

Usabilidad y Análisis de Juegos

Guillermo Jiménez Díaz (gjimenez@ucm.es)
Pilar Sancho Thomas (psancho@ucm.es)

Curso 2023-24

Quality Assurance

Recordemos: ¿Cuáles pueden ser las claves para que un videojuego sea un éxito?

- Que sea divertido
- **Que funcione correctamente y no tenga errores**

Algunos de los responsables de que esto se cumpla pertenecen a los departamentos de QA o de *Production testing (DevQA)*. Se encargarán de verificar el correcto funcionamiento de nuestro juego, pero no para dar calidad a malos diseños o a conceptos pobres.

Quality Assurance (QA) es el proceso de pruebas que se usa para comprobar que el videojuego funciona como ha de funcionar y que no hay errores que detengan la ejecución del juego (ni, preferiblemente, ningún otro error que afecte a la experiencia de juego del jugador). A diferencia de muchas de las pruebas del *Production testing*, estas pruebas son realizadas por personas: los *probadores*.

Fases de pruebas

Tradicionalmente (Figura 5.1) era una tarea que se hacía al final de la producción (cuando el juego estaba en *alfa*, es decir, en una versión con todas las mecánicas completamente implementadas pero, posiblemente, con assets aún no definitivos).

En realidad, las pruebas se planifican durante la fase de preproducción y se desarrollan durante todo el ciclo de producción (Figura 5.2). La mayoría de los bugs deberían ser detectados durante el desarrollo, a medida que se crean assets y se implementan funcionalidades. El departamento de QA trabajará a lo largo de la fase de desarrollo con las builds de juego que se generen de manera periódica y que se pueden considerar como una versión *alfa incompleta* sobre la que se irán realizando pruebas.

El testeo completo de un videojuego se suele producir en varias fases:

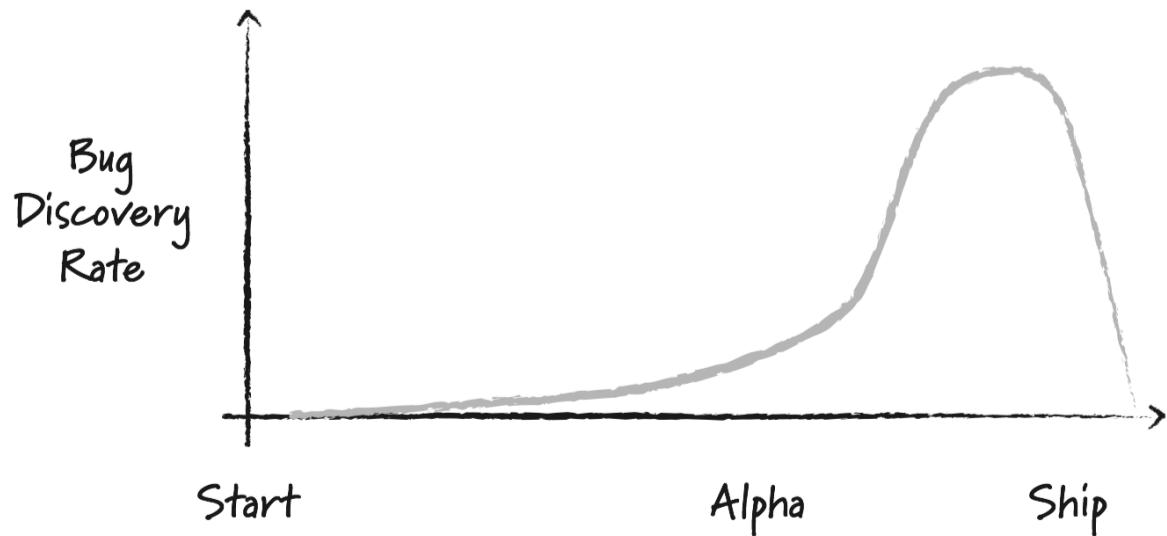


Figura 5.1: QA en proyectos tradicionales (Fuente: *Agile Game Development with Scrum*)

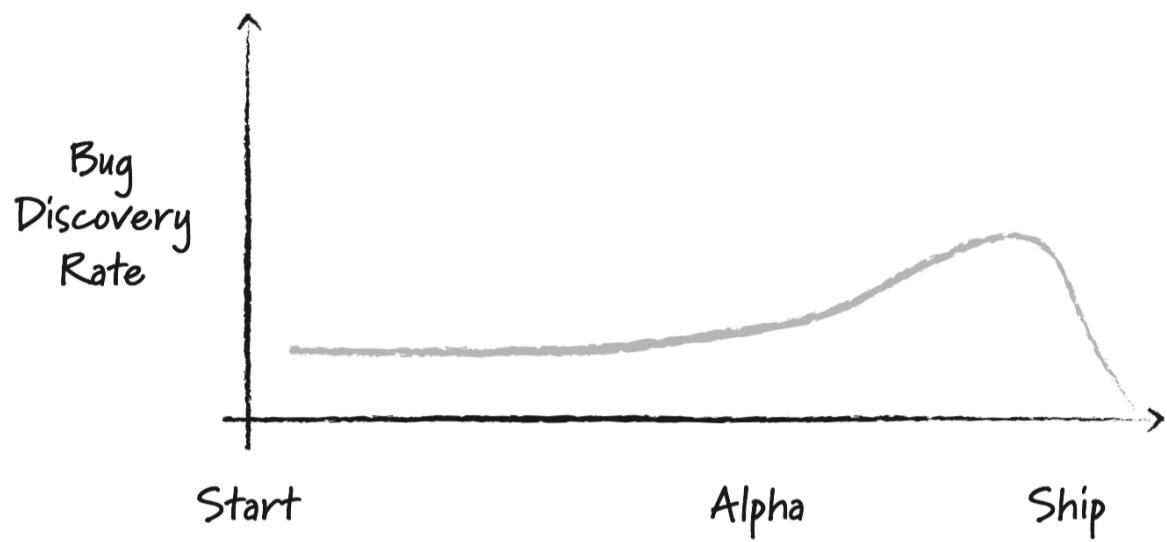


Figura 5.2: QA en proyectos ágiles (Fuente: *Agile Game Development with Scrum*)

