Lámpsakos
ISSN: 2145-4086
Universidad Católica Luis Amigó

Velasquez, Sandra Milena; Monsalve Sossa, Doris Elena; Zapata, Mauricio Eduardo; Gómez Adasme, Marta Ester; Ríos, Juan Pablo

Pruebas a aplicaciones móviles: avances y retos

Lámpsakos, núm. 21, 2019, Enero-Junio, pp. 39-50
Universidad Católica Luis Amigó

DOI: https://doi.org/10.21501/21454086.2983

Disponible en: https://www.redalyc.org/articulo.oa?id=613964508005

DOI: https://doi.org/10.21501/21454086.2983

Pruebas a aplicaciones móviles: avances y retos

Mobile applications testing: advances and challenges

Sandra Milena Velásquez*
Doris Elena Monsalve Sossa**
Mauricio Eduardo Zapata***
Marta Ester Gómez Adasme****
Juan Pablo Ríos*****

(Recibido el 11-06-2018. Aprobado el 12-11-2018)

Estilo de citación de artículo:
S. M. Velásquez, D. E. Monsalve Sossa, M. E. Zapata, M. E. Gómez Adasme, y J. P. Ríos "Pruebas a aplicaciones móviles: avances y retos, *Lámpsakos*, No. 21, pp. 39-50. (enero-junio, 2019). DOI: https://doi.org/10.21501/21454086.2983

Resumen.
En los últimos años, las aplicaciones móviles han penetrado todos los mercados de cualquier industria, razón por la que cada vez es más importante para los desarrolladores conocer las técnicas y métodos de prueba específicos para aplicaciones móviles. Dichas aplicaciones son diferentes a las de la web y de escritorio tradicionales, por lo que requieren un enfoque distinto en su construcción. Este enfoque debe generar confiabilidad en el usuario final, quien puede ser cualquier persona que utilice un teléfono inteligente. Tal situación conlleva nuevos desafíos para las empresas o personas que desarrollan aplicaciones móviles. Por este motivo, este trabajo busca reflexionar sobre los avances de las técnicas y métodos de prueba para las aplicaciones móviles, así como de los retos que las empresas y los desarrolladores pueden tener a la hora de crear una estrategia para probar sus aplicaciones y poder lanzarlas al mercado disminuyendo el riesgo de errores en producción.

Palabras clave: Aplicaciones móviles; Pruebas basadas en el riesgo; Desarrollo; Pruebas para aplicaciones móviles; Avances; Riesgo; Técnicas; Métodos; Confiabilidad; Ingeniero de Software.

* Magíster en Ingeniería, Centro de Servicios y la Gestión Empresarial. Grupo de investigación GIGAT. Antioquia, Colombia. Correo electrónico: smvelasquez@sena.edu.co

** Especialista en Pedagogía de la Virtualidad, Ingeniera de Sistemas, Centro de Servicios y la Gestión Empresarial. Grupo de investigación GIGAT. Antioquia, Colombia. Correo electrónico:dmssos@misena.edu.co

*** Ingeniero en Sistemas, Analista de Desarrollo Choucair Testing, Antioquia, Colombia. Correo electrónico: mezapata@choucairtesting.com

**** Especialista en redes corporativas e integración de tecnologías, Ingeniera de Sistemas, Centro de Servicios y la Gestión Empresarial. Grupo de investigación GIGAT. Antioquia, Colombia. Correo electrónico: Mega2808@misena.edu.co

***** Magíster en Ingeniería, R&D&i Manager Choucair Testing, Antioquia, Colombia. Correo electrónico: jprios@choucairtesting.com

DOI: https://doi.org/10.21501/21454086.2983

Abstract:

In recent years, mobile applications have pervaded all markets in any industry, this is why it is increasingly important for developers to know the techniques and specific testing methods of mobile applications. These applications are different from those of the traditional web and desktop, so they require a different approach in their development. This approach should generate reliability in the final consumer, who could be any person who uses a smartphone. This situation brings new challenges for companies or people who develop mobile applications. For this reason, this work intends reflecting on the advances in techniques and testing methods of mobile applications, as well as the challenges that companies and developers may have when creating a strategy to test their applications and release them to the market, reducing the risk of error in production.

Keywords: Mobile apps; Risk-based testing; Development; Testing for mobile applications; Advances; Risk; Techniques; Methods; Reliability; Software engineer.

