(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 544 465**

(51) Int. CI.:

G06F 11/34 (2006.01)
G06F 21/00 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **23.12.2011 E 11195582 (9)**

(97) Fecha y número de publicación de la concesión europea: **20.05.2015 EP 2608041**

(54) Título: **Supervisión de la actividad del usuario en dispositivos móviles inteligentes**

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
31.08.2015

(73) Titular/es:

DEUTSCHE TELEKOM AG (100.0%)
Friedrich-Ebert-Allee 140
53113 Bonn, DE

(72) Inventor/es:

GLASS, GREGOR;
HENKE, KATJA;
SCHNEIDER, LUTZ;
BATYUK, LEONID;
SCHMIDT, AUBREY-DERRICK;
RADDATZ, KARSTEN;
ALBAYRAK, SAHIN y
CAMTEPE, AHMET

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

DESCRIPCION

Supervisión de la actividad del usuario en dispositivos móviles inteligentes

Campo de la invención

La invención se refiere a un método para supervisar la actividad del usuario en dispositivos móviles y a un método para analizar la actividad del usuario en un servidor. Además, se proporciona un producto de programa de ordenador que comprende uno o más medios interpretables por ordenador que disponen de instrucciones ejecutables por el ordenador para realizar las etapas de al menos uno de los métodos mencionados anteriormente.

Antecedentes de la invención

Los dispositivos móviles computarizados, tales como, por ejemplo, teléfonos o tabletas, están proporcionando la posibilidad de navegar fácilmente por Internet, manejar correos electrónicos y hacer llamadas telefónicas en cualquier lugar. Debido a su creciente atractivo, cada vez más gente está comenzando a usar tales dispositivos creando un gran mercado para los proveedores de redes y servicios, desarrolladores de aplicaciones y especialistas comerciales. Dado que estos dispositivos proporcionan funcionalidades de fácil comprensión a sus usuarios, la complejidad de ciertas aplicaciones y servicios es demasiado alta para que los maneje un usuario medio. Ayudar al usuario a manejar esta complejidad es una dura tarea ya que los usuarios no pueden a menudo describir la situación técnica exacta que les está causando problemas.

Especialmente, los teléfonos móviles han llegado a ser hoy dispositivos de comunicación y computación central. Desde Agosto de 2006, se han registrado en Alemania más teléfonos móviles que habitantes. A medida que crecen las capacidades de estos dispositivos, ya no son de ninguna manera simples teléfonos centrados en la voz. Representan un escalón hacia delante materializando la visión de la computación ubicua de Mark Weiser (Mark Weiser, The computer for the 21st century, Scientific American, 265(3):94-104, Septiembre 1991). En esta visión, se describe que los ordenadores clásicos se reemplazarán por dispositivos pequeños, inteligentes, distribuidos y trabajando en red que se integrarán en los objetos y actividades diarias. Este reemplazo se puede observar en tiendas y almacenes que utilizan etiquetas para supervisar y controlar los artículos. La evolución de los dispositivos móviles, en particular los teléfonos inteligentes, se puede ver como parte de esta visión dado que representa una posibilidad para hacer uso de las capacidades técnicas y computacionales en un contexto de móviles. El teléfono inteligente es un término utilizado normalmente para describir los teléfonos móviles actuales en general, aunque no existe una definición industrial global. Un modo común de comprender este término es que estos dispositivos proporcionan características del estado de la técnica así como entornos de desarrollo de software que permiten la creación de aplicaciones de terceros.

Con las crecientes capacidades de los dispositivos móviles, cada vez más gente comienza usar dispositivos móviles tales como teléfonos inteligentes o tabletas. Las capacidades para ayudar al usuario a manejar a distancia la complejidad de algunas aplicaciones en tales dispositivos móviles están generalmente limitadas debido a la ausencia de soluciones que estén soportadas por los sistemas operativos. Debido a razones de seguridad, los interfaces normalizados de programación de aplicaciones (API) no proporcionan un código satisfactorio para la funcionalidad a distancia, aunque las capacidades a distancia mejorarían ampliamente la efectividad de los servicios de soporte.

Uno de los más reciente sistemas operativos con móviles es Android de Google, que es un paquete de software que incluye un sistema operativo, un programa intermedio y aplicaciones básicas. El primer dispositivo Android vio la luz en octubre de 2008, el T-Mobile G1. El sistema Android se basa en Kernel 2.6 de Linux y soporta la mayoría de sus funcionalidades. Android trata cada aplicación de igual modo, significando tanto, que un desarrollador sea capaz de reemplazar todos los programas Android únicos como que una aplicación Android pueda funcionar en cualquier dispositivo Android estando solo limitado por las funcionalidades proporcionadas. Los mecanismos de seguridad Android de Google están basados en los del sistema Linux. El control de accesos, por ejemplo, IDs de usuario y de grupo, se gestiona de modo que cada aplicación instalada, toma su propia ID de usuario con sus permisos específicos. Estos permisos permiten un ajuste preciso para acceder a los procesos utilizando ciertas funcionalidades, por ejemplo, enviar mensajes SMS o hacer una llamada telefónica. Debido a su concepto de plataforma abierta y a lo compartido que está, Android es uno de los focos de la mayoría de los desarrolladores de software malintencionado.

Los atributos internos de un sistema de ordenador están sujetos a considerables fluctuaciones. Con cada cambio de estado cambia asimismo el estado del sistema. Los planteamientos basados en Web conocidos de Google Analytics, Opentracker o Clicktale pueden utilizar pulsaciones del usuario, movimientos del ratón e información de datos del usuario para rastrear el entorno del usuario. En comparación con esto, supervisar el entorno del usuario en un dispositivo móvil, tal como un teléfono inteligente o una tableta o semejante, es mucho más complicado dado que la mayoría de los sistemas operativos móviles restringen el acceso a la correspondiente funcionalidad por razones de seguridad.

