El antiguo arte del testado

Emilio Molina
Interfaz Humana de Protocolo en Lollipop Robot
QA & Testing

En la trastienda de las disciplinas relacionadas con los videojuegos que han sido más malinterpretadas a lo largo de su breve historia nos encontramos con el apartado del testado.

Surgido en una convergencia entre el desarrollo de otras aplicaciones informáticas y la necesidad propia de unas estructuras cada vez más complejas en videojuegos (tanto a nivel de la base de código interno de funcionamiento como de jugabilidad del propio juego), el papel de probador (*tester*) de videojuegos pasó de recaer del propio programador (quien también solía ser el artista y el músico) a grandes empresas hiperespecializadas en el área.

No son pocas las veces que en la industria hemos escuchado la frase —usualmente de gente joven—: «quiero ser *tester*; se pasan todo el día jugando y les pagan para ello». Por hacer un símil rápido de lo distintas que son las expectativas de la realidad, alguno de mis conocidos propuso el ejemplo del comprobador de calidad de preservativos; cuando quizá alguno pueda pensar que se pasa la jornada de orgía en orgía, algo más acercado a la realidad es la de un individuo inflando con una máquina específica una serie inacabable de muestras aleatoriamente escogidas.

Lo cierto es que desconozco en qué consiste el testado industrial de preservativos, pero sí puedo contar de primera mano en qué consiste hoy en día la certificación de calidad (Quality Assurance o QA) y testado de videojuegos.

Lo primero que hay que aclarar para empezar a desmontar el mito es que el trabajo de *tester* es, probablemente, uno de los más duros del ámbito del desarrollo. Imaginemos que el estudio de desarrollo está trabajando en un proyecto encargado por terceros. No es el último *Call of Duty*, sino, inventándome un ejemplo, *Las aventuras subacuáticas de Barbie*, que debe salir a la venta en un plazo de unos tres meses a nivel mundial para una de las principales consolas del mercado. Veamos todo lo que implica.

DOI: http://dx.doi.org/10.6035/2174-0992.2015.9.13

La empresa del *tester* debe contar con un dispositivo especial para la prueba de dicha consola conocido como *TestKit* (una versión de la consola modificada para poder obtener información adicional en caso de problemas). En ocasiones no existe esta especialización y se usa el *DevKit* (la versión modificada para el propio desarrollo del videojuego). Tener el dispositivo específico requiere, a su vez, haber cumplimentado una serie de tediosas burocracias que no describiré aquí. En raras ocasiones el dispositivo de desarrollo y testado es la propia consola final, como era el caso de los juegos de la sección Xbox Live Indie Games de la Xbox 360.

El jefe de *testing* debe estar coordinado con la producción del videojuego para demandar y recibir el material que necesitará: una versión del juego y un documento donde se especifica qué comportamiento se espera del juego.

Cuando me refiero a la *versión del juego*, encontramos dos posibilidades: que se nos pida el testado continuo del videojuego (es más propio de otras aplicaciones informáticas llevadas a cabo mediante metodologías ágiles de desarrollo como SCRUM, pero cada vez es más común en videojuegos por la mayor facilidad de detectar y corregir graves problemas en etapas lo más iniciales posible) o que se nos entregue una serie de versiones basadas en los hitos (*milestones*) clásicos del desarrollo, que son la versión alfa (alrededor del 75% de la programación terminada, 50% de los gráficos terminados), beta (100% de la programación terminada, 75% de los gráficos terminados) o candidata a publicación (*release candidate*, 100% terminada). La nomenclatura y porcentajes varían mucho entre empresas pero, a efectos prácticos, definen distintos estadios dentro de las últimas etapas del desarrollo en las que va siendo importante que reciban un repaso exhaustivo para comprobar no sólo que no contienen problemas sino que además se ajustan a las especificaciones de lo que cada consola exige de los juegos que serán publicados en ellas (conocido como TRC, de Technical Certification Requirements), consistentes en una normativa de calidad (usualmente en inglés) en forma de lista de reglas (también llamados por ello *checklists* o *lotchecks*) sobre qué tipo de mensajes deben aparecer en pantalla o cuándo (por ejemplo, cuando a un mando le falta batería o se le acaba), mensajes de errores de red, imaginería de la marca en pantalla (logos, mandos, símbolos de algunos tipos...).

Los TRC vienen dados por la plataforma junto con el *DevKit* o *TestKit*, pero el documento de especificaciones del propio juego es otra cosa. Con esto me refiero al comportamiento de todo el juego, lo que implica secciones en las que nadie suele pensar nunca: el flujo de menús y el color de las secciones resaltadas de menú al ser seleccionadas, el sonido que debe emitir al ser seleccionada, las categorías y personas que deben aparecer en los títulos de crédito, los colores de dichas secciones (¿verdad que ya no suena tan atractivo?). Este documento pocas veces está completo (en ocasiones ni existe) y, de existir, menos aún está actualizado. Por esta razón, suele ser importante que el *tester* pueda estar en contacto directo con el diseñador del juego o con los propios desarrolladores,

DOI: http://dx.doi.org/10.6035/2174-0992.2015.9.13

para poder resolver ágilmente las dudas que puedan surgir, como dilucidar si un evento que no encaje con lo esperado es realmente un problema del juego o un fallo en el documento de diseño.