DOI: https://doi.org/10.21501/21454086.2983

1. INTRODUCCIÓN

Cada día las personas utilizan más teléfonos inteligentes de alta gama basados en sistemas operativos móviles modernos que ofrecen una excelente conectividad y una computación avanzada gracias a sus procesadores, altas resoluciones en sus pantallas táctiles, sus avanzados sensores, GPS, acceso por medio de Wi-Fi, datos de alta velocidad, entre otras características [1]; todo ello se ha visto reflejado en el aumento del uso de las aplicaciones móviles. Estas pueden considerarse como software desarrollado para ser ejecutado en dispositivos como tabletas, teléfonos o relojes inteligentes que poseen un sistema operativo apto para ello.

En los últimos años, la tecnología de internet móvil ha penetrado todos los mercados de las industrias, lo cual ha ocasionado que la seguridad de las aplicaciones se está volviendo cada vez más importante [2], por lo tanto, los desarrolladores deben encontrar nuevas formas de revisar y probar los contenidos, de manera que se puedan depurar las aplicaciones antes de salir al mercado, con el fin de evitar errores [3]. Es de anotar que no solo la seguridad es importante, también existen otros atributos que deben ser evaluados, como por ejemplo el desempeño.

Sin embargo, aprender a realizar pruebas a software no es tan simple como parece, a pesar de que en la actualidad existen esquemas de certificación de conocimiento de prueba como ISTQB, ISEB, GTB [4], [5], [6], libros de gran importancia para el aprendizaje del tema y estándares internacionales para la implementación de estas como ISTQB, BS 7925, IEEE 829, 1008, 1012, ISO / IEC 29119, SWEBOK [7]. El éxito de la realización de las pruebas de software obedece a las habilidades, conocimientos, intuición y experiencia de las personas que conforman el equipo de pruebas [8].

Ahora bien, para un probador o ingeniero de pruebas de software siempre existirán unas actividades primordiales y esenciales que no cambiarán, como lo son: el planear la estrategia, diseñar casos de prueba, ejecutar los casos de prueba diseñados, observar y analizar los resultados. A estas se suman otras actividades: métodos modernos, las pruebas exploratorias, la verificación en tiempo de ejecución [7], lo cual incluye el conocimiento en temas como la automatización y la administración del tiempo de prueba, en la lista de tareas del probador.

Las pruebas en aplicaciones móviles, como cualquier otro tipo de pruebas de software, se basan en la verificación y validación de las métricas priorizadas de acuerdo con la experiencia del usuario, la población a la cual va dirigida, el tipo de aplicación móvil, el tipo de dispositivo, la plataforma tecnológica empleada para su desarrollo y ejecución; que el probador establece a través del plan de pruebas y el diseño de casos de prueba, para los cuales debe aplicar técnicas de estimación y diseños (caja negra y caja blanca). Para la ejecución, el probador puede utilizar herramientas de automatización, como Appium [30], para lograr mayor cobertura y velocidad de la prueba. Durante esta etapa debe evaluar los resultados, así como reportar las situaciones que no están acordes con lo esperado. Finalmente debe realizar la gestión de su prueba, con el objeto de no incurrir en pérdidas por la no ejecución de casos, la no inclusión de características prioritarias o el desplazamiento de las fechas de entrega por las desviaciones encontradas.

2. FUNDAMENTOS DE PRUEBAS DE SOFTWARE

Atributos de calidad

Durante el ciclo de vida de los proyectos de software, los desarrolladores realizan las pruebas como una actividad que busca avalar que el producto cumpla con los requerimientos de los usuarios [9], garantizando su calidad.

Acerca de la calidad de productos de software encontramos varias normas como la ISO ISO/IEC 25021, que describe la evaluación de los requerimientos [10], la ISO/IEC 25000, SQuaRE (requisitos y calidad del software), la cual realiza un compendio de la ISO/IEC 14598 y la ISO/IEC 9126 orientada directamente al

DOI: https://doi.org/10.21501/21454086.2983

evaluador [11], [12]; la ISO/IEC 29119, norma que hace una recopilación de los criterios de calidad presentes en la ISO/IEC 9126 y que está orientada a la funcionalidad y la satisfacción del cliente [11], [12], [13]; y la IEEE-610, en la cual se relacionan los requisitos indispensables del producto (Fig. 1).

Fig. 1: Requisitos de calidad de productos de software. Fuente: adaptado de [14].

Dentro de estos requisitos de calidad del producto de software, la usabilidad es considerada como uno de los más importantes. Dicho atributo demuestra la facilidad con la que un consumidor puede utilizar una aplicación de software [15].

En el estudio de Harrison et al. [16] se realizó una revisión específica de los modelos para probar el requisito de usabilidad en aplicaciones móviles. La investigación encontró que la mayoría de los modelos de usabilidad implementados para probar aplicaciones móviles son incompletos, ya que solo se enfocan en tres atributos de usabilidad: efectividad, eficiencia y satisfacción; y relegan otros importantes, como la sobrecarga cognitiva. Para abordar este problema, [16] propusieron un nuevo modelo de usabilidad conocido como PACMAD (People At the Center of Mobile Application Development) [16].