En una evaluación de Falaki et al. del uso del teléfono inteligente, se supervisaron dispositivos para obtener información acerca del entorno del usuario (Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios

Lymberopoulos, Ramesh Govindan y Deborah Estrin; Diversity in smartphone usage; In Proceedings of the 8th international conference on Mobile systems, applications and services, MobiSys '10, páginas 179-194, Nueva York, NY, USA, 2010. ACM). Dos diferentes conjuntos de datos se utilizaron para analizar el entorno del usuario. El primero se adquirió utilizando teléfonos inteligentes basados en Android mientras que el segundo conjunto de datos (Windows Mobile) lo proporcionó una organización que investiga la costumbre en los usos del teléfono inteligente. El conjunto de datos basados en Android en esta evaluación se obtuvo de un depósito de registro de datos de clientes, SystemSens (H. Falaki, R. Mahajan y D. Estrin, Systemsens: A tool for monitoring usage in Smartphone research deployments; In Proceedings of the sixth ACM international workshop on mobility in the evolving internet architecture. ACM, 2011). Esto se realiza como un servicio de antecedentes que registra el estado de la pantalla (activada o desactivada), el inicio y la finalización de llamadas de voz entrantes y salientes, la duración de la interacción por aplicación, el tráfico de red que cada aplicación produce y finalmente el nivel de la batería. Los datos reunidos se almacenan en una base de datos SQLite en el dispositivo y se descargan en el servidor cuando se carga el aparato, con objeto de no alterar los datos de la batería. SystemSens recoge la mayoría de sus datos basados en eventos por medio de una API de registro de eventos introducida en Android 2.2 para minimizar el consumo de energía. Algunos datos tales como la utilización de la CPU y de la memoria se recogen por medio de consultas a intervalos fijos de 2 minutos.

Swatch es un supervisor de archivos registrados con capacidades de filtrado (Stephen E. Hansen, E. Todd Atkins y E. Todd; Automated system monitoring and notification with swatch; páginas 145-155, 1993). Su propósito es facilitar la supervisión de la salud del sistema y el estado de seguridad de los sistemas de ordenador. Se basa en la función syslog de UNIX. Swatch, que significa Simple WATCHer, supervisada los ficheros registrados y los filtra para obtener información importante del sistema. Permite especificar acciones tales como la ejecución de un escrito o enviar correos activados por ciertos patrones en el registro. La funcionalidad de Swatch se basa en expresiones corrientes que son la razón por la que se realiza en Perl. El syslog subyacente facilita la escala ascendente hacia diversos anfitriones para el acceso a un servidor de accesos exclusivo. Con objeto de obtener la información requerida, las diversas utilidades del sistema requieren modificación para informar más en conjunto al syslog. La determinación del estado de un sistema distribuido es más difícil que en sistemas normales de ordenador. Se han desarrollado diversas soluciones para detectar el estado en sistemas distribuidos pero no se extrae detallada información del usuario en donde las ideas clave básicas tienen la misma base para detectar actividades en un sistema.

K. Mani Chandy y Leslie Lamport presentan un algoritmo para determinar el estado global de un sistema distribuido durante un cálculo de ordenador (K. Mani Chandy and Leslie Lamport; Distributed snapshots: determining global states of distributed systems; ACM Trans. Comput. Syst., 3:63-75, Febrero 1985). Se emplea un procedimiento que adquiere información de otros procedimientos. Estos procedimientos pueden registrar su propio estado. Esto quiere decir que el procedimiento que adquiere la información tiene que trabajar en cooperación con esos otros procedimientos. El procedimiento de adquisición se superpone al cálculo de ordenador subyacente y tiene que operar en conjunción pero no altera el cálculo de ordenador subyacente. El algoritmo se desarrolló para detectar propiedades estables tales como "sistema bloqueado".

Otro planteamiento para supervisar el estado de un sistema distribuido ha sido desarrollado por Van Renesse et al. (Robbert Van Renesse, Kenneth P. Birman y Werner Vogels; Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining; ACM Trans. Comput. Syst., 21:164-206, Mayo 2003). Introducen una tecnología para supervisar un sistema distribuido llamado Astrolabe. Astrolabe es como un sistema DMS de gestión de información distribuida. Supervisa el estado dinámicamente cambiante del sistema y genera resúmenes de esta información al usuario. Mostraron que Astrolabe puede graduar miles de nodos sin degradación significativa de la eficacia. El retardo de propagación de la información entre los lodos está en un intervalo de decenas de segundos. Meng S. et al. presentaron su planteamiento para supervisar el estado de un centro de datos en nube. Su planteamiento está basado en ventanas en las que desarrollaron una estructura llamada supervisión WIndow-based StatE (WISE). Esta estructura avisa sólo cuando existe una violación continua del estado dentro de una ventana en el tiempo. Durante el desarrollo de WISE encontraron diversos desafíos. En primer lugar, encontraron agregación distribuida que quiere decir la habilidad de resumir la información de magnitudes supervisadas distribuidas voluminosas. En segundo lugar, mientras se agrega información en un sistema distribuido puede dar lugar a operar tareas similares de manera aislada lo cual es un consumo de recursos innecesario, que significa que la estructura tiene que compartir las tareas de agregación. Se debe mostrar que WISE resolvió estos desafíos y que reduce la comunicación entre 50%-90% comparado con la supervisión instantánea.

Xev es una herramienta que trabaja en sistemas basados en Linux para mostrar eventos basados en ventanas. Esto se hace creando una ventana propia o uniéndose a una ventana existente. Cada evento, por ejemplo, un arrastre de ratón, la pulsación de una tecla o un movimiento de ventana se indica y se puede almacenar para un tratamiento posterior. El principal propósito de esta herramienta reside en que indica que tiene que ser depurado. El problema principal de trabajar con Xev es que necesita un entorno de Servidor X para poderse ejecutar. Por ahora, este entorno no está disponible para dispositivos móviles en el mercado masivo actual que utilizan sistemas operativos con móviles tales como Android.

En un primer vistazo, un lugar prometedor para estudiar la información de la actividad relativa al usuario es el propio sistema operativo, por ejemplo, en el caso del sistema operativo Android mirar en la Máquina Virtual Dalvik que ejecuta los ejecutables Android/Dalvik. Sin embargo, en la mayoría de los sistemas operativos, el código fuente de

las partes importantes para la información de la actividad relativa al usuario no está bien documentado y ejecutar estas partes dentro de un método para supervisar la actividad del usuario daría como resultado una alta dependencia de los cambios realizados en el código fuente.