En ocasiones (las más raras) se llega a contar con un documento de casos de test, consistente en un listado (al estilo TRC) de las pruebas que el jefe de test ha confeccionado junto con el diseñador del juego o productor ejecutivo, y que se centran en probar las partes más representativas del juego o las que han sido susceptibles de fallos en pruebas anteriores. En otras ocasiones, sólo se pide un *smoke test*, un test lo más rápido posible de toda la funcionalidad, para asegurarse de que no hay fallos obvios que hayan de repararse antes de proceder a pruebas más pormenorizadas: no tiene sentido buscarle arañazos al chasis de un coche si tiene el motor gripado o la batería descargada.

Supongamos que tenemos dicho documento para nuestro juego de Barbie. Recordemos que se lanza a nivel mundial, por lo que una de la larga lista de pruebas consistirá en comprobar que cada posible texto del juego aparece correctamente visualizado en cada posible idioma, sin salirse de su botón o caja de texto (¿conocéis los memes sobre el idioma alemán?). Si apareciera un fallo en un texto de, por ejemplo, la última fase, lo reportaríamos (lo usual es utilizar una herramienta web específica llamada *bugtracker* para la gestión de incidencias) y, cuando se nos notificara que se ha corregido el fallo, deberemos volver a comprobarlo todo para asegurarnos tanto de que el fallo se ha corregido realmente, como de que no se han introducido nuevos problemas en el proceso. El origen de los *cheats* de los juegos suele radicar en facilidades que los programadores añaden para hacer más ágil su testado en este tipo de situaciones, pero que luego olvidan eliminar (o mantienen como guiño).

En el proceso de probar un videojuego (o una aplicación), lo más importante es mantener una actitud escéptica: normalmente, un desarrollador suele probar (inconscientemente) el juego de forma que a él le funciona; la labor del *tester* es desvelar los fallos, usando el sistema de formas que el desarrollador no hubiera previsto. Por ejemplo, en un juego de coches, jugándolo completamente marcha atrás, o chocando constantemente contra el decorado, o pulsando tres botones a la vez cuando el sistema sólo espera un único *click*, o pausando el juego en situaciones extremas. Un *tester* no comprueba si un juego funciona; debe intentar probar que no funciona.

Como reto para el lector dejo el clásico problema: «Hay cuatro cartas sobre una mesa: por un lado, las cartas tienen un número; por el otro, una letra. Identifique a qué cartas debe dar la vuelta para comprobar la regla que reza: "si hay una vocal en una cara de la carta, hay un número par por el otro lado". Las caras visibles de las cartas son: A 2 7 F».

Una de las pruebas que planteamos en la entrevista de trabajo es una simple pantalla de menú de introducción de usuario y contraseña con un botón para enviar, y pedimos la lista de posibles problemas que se les ocurra que pudieran

DOI: http://dx.doi.org/10.6035/2174-0992.2015.9.13

surgir. Normalmente detectan entre diez y veinte. En mi lista interna aparecen más de ochenta, y de vez en cuando aún se añade alguno que se me había escapado. Podéis haceros una idea de qué supone el *testing* de un juego online *multiplayer*.

En la industria suele hablarse de una clásica confrontación entre el programador y el *tester* pero, al menos desde nuestra experiencia, es sólo un mito, probablemente basado en casos de programadores que se tomaran los fallos como ataques personales, o *testers* que fueron incapaces de aportar la retroalimentación del problema de forma eficaz. Es fundamental una comunicación fluida entre ambas áreas para que el proyecto avance lo más rápidamente posible, y para ello se trabaja partiendo de la base de que todas las partes implicadas hablen con un lenguaje común. En un mundo ideal, el *tester* debe ser capaz de reportar un fallo con la información mínima necesaria para que el programador pueda detectar el problema rápidamente, con toda la ayuda posible para replicarlo e incluso tener alguna pista sobre su origen. En algunas compañías incluso se graban vídeos de las sesiones de test, o se comparte remotamente la sesión de usuario del *tester* (si está ejecutando el juego a través de un programa de depurado, que permite obtener información extra a costa de un menor rendimiento en el juego) para que el programador pueda averiguar por su cuenta cuántos datos adicionales pudiera necesitar.

También se suele oír que el *testing* es un trabajo de monos, aunque en realidad un buen *tester* ha de tener presumiblemente conocimientos multidisciplinares y/o formar parte de un equipo que pueda estar pendiente de problemas de arte, conectividad, IA, idiomas, seguridad, eficiencia (detectando cuellos de botella), sonido…

En resumen: si quieres ser *tester* tienes que tener el inglés muy fluido, saber trabajar en equipo y tener mucha tolerancia a trabajos repetitivos que a la vez requieran una alta capacidad de concentración, probablemente a cambio de un sueldo muy modesto y seguramente a costa de aborrecer el juego que vas a testar. Si quieres jugar a lo que quieras y que te paguen, no quieres ser *tester*: quieres ser crítico independiente de videojuegos.

Referencia de este artículo

Molina, Emilio (2015). El antiguo arte del testado. En: *adComunica. Revista Científica de Estrategias, Tendencias e Innovación en Comunicación*, nº9. Castellón: Asociación para el Desarrollo de la Comunicación adComunica, Universidad Complutense de Madrid y Universitat Jaume I, 193-196. DOI: http://dx.doi.org/10.6035/2174-0992.2015.9.13.

DOI: http://dx.doi.org/10.6035/2174-0992.2015.9.13