Se identificó además que el atributo de accesibilidad en ocasiones puede verificarse estáticamente, pero los widgets de la interfaz de usuario a menudo se crean dinámicamente y no son susceptibles de comprobación estática, lo cual implica un desafío para el probador, pues debe desarrollar un conjunto de pruebas exhaustivas para afrontar esta limitante. Es por lo que investigadores han presentado como alternativa la implementación de pruebas automáticas para analizar el acceso a las aplicaciones móviles, ejemplo de esto es la herramienta MATE (Mobile Accessibility Testing) [17], que explora automáticamente las aplicaciones y ejecuta diferentes controles para problemas de accesibilidad relacionados con la discapacidad visual.

Por lo tanto, se ha evidenciado en esta búsqueda que los estudios recientes hablan más de estudios de pruebas del área de automatización [18], [19] sin hacer énfasis en los tipos de pruebas, técnicas, y en las métricas que demuestren la conformidad del ciclo de vida del desarrollo, el cual es de más utilidad para personas que apenas están iniciando en este mundo de pruebas de software, en especial del área de aplicaciones móviles.

Un estudio de revisión de la literatura realizado por Muccini [20], acerca de los desafíos de las pruebas de aplicaciones móviles, sugiere más direcciones de investigación, ya que en el estudio identifica varios vacíos en las áreas de pruebas de servicios móviles, automatización de pruebas e integración de pruebas para aplicaciones móviles [20].

Marcos de trabajo y niveles de pruebas

En cuanto a marcos de trabajo para pruebas de software, tenemos principalmente dos: los secuenciales y los iterativos e incrementales. De ambos se pueden apreciar algunos ejemplos en la Fig. 2.

Actualmente, los marcos de trabajo preferidos para la creación de aplicaciones son los orientados al agilismo [21]; esto no debe constituirse en una camisa de fuerza para el probador, pues independiente del marco de trabajo, los niveles de pruebas siempre aplicarán.

DOI: https://doi.org/10.21501/21454086.2983

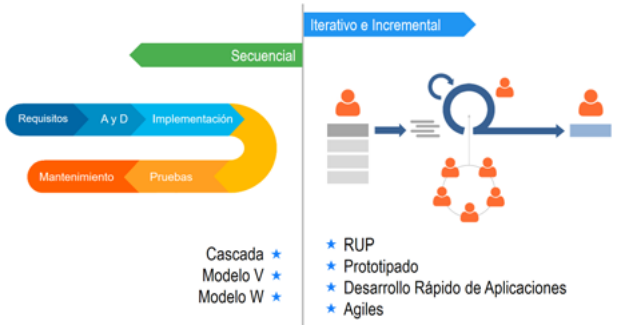

Fig. 2: Marcos de trabajo de desarrollo.

Cuando se habla de nivel de pruebas se hace referencia al alcance de evaluación del producto, lo cual depende de su propósito, uso, comportamiento o estructura [22]. Según el alcance, estos niveles están compuestos por una serie de pruebas como lo son la prueba unitaria, la prueba de componentes, prueba de integración, prueba de sistemas, prueba de aceptación y la prueba de regresión (tabla 1).

Tabla 1. Tipos de pruebas según la clasificación por niveles

Tipo de prueba	Descripción
Pruebas unitarias [23]	Buscan probar las unidades más pequeñas del software, el componente o módulo de software, con el objetivo de validar cada uno, teniendo en cuenta que las metodologías de desarrollo de software separan entre los diferentes programadores, la complejidad de los requisitos funcionales sobre los no funcionales.
Pruebas de componentes [24]	Seleccionan un módulo, lo aíslan del resto del código, con el fin de establecer si se comporta exactamente como se espera. Es decir, cada componente se prueba por separado antes de integrarlo en un servicio.
Pruebas de integración [22, 24]	Se concentran en verificar que los diferentes componentes del software estén bien ensamblados, una vez que se probaron unitariamente.
Pruebas de sistema [22]	Buscan probar el sistema en su conjunto y con otros sistemas con los que se relaciona, con el fin de verificar que las especificaciones, tanto técnicas como funcionales, se cumplen.
Pruebas de aceptación [22, 24]	Son realizadas por los usuarios que verifican que el sistema o aplicación esté listo para la salida a producción.
Pruebas de regresión [24]	Ejecución de casos de pruebas anteriormente efectuados cuando se implementan cambios en el software.

Se debe tener presente que cada nivel de pruebas está a cargo de un responsable de su ejecución, como se muestra en la Fig. 3.

Fig. 3: Niveles de pruebas y sus responsables.