En el caso de Android, se proporciona una clase llamada Instrumentation que ayuda a supervisar las aplicaciones ejecutadas. Cuando se trabaja con Instrumentación activada, esta clase se iniciará antes de cualquier código de aplicación, permitiendo supervisar toda la interacción que el sistema tiene operando con la aplicación. Una realización de Instrumentation se describe en el sistema por medio de una etiqueta AndroidManifest.xml's ¡instrumentation¿. Por lo tanto, la clase Instrumentation es una herramienta que proporciona acceso simple a la información relativa a las Android Activities que son en la mayoría de los casos las pantallas relacionadas con las aplicaciones. Sin embargo, la clase tiene que ser registrada en el AndroidManifest.xml de cada aplicación supervisada ya que la aplicación será iniciada por la clase Instrumentation para que se pueda supervisar. El resultado es que cada aplicación instalada necesitaría ser modificada para un planteamiento de supervisión genérico haciéndola inoperativa para un método de supervisar la actividad del usuario en dispositivos móviles.

Una etapa esencial en la supervisión de la actividad del usuario en la identificación de la aplicación prioritaria operativa. El caso de Android, la aplicación prioritaria representa la aplicación/Android Activity vista en realidad. Todas las Activities abiertas se almacenan en la Activity Stack en la que la aplicación prioritaria se encuentra en el tope de esta pila. Android proporciona funcionalidad para investigar la pila y por ello, permitir identificar la aplicación prioritaria.

En los dispositivos Android, utilizando la clase KeyListener, sería posible registrar todas las entradas relativas a una cierta Android View. Views ocupa áreas rectangulares en la pantalla en la que cada vista es responsable para dibujar y manejar eventos de una cierta sub parte de una Activity. Parte de estas entradas son las teclas pulsadas que se pueden registrar para una aplicación correspondiente. El factor limitativo que lo hace inapropiado para un método de supervisar la actividad del usuario en dispositivos móviles es que cada aplicación supervisada necesitaría modificaciones para registrar las teclas similarmente al uso de la clase Instrumentation.

Para supervisar la actividad del usuario, no es suficiente identificar solamente la aplicación prioritaria. Puede ser muy beneficioso analizar los elementos de interfaz del usuario de una aplicación en funcionamiento en ese punto ya que proporcionaría información más detallada de la interacción actual del usuario con el sistema.

En Android, todas las Views, por ejemplo, los elementos de interfaz del usuario en Android, se definen en ficheros XML y estos ficheros los necesita el sistema operativo. La búsqueda de las Views es un problema esencial ya que cada View tiene una única ID que es gobernada por la Activity correspondiente. Extraer esta información por medio de la Máquina Virtual Dalvik llevaría al cambios importantes en el diseño del sistema operativo haciéndolo no aplicable para dispositivos de consumo.

En Android, el contexto de aplicación es una clase que permite el acceso a recursos y clases de aplicación específica. Adicionalmente, se puede acceder a acciones tales como enviar y recibir efectos Android. En General, toda la información relativa a una Activity se almacena en su contexto. De nuevo, extraer esta información no es posible en un teléfono económico y una realización complementaria requeriría cambios importantes en el diseño del sistema operativo.

El documento US2011038542 describe un método para supervisar la actividad del usuario en el que se detecta la actividad del usuario en la unidad de entrada, se crea un resumen que representa todo los elementos mostrados en la pantalla y se crea una captura de pantalla de los elementos mostrados en la unidad de salida.

Resumen de la invención

Es un objetivo de la presente invención proporcionar un método para supervisar la actividad del usuario en dispositivos móviles y un producto de programa de ordenador correspondiente. Este objetivo se consigue con las características de las reivindicaciones independientes 1 y 3.

Un aspecto de la invención se refiere a un método para supervisar la actividad del usuario en dispositivos móviles en los que se proporcionan datos detallados acerca del contexto de aplicación y se proporciona soporte para los servicios del usuario con un mínimo de conversación técnica con el usuario del dispositivo móvil. Este aspecto de la invención se lleva a cabo utilizando una tecla genérica de almohadilla de la lista de gestión de elementos de interfaz del usuario que genera un identificador único para la correspondiente acción del usuario.

Se utilizan las pulsaciones de las teclas o cualquier otra actividad del usuario en un dispositivo de entrada con objeto de detectar la actividad del usuario en general. Se detecta la aplicación prioritaria en ejecución que muestra la aplicación actual en uso. Utilizando la lista de gestión de elementos de interfaz del usuario, se identifica la ventana exacta dentro de una aplicación que el usuario está viendo o editando en ese momento. La lista de elementos del usuario se resume para obtener un único identificador de la actividad del usuario. Se crea una captura de imagen para hacer verificable la actividad actual del usuario por el servicio de atención al cliente o por el mostrador de ayuda y para mejorar las posibilidades de depuración. Los elementos recogidos se almacenan en una base de datos local

y/o distante, pudiendo estar la base de datos en un servidor. En el lado del servidor, los datos recogidos pueden etiquetarse de manera precisa o por el contrario ser enriquecidos.

Un dispositivo móvil es un dispositivo computacional no estacionario construido sobre una plataforma computacional móvil, con habilidad computacional y conectividad avanzadas, que tiene una unidad de procesador capaz de ser operada por un sistema operativo. El dispositivo móvil puede ser por ejemplo un teléfono inteligente, una tableta, una PDA o similar.

La unidad de entrada es una unidad en la que el usuario puede introducir comandos para operar activamente el dispositivo móvil. Tal unidad de entrada puede ser una pantalla sensible al tacto por ejemplo, o un teclado o un ratón.

La aplicación en ejecución prioritaria es la aplicación que el usuario está usando, viendo o editando en ese momento. La aplicación de ejecución prioritaria puede ser la única aplicación ejecutada en el dispositivo móvil o puede ser una sólo de entre una diversidad de aplicaciones en ejecución. Identificar la aplicación paritaria es una etapa en la actividad de seguimiento del usuario. Esta es la base de las actividades detalladas de investigación de los usuarios dentro de las aplicaciones identificadas.

