente la realización de pruebas en cada uno de los distintos modelos reales de ATM, resulta notorio el ahorro de esfuerzo y recursos obtenido. Inclusive, si se cuenta con ATMs que operen sobre este sistema, los cambios de configuración y agregado de funcionalidades son extremadamente simples: únicamente deben instalarse en el ATM los archivos XML de configuración necesarios, lo cual puede realizarse fácilmente de forma remota, y el ATM ya se encontrará listo para su nueva forma de operación.

Logros Alcanzados

El sistema ATM Simulator construido en el proyecto cubre los objetivos planteados.

- Permite el diseño de aplicaciones de ATM, brindando entornos gráficos para el diseño de las máquinas de estado y pantallas que facilitan su desarrollo. Tales aplicaciones pueden ser configuradas para interactuar con dispositivos periféricos, establecer canales de comunicación, y presentar al usuario una interfaz amigable basada en tecnología Web.
- Permite además visualizar su comportamiento en un entorno gráfico de simulación desarrollado especialmente para tal fin. Esta aplicación brinda, además, la posibilidad de simular fallas sobre los dispositivos periféricos a solicitud del usuario.

Cualidades del producto

Alta configurabilidad

El grado de configurabilidad del sistema ha sido uno de los elementos prioritarios considerados en la etapa de diseño. Todos los componentes desarrollados son configurables a partir de

archivos XML, lo cual posibilita adaptar la aplicación a distintas realidades sin necesidad de recompilar su código.

Versatilidad

El proyecto brinda como resultado un paquete de librerías y utilitarios que ofrecen al usuario la posibilidad de diseñar visualmente, configurar, y ejecutar una variada gama de aplicaciones, cuyo modelo base orientado a eventos cuente con poco procesamiento de datos.

La herramienta permite diseñar una amplia variedad de aplicaciones, dentro de las que se destacan:

- Aplicaciones visuales que no requieran procesamiento de datos. Un ejemplo de esto es una aplicación de información a clientes en un centro comercial.
- Aplicaciones que interactúan con otras por medio de una red de comunicaciones, particularmente una red TCP/IP. Es posible implementar un servicio de comunicaciones que permita, por ejemplo, compartir información financiera entre entidades bancarias o realizar transacciones comerciales. Inclusive habilita el desarrollo de soluciones no orientadas al ámbito financiero, como ser una aplicación básica de mensajería instantánea (Chat).
- Aplicaciones que se comunican con dispositivos periféricos, por ejemplo una máquina expendedora de refrescos.
- Aplicaciones más complejas que requieran comunicación con servidores, interacción con dispositivos y usuario mediante una interfaz gráfica, por ejemplo una aplicación de ATM.

Extensibilidad

El alto grado de modularidad del sistema, permite al desarrollador incorporar de forma sencilla, nuevas funcionalidades que potencien las capacidades del mismo. Dichos componentes pueden ser adicionados, en forma de plug-in, sin necesidad de realizar modificaciones a los ya existentes.

Conclusiones

El resultado permitió cumplir efectivamente con los objetivos planteados. De hecho, la solución desarrollada, habiendo sido diseñada en base a patrones que aseguran su flexibilidad y adaptabilidad, permitió su reutilización para la resolución de requerimientos que en principio estaban definidos por fuera del alcance del proyecto. La aplicación servidor, mencionada anteriormente, es un buen ejemplo de este punto.

Esto demuestra el alto grado de configurabilidad del sistema construido, el cual constituye uno de los pilares centrales de su arquitectura. El esfuerzo se ha enfocado en la creación de una herramienta fuertemente parametrizable, que brinde facilidades para su personalización y extensibilidad.

Trabajo a futuro

Si bien el sistema posibilita el desarrollo de una gran variedad de aplicaciones, el mismo no ofrece aún funcionalidades esenciales para el desarrollo de un amplio espectro de soluciones. Esta limitación puede ser solucionada mediante la implementación de una serie de nuevos plug-ins que incorporen, por ejemplo, persistencia en base de datos o llamadas a procesos externos en las aplicaciones.

De la misma manera, pueden implementarse alternativas a los protocolos de interacción con los dispositivos periféricos y de comunicación, para permitir el desarrollo de otro tipo de aplicaciones.

Referencias

[1] "This is not your father's ATM". ACI Trends Paper http://www.aciworldwide.com/pdfs/aci_trends_atm.pdf.

[2] "CWA 14050 - Extensions for Financial Services (XFS) interface specification – Release 3.03".http://www.cenorm.be/cenorm/businessdomains/businessdomains/isss/cwa/cwa14050.asp

[3] "WOSA/XFS (CEN-XFS)". S&N Website. http://www.s-und-n.de/index.php?option=com_content&task=view&id=91&Itemid=136&lang=en

[4] J/XFS Website. http://www.jxfs.net/

Bibliografía adicional

"What's bugging ATM vendors?". Banker Middle East. http://www.bankerme.com/bme/2004/jan/e_banking.asp

"J/XFS Workshop - CWA 14923: 2004". http://www.cenorm.be/cenorm/businessdomains/businessdomains/isss/cwa/cwa14923_04.asp

"Bank on it: introduction to J/XFS". IBM Corp. http://www-128.ibm.com/developerworks/java/library/j-jxfs1/

"IFX Forum". http://ifxforum.org.

"IFX-Framework Project". http://ifx-framework.sourceforge.net/.

"Web-Enabled ATMs: What Could Possibly go Wrong?". Bill Gerba. Wirespring. http://www.wirespring.com/dynamic_digital_signage_and_interactive_kiosks_journal/articles/Web_Enabled_ATMs__What_Could_Possibly_go_Wrong_-217.html