3. ELEMENTOS PRESENTES EN LA TECNOLOGÍA MÓVIL

El probador debe tener presente siempre en su labor el mapa mental de la Fig. 4.

Redes

Las redes de transmisión celular son la infraestructura base de la tecnología móvil. El probador considera cómo operará la aplicación en la red, que puede funcionar bajo alguno de los siguientes modos: nunca, parcialmente o siempre conectada. Esto es importante, pues el acceso a los datos puede hacerse a través del mecanismo de inserción ("push") o extracción ("pull"). La inserción hace referencia a cuando el servidor envía información al cliente, y la extracción cuando el cliente solicita la información al servidor.

También se debe considerar la velocidad de conexión del tipo de red. Las aplicaciones cuyo funcionamiento exija una conexión permanente deben considerar el tipo de red de transmisión, pues no tener presente las características de la infraestructura puede conllevar a problemas de rendimiento de la aplicación. La tabla 2 describe las velocidades para las redes más conocidas.

DOI: https://doi.org/10.21501/21454086.2983

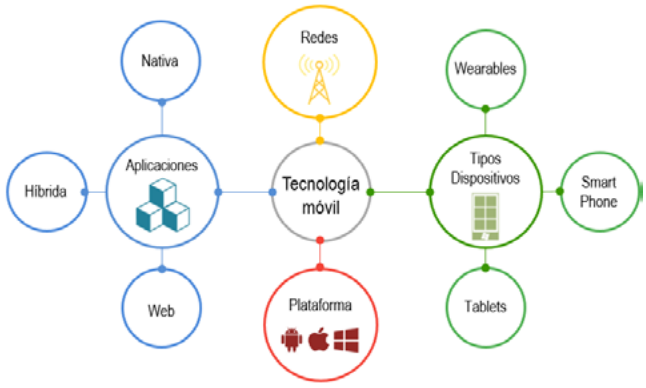

Fig. 4: Mapa mental de la tecnología móvil.

Tabla 2. Velocidades de conexión de las redes de transmisión celular más conocidas

Red	Denominación	Velocidad
GSM	2G	Hasta 0,009 Mbps de descarga
GPRS	2.5G	Hasta 0,08 Mbps de descarga
EDGE	2.75G	Hasta 0,236 Mbps de descarga
UMTS	3G	Hasta 0,384 Mbps de descarga
HSPA	3.5G	Hasta 7,2 Mbps de descarga
HSPA+	3.75G	Hasta 22 Mbps de descarga
LTE	4G	Hasta 75 Mbps de descarga

Tipos de dispositivos

Debido a la cantidad de fabricantes existe una numerosa oferta de dispositivos móviles. El probador debe apoyarse en estudios del mercado y sitios que recopilen estadísticas de uso, con el objeto de seleccionar los dispositivos relevantes para su prueba.

Actualmente se clasifican los dispositivos en tres grandes grupos: teléfonos inteligentes, tabletas y dispositivos incorporados ("wearable"). Esta clasificación se fundamenta en el uso que se le da a cada grupo.

Otro elemento que debe tener presente relacionado con los dispositivos móviles son los componentes del mismo. De acuerdo al dispositivo seleccionado para la prueba, debe conocer qué tipo de elementos posee, con el objeto de estructurar los diferentes casos de prueba relacionados con las funcionalidades de la aplicación. Casos de prueba que pueden estar asociados a conexiones wifi, GPS, bluetooth, tecnología NFC, batería, tarjeta SIM, memoria, sensor de movimiento, sensor de luz, sensor de localización, entre otros.

Tipos de aplicaciones

Existen tres tipos de aplicaciones: nativas, híbridas y web. A diferencia de las aplicaciones web (basadas en el navegador), las aplicaciones nativas e híbridas están instaladas físicamente en el dispositivo y, por lo tanto, siempre están disponibles para el usuario.

Dependiendo del tipo de aplicación, el probador establece qué tipos de casos debe incorporar. Por ejemplo, para las aplicaciones nativas e híbridas es indispensable adicionar casos para revisar la instalación y desinstalación de la aplicación en el dispositivo.

Para el caso de una aplicación web, el probador debe considerar la arquitectura de la solución. Es decir, si es una arquitectura de una sola capa o multicapa. Dependiendo de esto y del volumen de usuarios esperado para la aplicación, se pueden incluir casos de prueba de desempeño.

La tabla 3 muestra un comparativo de las características básicas para cada tipo de aplicación.