Un interfaz de usuario es una parte de una aplicación en la que tiene lugar la interacción entre el usuario y la aplicación. Típicamente el interfaz del usuario consta de diferentes elementos de interfaz del usuario. Una aplicación, que está adaptada para la interacción con usuario, ejecutándose en el sistema operativo con móviles, tiene una lista de gestión usuario interfaz elemento, que gestiona los elementos de interfaz del usuario y comprende información relativa de los mismos. Analizando los elementos de interfaz del usuario, será posible obtener información detallada acerca de la utilización de la aplicación actual del usuario. Los elementos de interfaz del usuario comprenden información acerca de si un usuario está en ese momento escribiendo un mensaje del texto, leyendo correos electrónicos o cambiando ajustes. Basándose en la estructura de una aplicación, los elementos de interfaz del usuario, incluyendo la lista de gestión de los elementos de interfaz del usuario, representan la información más precisa que describe la actividad del usuario.

La lista de los elementos de interfaz del usuario es resumida utilizando una función resumen, es decir, se genera una tecla almohadilla de la lista de gestión de los elementos de interfaz del usuario. De acuerdo con la convención de nombres, está tecla almohadilla es generalmente precisa. Por razones de gestión, el resumen se realiza con objeto de recibir un patrón comparable que se pueda usar para verificar la actividad del usuario. Debido a las convenciones de nombres de las aplicaciones, el código de resumen es único y sólo se generará, si un usuario correspondiente está usando exactamente el mismo elemento de interfaz del usuario que se registró previamente.

La creación de capturas de pantalla permite etiquetar cada elemento de interfaz del usuario identificado al tiempo que se le puede añadir información manual adicional. Estas descripciones manuales son útiles ya que en la mayoría de los casos no existe información pública adicional.

El método para supervisar la actividad del usuario en un dispositivo móvil comprende adicionalmente la etapa de almacenar al menos uno de los siguientes elementos: la información acerca de la actividad del usuario detectada y/o guardada, la información acerca de la aplicación prioritaria en ejecución, la lista resumida de elementos de interfaz del usuario y la captura de pantalla en una base de datos local y/o distante.

Dichos elementos se pueden guardar también primeramente en una base de datos local y enviada después a una base local distante en la que se almacenan; o dichos elementos pueden también ser almacenados en ambas bases de datos, la local y la distante, simultáneamente. Una ventaja del almacenamiento de datos en el lado servidor es que las actividades identificadas para todas las aplicaciones que tienen el mismo número incorporado en cualquier dispositivo móvil serán las mismas. Esto permitiría una comparación general de las complicaciones del usuario con una cierta aplicación.

En una realización de la invención, el método para supervisar la actividad del usuario en un dispositivo móvil puede comprender además la etapa de extraer al menos uno de los siguientes elementos adicionales: al menos un nombre de clase totalmente cualificado y/o al menos una ID y/o al menos una relación de al menos un elemento de interfaz del usuario de la lista de gestión de elementos de interfaz del usuario y almacenar los elementos extraídos en una base de datos local y/o distante.

En particular, los elementos pueden ser extraídos del elemento del usuario, que es el que está usando el usuario en ese momento, por ejemplo viendo o editando. Una relación del elemento de interfaz del usuario puede ser un elemento de interfaz del usuario hijo o padre existente que tenga alguna relación con el correspondiente elemento de interfaz del usuario.

En una realización de la invención, la aplicación en ejecución prioritaria se identifica interrogando a un gestor de aplicaciones en ejecución en cuanto se detecta actividad del usuario en el dispositivo de entrada.

El método de interrogar al gestor de aplicaciones en ejecución es particularmente ventajoso en dispositivos móviles con recursos del sistema limitados.

De acuerdo con una realización de invención, la captura de pantalla se crea leyendo los datos de una memoria temporal intermedia y creando una imagen a partir de esta memoria. Éste método se puede utilizar en particular en sistemas operativos que no tienen incorporada la posibilidad de realizar capturas de pantalla.

En una realización de la invención, la base de datos distante es una base de datos de un servidor distante.

De acuerdo con una realización de la invención, al menos uno de los elementos almacenados en una base de datos local y/o distante se etiqueta con una etiqueta respectiva precisa. La etiqueta precisa puede comprender, por ejemplo, una marca horaria, un nombre del cliente, un nombre de archivo o similar.

De acuerdo con la invención, un producto de programa de ordenador comprende uno o más medios interpretables por ordenador que tienen instrucciones ejecutables por ordenador para realizar las etapas de al menos uno de los métodos mencionados anteriormente.

Para realizar un método de supervisión de la actividad del usuario en un dispositivo móvil, es necesario configurar un sistema de supervisión que indique el estado actual a petición. El utilizar tal sistema de supervisión proporciona incluso la posibilidad de analizar flujos completos de acciones que adicionalmente mejorarán los resultados de las medidas de soporte o comerciales. En general, los datos supervisados están influenciados por tres sujetos principales: el hardware, el sistema operativo incluyendo todas las aplicaciones y datos manejados y el usuario. Cada uno de estos tres sujetos ejerce influencia mutua y da lugar valores a supervisar que describen en el estado actual de un sistema supervisado. Cuando se desarrolla un sistema de supervisión, no todos los valores que describen el estado de un dispositivo son accesibles. Especialmente, las acciones del usuario son difíciles de seguir ya que cada aplicación construye su propio dominio de acceso restrictivo desde fuentes externas. Es posible utilizar información genérica tal como el proceso en ejecución con objeto de estimar la actividad del usuario pero normalmente, estas medidas no darán información detallada sobre las acciones actuales del usuario.

La presente invención se puede utilizar de forma que servicios autorizados, tales como mostradores de ayuda a distancia, puedan ayudar a identificar, comprender y resolver problemas técnicos de los usuarios y detectar anomalías en las aplicaciones ejecutadas en un dispositivo móvil. El conocer especialmente el estado o situación técnica exacta se puede utilizar con objeto de analizar los problemas y proporcionar recomendaciones detalladas. Más aún, no sólo los servicios de soporte se beneficiarían de tal información: por ejemplo, los desarrolladores de aplicaciones pueden analizar la funcionalidad más utilizada de su aplicación para fijar prioridades en tareas futuras de desarrollo y los operadores y especialistas comerciales pueden utilizar estos datos para la comercialización basada en la actividad y sensible con el contexto y para los sistema recomendados.

Otro ejemplo de una posible utilización de la presente invención es el campo de la detección de software malicioso e intrusivo. Como un ejemplo en el que información más detallada puede ser muy útil es la detección de software malicioso que envía mensajes a servicios de fortuna para obtener beneficios. Se puede observar un aumento significativo de tal software malicioso dirigido a los dispositivos móviles y en particular a los teléfonos inteligentes. Utilizando supervisión normal, es posible detectar que se ha enviado un mensaje. En algunos casos, podría ser incluso posible detectar la actividad del usuario dando lugar a la detección positiva si los mensajes se enviaron sin intervención del usuario. Pero ya que se supone que los creadores de software malicioso están altamente cualificados, pueden ellos ocultar la transmisión del mensaje dentro de frases de actividad arbitraria del usuario. Sin tal información más comprensiva, no será posible para un sistema de detección de anomalías diferenciar entre un usuario que envía un mensaje y un usuario que sólo lee este mensaje corto mientras que un software malicioso está enviando mensajes a un servicio de fortuna.

Breve descripción de los dibujos

La figura 1 muestra esquemáticamente un listado para detectar una Activity prioritaria por medio de un ActivityManager en Android;

la figura 2 muestra esquemáticamente un listado para crear una captura de pantalla de una memoria temporal intermedia de tramas en Android;

la figura 3 muestra esquemáticamente una matriz de un pixel en la memoria temporal intermedia de tramas;

la figura 4 (a) muestra esquemáticamente diferentes etapas en una conversación de mensajes de texto;

la figura 4 (b) muestra contextos de aplicación adicional para una conversación de mensajes de texto;

la figura 5 muestra un servidor mostrando el contexto de aplicación en un dispositivo móvil utilizando Symbian; y

la figura 6 muestra un diagrama de actividad de un programa de ordenador en un dispositivo móvil que ejecuta un método para supervisar la actividad del usuario en un dispositivo móvil con Android de acuerdo con una realización de la invención.

Descripción detallada de la invención

La materia objeto de la presente aplicación - supervisar la actividad del usuario en un dispositivo móvil – está adaptada para su uso en cualquier tipo de sistema operativo con móviles debido a su aplicabilidad general. Una realización es el uso en un sistema operativo Android de Google. Dado que la ejecución técnica en un sistema operativo con móviles no puede ser obviamente resuelta debido a la carencia de soluciones normalizadas fuera de lo establecido, se necesitan diversos pequeños atajos. La siguiente descripción proporcionará una explicación detallada de las acciones realizadas.

Con objeto de detectar la actividad del usuario en un dispositivo móvil, se detecta la interacción del usuario con una unidad de entrada. Para no limitarse a una aplicación correspondiente, es preferible disponer de una detección general de accionamiento de teclas en un nivel del sistema operativo. Similarmente a otras distribuciones Linux, el Kernel de Android, resume los dispositivos de entrada como por ejemplo teclados, pantallas sensibles al tacto o ratones ópticos, como un conjunto de dispositivos serie localizados bajo /dev/input/. Los nombres de los dispositivos individuales pueden variar de acuerdo con el fabricante y con la configuración del dispositivo. Sin embargo, el formato de los mensajes emitidos por los dispositivos es el mismo en todos los teléfonos. Por lo tanto, se puede crear una herramienta, en particular una herramienta basada en Java-Native-Interface, que puede observar qué dispositivos serie están localizados bajo /dev/input/. Se genera un proceso individual por cada dispositivo de entrada, con un observador obteniendo y analizando los eventos de entrada y enviándolos a un registro de eventos, en particular un registro de eventos en un espacio Java. El observador puede estar realizado en lenguaje C y puede comunicarse con la aplicación Android, en particular utilizando Java-Native-Interface. De acuerdo con una realización, dado que los eventos de entrada, particularmente aquellos emitidos por un interfaz sensible al tacto son numerosos y casi siempre con incidentes, el registro de eventos puede clasificar su contenido por las marcas horarias y/o etiquetas precisas. Los eventos de entrada pueden constar de un punto de arranque, acciones intermedias (por ejemplo el desplazamiento de un ratón) y un punto de finalización. Tras cada entrada recibida, el registro de eventos puede ser examinado por un dispositivo de detección. Si se detecta un evento de entrada completo, se puede registrar en una base de datos y el registro de eventos puede ser limpiado para preservar la memoria.

En general, cualquier sistema operativo dispone de un gestor de tareas que comprende información acerca de las aplicaciones actualmente en ejecución en el sistema operativo. Cada aplicación ejecutada en Android que utiliza el permiso android: permission=GET TASKS puede acceder un caso de un ActivityManager, que es el gestor de tareas del sistema operativo Android. El ActivityManager dispone de información acerca de todas las actividades en ejecución. La información la conserva un objeto ActivityManagerRunningTaskInfo. Para recuperar la actividad prioritaria, por ejemplo, la actividad en ejecución prioritaria, el objeto ActivityManagerRunningTaskInfo tiene un campo llamado topActivity que recupera el componente actividad al tope de la pila historial de tareas.

De acuerdo con una realización, se le consulta al ActivityManager y el nombre de la aplicación en ejecución se compara con objeto de detectar un cambio de la aplicación en ejecución prioritaria. De acuerdo con otra realización, como se muestra esquemáticamente en la figura 1, al ActivityManager sólo se le consulta cuando hay actividad del usuario en el dispositivo de entrada, por ejemplo, en la pantalla sensible al tacto. De acuerdo con una realización, se puede almacenar el nombre de la aplicación en ejecución prioritaria reconocida y/o se etiqueta con una marca horaria y/o una etiqueta precisa.

Para supervisar la actividad del usuario, puede no ser suficiente identificar solamente la aplicación prioritaria. Puede ser conveniente analizar los elementos de interfaz del usuario de una aplicación en ejecución en este punto ya que proporcionaría información más detallada acerca de la interacción actual del usuario con el sistema. En Android, tal usuario de la lista de gestión de elementos de interfaz del usuario, que gestiona y maneja los elementos de interfaz del usuario, es preferible que sea una View Hierarchy en la que los elementos de interfaz del usuario son preferiblemente Views.