Tabla 3. Comparación de los tipos de aplicaciones móviles

Característica	Nativa	Híbrida	Web
Uso de características específicas del dispositivo	Alta	Media	Baja
Uso de APIs nativas	Alta	Media	Baja
Uso de gráficos	Alta	Media	Media
Calidad de la interfaz de usuario	Alta	Media	Baja
Capacidad de actualización	Baja	Media	Alta
Portabilidad	Ninguna	Alta	Alta
Instalación en el dispositivo	Sí	Sí	No

DOI: https://doi.org/10.21501/21454086.2983

Plataformas

La plataforma o sistema operativo móvil es el encargado de manejar, controlar y gestionar todo lo relacionado con los recursos de hardware de un dispositivo móvil. Esto trae consigo grandes retos para los fabricantes y desarrolladores, pues deben optimizar la gestión de procesos, el manejo de memoria principal y secundaria y la gestión de archivos, sin sacrificar el rendimiento y la seguridad.

Hoy en día el mercado es dominado por dos sistemas operativos móviles: Android y iOS. Ambos tienen definida una estructura por capas, es decir, cada nivel o capa definida representa una abstracción de la arquitectura del sistema operativo, lo que permite un mayor entendimiento del funcionamiento de estos. Estas capas son: el kernel, el middleware, el entorno de ejecución y la interfaz de usuario.

El probador debe conocer los elementos básicos que conforman la arquitectura de ambos sistemas operativos, así como su fragmentación (diferentes versiones), con el objeto de seleccionar las combinaciones de marca del tipo de dispositivo, versión del sistema operativo y el tamaño de pantalla para los casos de prueba.

4. PRUEBAS DE APLICACIONES MÓVILES

Planeación

Una vez establecido el marco de trabajo de desarrollo bajo el cual se construirá la aplicación móvil, el probador debe establecer la estrategia que servirá de hilo conductor para la prueba. Debe considerar varios elementos para su elaboración: el alcance deseado (depende del marco de trabajo); los supuestos que se deben tener presente; las renuncias, es decir, que no hará parte del alcance; los tiempos estimados para la prueba; y la repartición de actividades entre los integrantes del equipo de pruebas.

Un aspecto muy contundente en el momento de elaborar la planeación para la prueba de una aplicación móvil es la utilización del triángulo de las restricciones [25]; dicha herramienta para este tipo de proyectos es altamente efectiva, pues ayuda a delimitar las aristas asociadas con la fecha de entrega, la funcionalidad a probar y los recursos a emplear con base en las limitantes y/o prioridades de cada una de las aristas.

Otro elemento útil y que debe estar en la planeación es la utilización del riesgo como herramienta, ya sea como apoyo para priorizar las funcionalidades a probar, o como herramienta para gestionar las situaciones que puedan provocar desviaciones en esfuerzo o tiempos estimados.

Adicionalmente, debe considerar tres aspectos fundamentales que delinearán el marco de la prueba:

El público objetivo

A diferencia de la prueba de un aplicativo web o de escritorio, la de una aplicación móvil normalmente está destinada a ser empleada por un segmento no tan homogéneo como el de los otros tipos de aplicaciones. Por ello se hace necesario, y como medida que se anticipe a un fracaso de la aplicación, conocer, en lo posible, el mayor detalle del segmento poblacional al que va dirigida la aplicación. Ello involucrará saber de dicho segmento: edades, ubicación geográfica, tipo de dispositivos que predominan, tipos de red presentes, entre otros. Todas estas situaciones generan información muy útil al momento de plantear la estrategia.

Los atributos de calidad funcionales, base de la aplicación móvil

El probador debe conocer los requisitos o historias de usuarios asociadas a la aplicación móvil. Esto, aparte de dar la idea de cuál es el propósito general de la aplicación construida, permite plantear los casos de prueba esenciales para probar las funcionalidades de la aplicación.

DOI: https://doi.org/10.21501/21454086.2983

Los atributos no funcionales

Se deben contemplar en la estrategia de pruebas otros aspectos o comportamientos que podrán afectar el adecuado funcionamiento de la aplicación. Estos aspectos hacen a los atributos no funcionales que pueden evaluarse para la aplicación, como lo son: fiabilidad, compatibilidad, portabilidad, eficiencia de desempeño, mantenibilidad, usabilidad y seguridad; descritos en la Fig. 1.

Diseño de casos

Esta actividad involucra la revisión de las bases de pruebas existentes [26] para la aplicación, análisis de la testability, identificar y priorizar las condiciones de pruebas, el diseño de los casos de prueba (positivos y negativos), diseños de las rutinas (scripts) de automatización y la obtención de datos para la prueba.

Técnicas comunes

Para la creación de los casos de prueba se debe recurrir a las técnicas comunes de diseño de casos, tanto desde el análisis dinámico, como estático.

Dentro del análisis dinámico (se ejecutan los componentes), el probador puede emplear técnicas de caja negra como partición de equivalencias, análisis de valores límite, pruebas de transición de estado, tablas de decisión, algoritmo dual ("pairwise") y técnicas estadísticas. También puede hacer uso de técnicas de caja blanca, siempre y cuando tenga acceso al código de la aplicación; dentro de estas puede emplear el análisis por cobertura de sentencia, rama, condición y camino.

Para el análisis estático (no hay ejecución de componentes) puede emplear técnicas como revisiones guiadas ("walkthroughs"), análisis del flujo de control, análisis del flujo de datos y métricas compilador/analizador.

Adicionalmente, el probador puede basarse en la experiencia como la predicción de errores ("error guessing") en la práctica o las pruebas exploratorias ("exploratory testing").

Casos propios de la tecnología móvil

Debido a las características de la tecnología móvil, se deben contemplar otras situaciones que pueden afectar el comportamiento funcional o transaccional de la aplicación. El probador, de acuerdo con la estrategia de prueba diseñada, puede evaluar las siguientes características para una aplicación:

- El proceso de instalación y desinstalación
- El control de errores que posee
- Cambios de red
- El comportamiento en escenarios multitarea
- Las visualizaciones en modo horizontal y vertical
- Pruebas en diferentes tipos de resoluciones y tamaños de pantalla
- Si se presenta integración con servidor, tener presente los diferentes escenarios de intercambio de información
- Comprobar si la aplicación afecta o es afectada por otras aplicaciones
- El comportamiento frente al uso de los controles propios del dispositivo
- Adicionar escenarios asociados a características propias de estos dispositivos como girar, agitar, realizar diferentes tipos de gestos táctiles sobre la pantalla
- Pruebas de interrupciones por mensajes, llamadas, conexión y desconexión de cables

DOI: https://doi.org/10.21501/21454086.2983

- Comportamiento frente a los diferentes métodos de entrada incluyendo los sensores
- Pruebas de consumo de energía.

Estas son algunas de las situaciones que pueden ser contempladas desde el punto de vista del comportamiento móvil de la aplicación y que pueden afectar el funcionamiento de la misma en el dispositivo móvil.

Ejecución

Durante esta etapa, el probador debe verificar la completitud de casos y datos de prueba (priorización y organización en procedimientos), ejecutar la prueba de acuerdo con la estrategia, los riesgos y la priorización de funcionalidades a evaluar [27]. Realizar el registro de los resultados (evidencia), análisis de los mismos y el reporte de incidentes (errores, preguntas, sugerencias y hallazgos) encontrados durante la ejecución.

Tipos de ejecución

Los casos de prueba pueden ser ejecutados en forma manual o automatizada. El probador puede emplear las técnicas de automatización de casos de prueba para lograr mayor cobertura de la prueba y velocidad en la ejecución de casos repetitivos así, puede centrar su esfuerzo en la ejecución de los casos de pruebas críticos.

Una buena práctica es la automatización de errores encontrados, con el fin de reutilizar estas rutinas en nuevas versiones de la aplicación. Con esto se logra eficiencia en la ejecución de la prueba [28].

El probador debe estar evaluando constantemente si se cumplieron los criterios de finalización de la prueba, con el objeto de ejecutar la regresión y dejar en un estado terminal las incidencias reportadas.

Herramientas

Durante la ejecución de la prueba se puede hacer uso de diferentes tipos de herramientas que generan velocidad y control en el avance de la prueba. Básicamente, el probador de aplicaciones móviles debe tener conocimiento en herramientas que permitan automatizar, emular y simular la aplicación, reportar incidencias (bugtracker), comunicarse con el equipo, gestionar el ciclo de vida de la prueba [29].

Hoy en día, con los nuevos marcos de construcción de software, existe una gran variedad de herramientas que se emplean durante la ejecución del proyecto de software. Muestra de ello es DevOps, en donde cada etapa (colaborar, construir, probar, desplegar y ejecutar) posee una extensa gama de herramientas. La tabla 4 muestra ejemplos de herramientas para las etapas de DevOps.