El código fuente de Android consta de una infraestructura para depurar las View Hierarchies. El depósito usado para este propósito se llama Hierarchy Viewer y está incluido en el Android System Developer Kit. Con objeto de que este depósito funcione, el emulador, por ejemplo, la herramienta, hospeda un servicio tipo telnet que vuelca la ventana actual completa con información detallada acerca de las propiedades de diseño de cada vista individual. De acuerdo con una realización, la herramienta incluida en el Android System Developer Kit está modificada con objeto aumentar su eficacia reduciendo el conjunto de información generada de una cierta View a lo siguiente: nombre de clase totalmente cualificado, ID y una relación con las otras Views, por ejemplo, información acerca de la existencia de una View hijo o una View padre. Esta reducción es un camino fiable para generar firmas/elementos de View Hierarchies que son únicas para cada pantalla y estado dentro de una aplicación.

De acuerdo con otra realización, la información relativa al proceso puede enriquecer los datos de supervisión, que se almacenan en una base de datos por ejemplo, nombre del proceso, ID del proceso, ID del proceso padre, ID del usuario, tamaño de la memoria y un número de procesos iniciados.

De acuerdo con una realización, se puede crear una captura de pantalla utilizando un interfaz de programación de aplicaciones (API) que lo proporciona el sistema operativo o un desarrollador tercero. De acuerdo con otra

realización, se puede crear una captura de pantalla leyendo los datos de la memoria temporal intermedia de tramas y creando una imagen a partir de estos datos. Este procedimiento es particularmente importante para sistemas operativos que no proporcionan medios de generar capturas de pantalla. Es posible crear capturas de pantalla leyendo los datos de la memoria temporal intermedia de tramas y creando una imagen de estos datos ya que la última imagen de la pantalla se almacenó en la memoria temporal intermedia de tramas. En Android, la memoria intermedia temporal de tramas se puede encontrar en /dev/graphics/fb0.

De acuerdo con una realización, los datos se interpretan como se presentan esquemáticamente en el listado mostrado la figura 2. Un bucle lee los datos de un pixel y los almacena en una nueva matriz de píxeles. El pixel se almacena en un conjunto de 16 bits y cada color tiene 5 bits, en el que el último bit esté vacío ya que no se almacena un valor alfa en la memoria temporal intermedia de esta realización. De acuerdo con esta realización, las líneas 18-19 representan la extracción del color azul, las líneas 21-22 representan la extracción del color verde y las líneas 24-25 representan la extracción del color rojo. En esta realización, como se mencionó anteriormente, no existe valor/canal alfa, ya que los últimos bits se enmascaran con FF (línea27).

La figura 3 muestra esquemáticamente tal matriz para los valores de color de un pixel en la memoria temporal intermedia de tramas. La figura 4 (a) presenta diferentes etapas en la conversación de mensaje de textos de acuerdo con una realización utilizando un dispositivo móvil con sistema operativo Android. La figura 4 (b) describe contextos adicionales de aplicación tales como un menú para la selección de los tonos de aviso del mensaje del texto y un menú de configuración de mensajes de texto de acuerdo con la realización. Cada contexto recibe su propio código resumen incluso sobre cambios menores: F896E95C7B7B506B2FB56C1BE46943A9, 3723573169F244901307DDD2D1572F292, 6566A266CCCBA359D5E708B5E6BB1, 12D65BB4E3EBB4FBD4F2B9E62EC7C14, y 9D7F5636E2989AAF607CC01.07A50EC4E. Utilizando estos códigos resumen, se consigue una exacta identificación de la actividad prioritaria del usuario. De acuerdo con una realización, la clase Bitmap de Android puede leer la nueva matriz de píxeles y crear un Bitmap que se puede almacenar como un archivo de imagen, preferiblemente una imagen PNG.

De acuerdo con una realización, los elementos extraídos, tales como por ejemplo la actividad detectada del usuario en el dispositivo de entrada o en la captura de pantalla creada, se almacenan en una base de datos distante en un servidor. De acuerdo con otra realización, se describe una aplicación de cliente capaz de utilizar los elementos extraídos, en particular la lista de gestión resumida de elementos de interfaz del usuario, para mostrar el contexto de aplicación del lado cliente a un servicio de atención al cliente registrado en el servidor. Este contexto de aplicación del lado cliente puede ser una captura de pantalla posiblemente enmendada con información extraída de la lista de gestión de elementos de interfaz del usuario, del elemento de interfaz del usuario actualmente utilizado o la lista de gestión resumida de elementos de interfaz del usuario. De acuerdo con otra realización, cada elemento y/o variedad de elementos relativos al mismo caso del cliente puede ser etiquetada manual o automáticamente, por ejemplo, con “creación de un mensaje de texto”. Adicionalmente, es posible etiquetar los elementos con cualquier otra información relativa al caso del cliente, como por ejemplo el nombre del cliente, la fecha, la hora, etc.

Un interfaz del servidor que presenta el contexto de la aplicación del lado cliente en el dispositivo móvil del cliente se muestra en la figura 5, en el que se utiliza el OS Symbian como sistema operativo con móviles en el dispositivo móvil. Las capturas de pantalla de los estados detectados se crean una vez y se transfieren al servidor para su visualización que básicamente habilita que personas trabajando a distancia investiguen el uso en un dispositivo.

De acuerdo con una realización, se diseña una aplicación de cliente en el dispositivo móvil dentro de una arquitectura modular. Cada tarea del cliente dispone de un módulo correspondiente que realiza la funcionalidad deseada. De acuerdo con una realización tal como se presenta esquemáticamente en la figura 6, el sistema comprende un Tracking Module y un Communication Module para Android.

En el Tracking Module, las IDs de pantallas se generan a partir de cada elemento de interfaz del usuario que fija la atención en ese momento en la pantalla, es decir, en el dispositivo de salida. Las Screen IDs se usan en conexión con servicios a distancia para gestionar de modo más completo y detallado los estados del sistema. Tales estados completos pueden consistir en códigos resumen, perfiles de accionamiento de teclas, listado de procesos funcionando en paralelo y así sucesivamente. En Android, las IDs de pantallas se pueden extraer de un SmartMobileService que se ejecuta como un servicio persistente Android. Las IDs de pantallas extraídas se pueden enviar por medio de un método DataSetsManager sendDataToService que emplea un QueryRestService para enviar finalmente todos los datos relevantes a un servicio responsable de la web. El método sendDataToService es un método de enganche que se puede ampliar para añadir información adicional de recolectores existentes de datos, como por ejemplo, DataCollector A, B, C y D en la figura 6.