Tabla 4. Ejemplos de herramientas empleadas en las etapas de DevOps

Etapa	Tipo de herramienta	Ejemplos
Colaborar	Gestión del ciclo de vida	Jir, Mingle, Trello, Visual Studio Team Foundation Server, Asana
	Comunicaciones	Slack, Microsoft Teams, HipChat, Flowdock, Nestor
	Compartir conocimiento	Github pages, Hugo, Flarum, Open API, Confluence
Construir	SCM/VCS	Git, Github, Gitlab, Bitbucket, Gitbucket
	Integración continua	Jenkis, Travis CI, Bamboo, Wercker, Codeship
	Construir	Sbt, Gradle, Grunt, Maven, Docker
	Gestión de base de datos	DBmaestro, DBDeploy, Flyway, Flocker, Redgote
Probar	Probar	Selenium, JUnit, Appium, Cucumber, JMeter, Pytest
Desplegar	Desplieque	Octupus Deploy, Rundeck, Nolio, Elasticbox, Spinnaker
	Gestión de la conFig.ción	Puppet, Chef, CFEngine, Ansible, Vagrant
	Gestión de artefactos	Dockerhub, Quay, Registry, Bower, Archiva
Ejecutar	Nube / IaaS / PaaS	Amazon Webservices, Dokku, Flynn, Microsoft Azure, Heroku
	Orquestación y Agendamiento	Kubernetes, Rancher, Swarm, Mesosphere, Nomad
	BI y Monitoreo	Datadog, Google Analytics, Dynatrace, Kibana, Sentry

DOI: https://doi.org/10.21501/21454086.2983

Gestión y entrega

El probador, durante el ciclo de vida de la prueba, debe realizar la toma de datos con base en las métricas definidas, con el objeto de decidir las acciones que se deben realizar (medidas correctivas o preventivas) a tiempo. Adicionalmente, en caso de cambios en el alcance, validar el impacto de estos en la estrategia de la prueba.

Una vez finalizada la prueba, el probador reporta las lecciones aprendidas, las mejoras al proceso, la deuda técnica que genera en el proyecto para futuras versiones, y actualizar la base de datos de la prueba con los casos relevantes, errores automatizados, las rutinas de automatización y los datos que pueden ser reutilizados en comprobaciones futuras de la aplicación.

5. CONCLUSIONES

Con la incursión de las tecnologías móviles y sus características, se hace necesario que los probadores amplíen sus habilidades para evaluar las aplicaciones diseñadas para dispositivos móviles. Estas habilidades se deben centrar en evaluar la aplicación a probar desde varias dimensiones; no solo el aspecto funcional debe primar.

En la actualidad, la fragmentación móvil es el mayor desafío que enfrentan los constructores de aplicaciones para dispositivos móviles, pues la cantidad de opciones a probar dadas por la combinatoria de las variables: sistemas operativos móviles, versión del sistema operativo móvil, tamaño de la pantalla y fabricante del dispositivo, es enorme. Por tal motivo, surge la necesidad de encontrar estrategias de ejecución en los dispositivos físicos y otros entornos, que generen un balance en los costos y la cobertura del proyecto.

AGRADECIMIENTOS

Al Sistema de Investigación, Desarrollo Tecnológico e Innovación del SENA (SENNOVA) por el financiamiento del proyecto "Modelo de pruebas para mitigar los riesgos de fragmentación de dispositivos móviles bajo modelos en-sito y nube pública".

REFERENCIAS

[1] S. Zein, N. Salleh, and J. Grundy, "A systematic mapping study of mobile application testing techniques", *Journal of Systems and Software*, vol. 117, pp. 334-356, 2016. DOI: https://doi.org/10.1016/j.jss.2016.03.065

[2] S. Yin, J. Sheng, T. Wang and H. Xu, "Analysis on Mobile Payment Security and Its Defense Strategy", in International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 941-946, 2018.

[3] B. Burg, R. Bailey, A. J. Ko, and M. D. Ernst, "Interactive record/replay for web application debugging", in Proceedings of the 26th annual ACM symposium on User interface software and technology, pp. 473-484, 2013.

[4] ISTQB: International Software Testing Qualifications Board. Available: www.istqb.org

[5] ISEB: Information Systems Examinations Board of British Computer Society. Available: https://certifications.bcs.org/

[6] GTB: German Testing Board. Available: www.german-testing-board.info

[7] L. Strazdiņa, V. Arnicane, G. Arnicans, J. Bičevskis, J. Borzovs, and I. Kuļešovs, "What Software Test Approaches, Methods, and Techniques are Actually Used in Software Industry?", in Doctoral Consortium/Forum@DB&IS, 2018.

DOI: https://doi.org/10.21501/21454086.2983

[8] S. Dalal, K. Solanki, and S, "Challenges of regression testing: a pragmatic perspective". *International Journal of Advanced Research in Computer Science*, vol. 9, no. 1, pp. 499-503, 2018. DOI: http://dx.doi.org/10.26483/ijarcs.v9i1.5424

[9] D. E. Soto Durán, A. X. Reyes Gamboa, y J. Jiménez Builes, "Aplicación de la Gestión de Conocimiento al proceso de pruebas de software", *Ingenierías USBMed*, vol. 8, no. 2, pp. 6-13, 2017. DOI: https://doi.org/10.21500/20275846.2836

[10] M. Steiner, M. Blaschke, M. Philipp, and T. Schweigert, "Make test process assessment similar to software process assessment—the Test SPICE approach". *Journal of Software: Evolution and Process,* vol. 24, no. 5, pp 471-480, 2010. DOI: https://doi.org/10.1002/smr.507

[11] I. Acosta, E. Nieto, y C. Barahona, "Metodología para la evaluación de calidad de los productos software de la Universidad de Cundinamarca". *ENGI Revista Electrónica de la Facultad de Ingeniería*, vol. 3, no. 2, pp. 13-16.

[12] S. Ali, and T. Yue, "Formalizing the ISO/IEC/IEEE 29119 Software Testing Standard". In 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), Ottawa, Canadá, 2015, pp. 396-405.

[13] A. Dávila, C. García, and S. Cóndor, "Análisis exploratorio en la adopción de prácticas de pruebas de software de la ISO/IEC 29119-2 en organizaciones de Lima, Perú", *RISTI-Revista Ibérica de Sistemas e Tecnologías de Información*, no. 21, pp. 1-17, 2017. DOI: http://dx.doi.org/10.17013/risti.21.1-17.

[14] ISO 25000. ISO/IEC 25010 [Online]. Available: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[15] J. G. Enríquez, y S. I. Casas, "Usabilidad en aplicaciones móviles", *Informes Científicos-Técnicos UNPA,* vol. 5, no. 2, pp. 25-47, 2013. DOI: http://dx.doi.org/10.22305/ict-unpa.v5i2.71

[16] R. Harrison, D. Flood, and D. J. Duce, "Usability of mobile applications: literature review and rationale for a new usability model" *Journal of Interaction Science,* vol. 1, no. 1, pp. 1-16, 2013. DOI: https://doi.org/10.1186/2194-0827-1-1

[17] M. Medeiros Eler, J. M. Rojas, Y. Ge, and G. Fraser, "Automated Accessibility Testing of Mobile Apps", in Software Testing, Verification and Validation (ICST), 2018 IEEE 11th International Conference on, 2018, pp. 116-126.

[18] D. Amalfitano, N. Amatucci, A. M. Memon, P. Tramontana, and A. R. Fasolino. "A general framework for comparing automatic testing techniques of Android mobile apps", *Journal of Systems and Software*, vol. 125, pp. 322-343, 2017. DOI: https://doi.org/10.1016/j.jss.2016.12.017

[19] L. Deng, J. Offutt, P. Ammann, and N. Mirzaei, "Mutation operators for testing Android apps", *Information and Software Technology*, vol. 81, pp. 154-168, 2017. DOI: https://doi.org/10.1016/j.infsof.2016.04.012

[20] H. Muccini, A. Di Francesco, and P. Esposito, "Software testing of mobile applications: challenges and future research directions". In Proceedings of the 2012 7th International Workshop on Automation of Software Test (AST), 2012, pp. 29-35.

[21] InfoQ. (2018, July 01). 2018 State of Testing Report. Available: https://www.infoq.com/articles/state-of-testing-report-2018

[22] P. Bourque, and R. E. Fairley (Eds.). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0.* IEEE Computer Society Press.

DOI: https://doi.org/10.21501/21454086.2983

[23] B. Bruegge, A. H. Dutoit. *Ingeniería de software orientado a objetos*. México: Pearson educación, 2002.

[24] H. A. Parada Gélvez. *Contribución a la gestión de los procesos de pruebas de software y servicios*. Madrid: Universidad Politécnica de Madrid, 2010.

[25] Scott Ambler. The "Broken Iron Triangle": Software Development Anti-Pattern. Available: http://www.ambysoft.com/essays/brokenTriangle.html

[26] J. L. Aristegui, "Los casos de prueba en la prueba de software", *Lámpsakos,* no. 3, pp. 27-34, 2010. DOI: https://doi.org/10.21501/21454086.785

[27] A. Granollers i Saltiveri, J. Lorés Vidal, y J. J. Cañas Delgado. *Diseño de sistemas interactivos centrados en el usuario*. Barcelona: UOC, 2011.

[28] R. Pressman. *Ingeniería del software: un enfoque práctico*. México D.F.: McGraw-Hill, 2006.

[29] E. Torres, E. Sevillano, y J. Lodos. "Herramienta para la ejecución de componentes de pruebas", en: Memorias de la XIV Convención y Expo Internacional de Informática. V Taller Internacional de Calidad en las Tecnologías de la Información y las Comunicaciones. La Habana: Universidad de las Ciencias Informáticas, 2011, pp. 145-156.

[30] P. Macharla, "Working with Appium," in Android Continuous Integration: Build-Deploy-Test Automation for Android Mobile Apps, Berkeley, CA: Apress, 2017, pp. 95–115.