En el Communication Module, se requiere enviar un valor entero llamado reqType, un valor de carácter para el formateo de los datos y un entero para un UID para un proceso de inicialización correspondientes a las firmas recibidas o enviar eventos a un Rest-Webservice. En el Communication Module, existe un método, getREQ(LinkedList<String>datas), que necesita los datos solicitados como una lista de conjuntos que se tienen que enviar a un servicio Rest-Web. Si es satisfactorio, el método devuelve entonces un caso de una clase ModelsType que mantiene una estructura completa XML analizada que se recibió a partir de una solicitud al servicio Rest-Web.

El propósito básico del Communication Module es comunicarse con componentes exteriores, por ejemplo, un servidor que puede almacenar todo los estados identificados con las correspondientes capturas de pantalla. Esta etapa no se precisa necesariamente para el propio proceso de identificación pero permite la integración de servicios adicionales basándose en la correspondiente información resultante del proceso de identificación.

REIVINDICACIONES

1. Un método para supervisar la actividad del usuario en un dispositivo móvil funcionando en un sistema operativo con móviles Android que consta de una unidad de entrada y una unidad de salida que comprenden las siguientes etapas según el orden siguiente:

(a) detectar la actividad del usuario en dicha unidad de entrada;

(b) identificar una aplicación en funcionamiento prioritaria;

(c) resumir un listado de gestión de elementos del interfaz de usuario de la aplicación en funcionamiento prioritaria;

(d) crear una captura de pantalla que comprende los elementos visualizados en dicha unidad de salida;

en el que la aplicación en funcionamiento prioritaria se identifica por solicitar el campo topActivity del objeto ActivityManager.RunningTaskInfo de un gestor de aplicaciones en funcionamiento siempre que se detecte la actividad del usuario en el dispositivo de entrada, y

en el que la actividad del usuario en dicho dispositivo de entrada la detecta un proceso de observación producido por cada dispositivo de entrada, obteniéndose los eventos de entrada por medio de dichos procesos de observación y enviando los eventos de entrada a un depósito de eventos por medio de dicho proceso de observación, examinando el depósito de eventos por medio un servicio de detección tras la recepción del evento de entrada y almacenando dichos eventos de entrada en una base de datos local o distante

2. El método según la reivindicación 1 que comprende además la etapa de registrar la actividad del usuario en dicho dispositivo de entrada.

3. El método según la reivindicación 1 o 2, que comprende además:

almacenar al menos uno de los siguientes elementos: la información acerca de la actividad del usuario detectada y/o registrada, la información acerca de la aplicación en funcionamiento prioritaria, el listado de gestión resumido del elemento de interfaz del usuario y la captura de pantalla en una base de datos local y/o distante.

4. El método según cualquiera de las reivindicaciones 1 a 3 que comprende además la etapa de vaciar dicho depósito de eventos tras el examen del depósito de eventos.

5. El método según cualquiera de las reivindicaciones 1 a 4, que comprende además:

extraer al menos uno de los siguientes elementos adicionales: al menos un nombre de clase completamente cualificado y/o al menos una ID y/o al menos una relación de al menos un elemento del interfaz del usuario a partir de la lista de gestión del elemento del interfaz del usuario y almacenar los datos extraídos en una base de datos local y/o distante.

6. El método según cualquiera de las reivindicaciones 1 a 5, en el que se crea la captura de pantalla leyendo los datos de una memoria intermedia temporal y creando una imagen procedente de esta memoria intermedia temporal.

7. El método según cualquiera de las reivindicaciones 1 a 6, en el que la base de datos distante es una base de datos de un servidor distante.

8. El método según cualquiera de la reivindicaciones 2 a 7, en el que al menos uno de los elementos almacenados en una base de datos local y/o distante se etiqueta con una etiqueta precisa respectiva.

9. Un producto de programa de ordenador que comprende uno o más medios interpretables por ordenador que disponen de instrucciones ejecutables por ordenador para realizar las etapas del método según cualquiera de las reivindicaciones 1 a 8.

```
1 for (ActivityManager.RunningTaskInfo rti
2       : runningTasks) {
3       // If we have actual front activity
4       if (rti.topActivity != null) {
5       // Check this activity
6       compName = rti.topActivity;
7       break;
8       }
9 }
```

Fig. 1

```
1 mask = 0
2 mask1 = 255/31
3 mask2 = 255/63
4
5 for(int i = 0; i<pixels.length; i++){
6   iarr[0] = inbuffer[2*i]<0 ?
7   inbuffer[2*i]+256 : inbuffer[2*i];
8
9   iarr[1] = inbuffer[2*i+1]<0 ?
10  inbuffer[2*i+1]+256 : inbuffer[2*i+1];
11
12  //Create the mask and shift.
13  //Its little endian so switch the position
14  mask = (iarr[1] << 8)| iarr[0];
15
16  pixels[i] = 0;
17
18  pixels[i] |=
19  ((((mask) & 0x1F)*mask1));
20
21  pixels[i] |=
22  (((mask>> 5) & 0x3F)*mask2) << 8;
23
24  pixels[i] |=
25  (((mask>> 11) & 0x1F)*mask1) << 16;
26
27  pixels[i] |= 0xFF000000;
28 }
```

Fig. 2

Bits	0	1	2	3
Color	Azul	Azul	Azul	Azul
Bits	4	5	6	7
Color	Azul	Verde	Verde	Verde
Bits	8	9	10	11
Color	Verde	Verde	Rojo	Rojo
Bits	12	13	14	15
Color	Rojo	Rojo	Rojo	Vacío

Fig. 3

Fig. 4 (a)

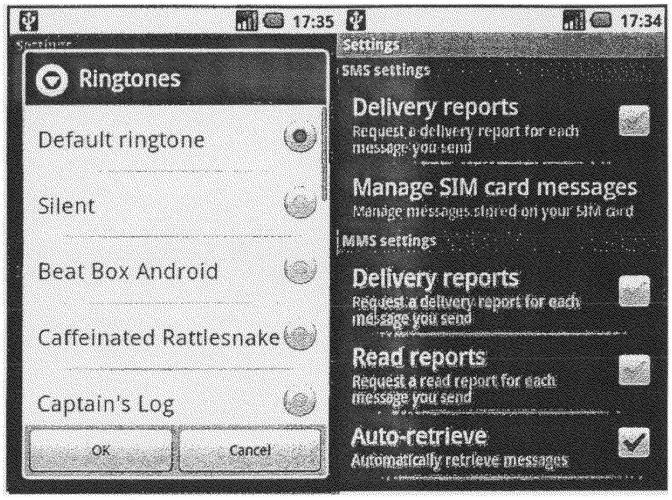
17:35
17:34
Settings
Ringtones
Default ringtone
Silent
Beat Box Android
Caffeinated Rattlesnake
Captain's Log
OK
Cancel
SMS settings
Delivery reports
Request a delivery report for each message you send
Manage SIM card messages
Manage messages stored on your SIM card
MMS settings
Delivery reports
Request a delivery report for each message you send
Read reports
Request a read report for each message you send
Auto-retrieve
Automatically retrieve messages

Fig. 4(b)

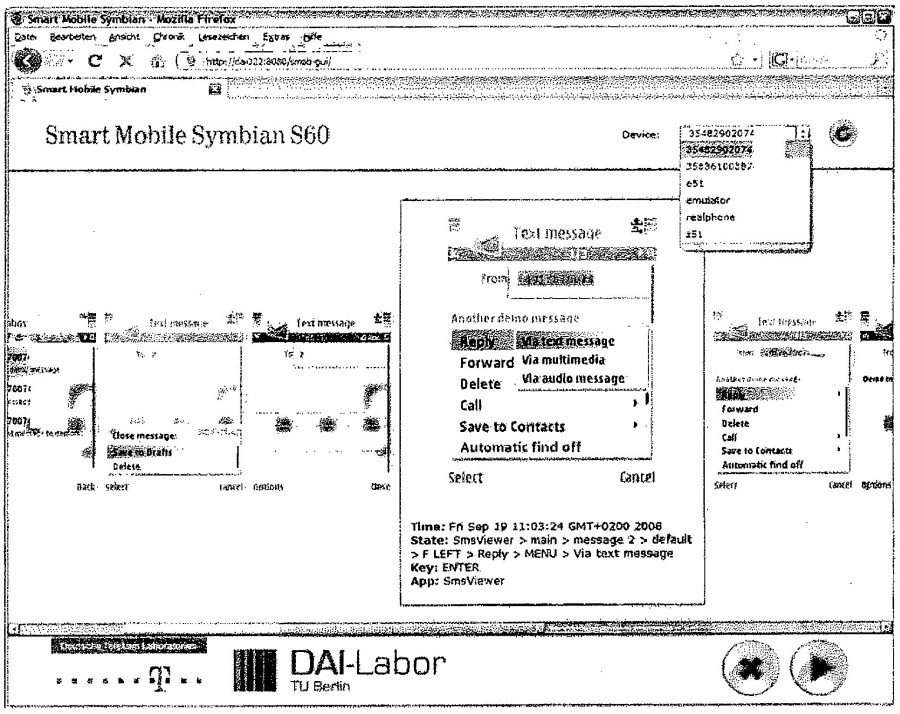
Smart Mobile Symbian S60
Text message
Another demo message
Reply
Via text message
Forward
Via multimedia
Delete
Via audio message
Call
Save to Contacts
Automatic find off
Select
Cancel
Time: Fri Sep 19 11:03:24 GMT+0200 2008
State: SmsViewer > main > message 2 > default > F LEFT > Reply > MENU > Via text message
Key: ENTER
App: SmsViewer
DAI-Labor
TU Berlin

Fig. 5

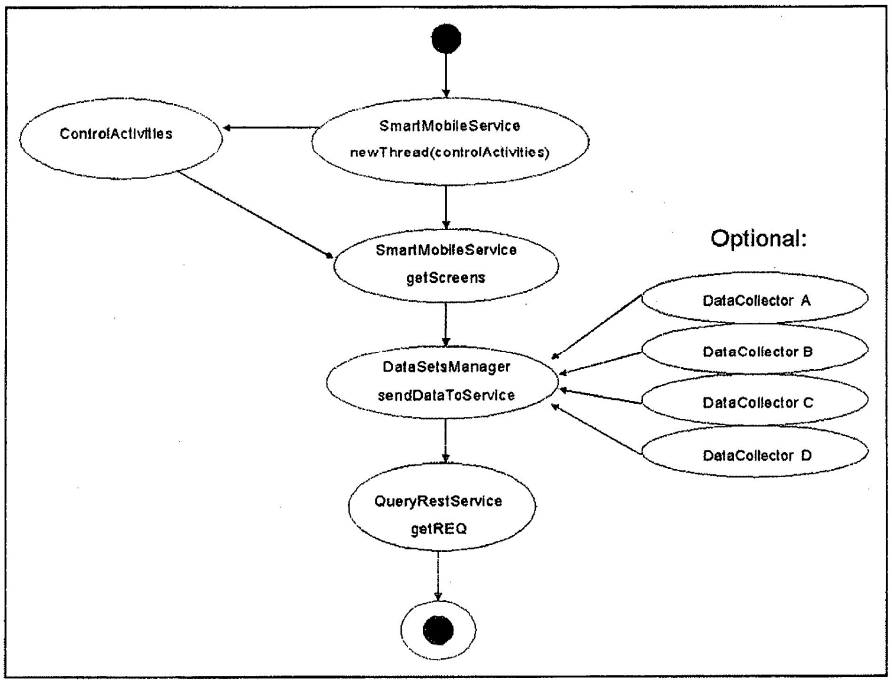
ControlActivities
SmartMobileService
newThread(controlActivities)
SmartMobileService
getScreens
Optional:
DataCollector A
DataCollector B
DataCollector C
DataCollector D
DataSetsManager
sendDataToService
QueryRestService
getREQ

Fig. 6a