- **Vertical slice:** Versión jugable con el core del juego, en entornos que no tienen por qué ser los definitivos y con arte placeholder
- **Alpha:** build completa (se juega de principio a fin) con posibles assets provisionales y funcionalidades quasi-completas, a falta de pulirlas. El juego funciona correctamente en la plataforma destino en modo *debug*. En realidad, como hemos dicho antes, suele no ser una única prueba sino que se prueban varias “pre-alpha” a lo largo de toda la etapa de desarrollo.
- **Beta:** build de juego completo sin errores críticos y en el que se incorporan los assets definitivos. No hay más desarrollo salvo para corregir bugs y errores en los assets.
- **Gold o Code Release Candidate:** es una versión completa del juego, preparada para ser producida. Es la versión para enviar a **compliance**, que no solo garantiza que el videojuego funciona adecuadamente sino que, además, se ajusta adecuadamente a los estándares impuestos por la plataforma en la que se va a ejecutar el juego.

Departamento de QA

El departamento de QA es responsable de **escribir los planes de prueba** y de **validar el juego siguiendo dichos planes**. El juego ha de ser validado en todas sus áreas y, dependiendo del tamaño, puede requerir una gran cantidad de trabajo, ya que puede implicar repetir las mismas pruebas en distintas localizaciones, plataformas y configuraciones.

Roles

El departamento de QA se suele componer de los siguientes roles:

- **Lead QA tester:** Es el responsable de planificar y estimar el coste de las pruebas, diseñar el plan de pruebas y de gestionar al equipo de QA. Controla y limita la “explosión combinatoria” de posibilidades durante el diseño (imaginemos probar todas las combinaciones posibles de un personaje que tiene 10 características diferentes para configurar, cada una de ellas con 10 valores diferentes). Participa en las decisiones relacionadas con dar por cerrado el juego (cuál es la que se considerará como RC o *release candidate*).
- **Tester o probador:** Es el responsable de probar el juego *siguiendo el plan de pruebas*. Busca defectos en el juego y los reporta. Así mismo, también es responsable de revisar que dichos errores han sido resueltos por el equipo de desarrollo. Los *core*

testers son aquellos que trabajan en exclusiva para un proyecto. Sin embargo, algunos probadores pueden trabajar simultáneamente en distintos videojuegos (por ejemplo, los probadores especializados en **certificación**).

Cualidades de un buen probador

Son los responsables finales de la realización de los tests. Suele ser un buen medio de entrada a la industria de los videojuegos pero hay que eliminar la idea errónea de que “el probador se pasa todo el día jugando a juegos”. Tal y como veremos, es una tarea repetitiva y planificada, que no suele consistir en tan solo jugar a un videojuego.

Algunas de las características más deseables que se esperan de un probador son:

- Buena capacidad de comunicación, para discutir los errores encontrados con el resto de miembros del equipo.
- Buena capacidad de redacción, para detallar correctamente los errores encontrados.
- Capacidad de análisis para ser capaz de reproducir errores (y detallar cómo reproducirlos).
- Atención a los detalles, para ser capaz de encontrar el número máximo de errores.
- En la mayoría de las empresas se requiere un buen nivel de inglés (sobre todo si son internacionales).

Planes de pruebas

Los planes de pruebas pueden estar en diferentes versiones pero, en todo caso, son detallados y exhaustivos de cara a los testers. A modo de ejemplo, en la Figura 5.3 aparece un plan de pruebas sencillo de *Pass-Fail*. El probador, a medida que avanza por el nivel, ha de ir chequeando cada uno de los casos de prueba que se enuncian en el plan. En caso de fallo, es conveniente añadir comentarios que expliquen por qué no ha pasado el test. Si no son capaces de testearlos se suelen utilizar marcas específicas como CNT (*Cannot test*)

Nivel 1	Requisitos	Pass/Fail	Comentarios
Intro misión	La cutscene se ejecuta correctamente	PASS	
	El audio suena correctamente	N/A	
	La cutscene puede ser saltada	CNT	
Objetos interactivos	Las luces se pueden encender y apagar	FAIL	
	La máquina expendedora se activa al interactuar con ella	FAIL	

Figura 5.3: Ejemplo de un plan de pruebas *Pass-Fail*

Cada caso de prueba se puede describir de la siguiente forma:

- Descripción: explicación de lo que se está probando.
- Pasos: Acciones, en orden cronológico, que se han de realizar para alcanzar el estado que pretende ser probado.
- Resultado esperado: resultado que se espera tras realizar las acciones, supuesto que el videojuego funciona correctamente.

Un ejemplo de guión más complicado (y más cercano a la realidad) sería la revisión del menú de selección de personajes de Knights of the Old Republic para Xbox, extraído de *Game Testing All in One*:

1. Select New Game from the Main Menu.
 - Check that the Male Scoundrel picture and title are highlighted (Figure 5.4).
 - Check that the scoundrel character description is displayed correctly.
2. Scroll to the left using the D-Pad.
 - Check that the Female Scoundrel picture and title are highlighted.
 - Check that the scoundrel character description is unchanged.
3. Scroll to the right using the D-Pad.
 - Check that the Male Scoundrel picture and title are highlighted.
 - Check that the scoundrel character description is unchanged.
4. Scroll to the right using the LEFT analog stick.
 - Check that the Male Scout picture and title are highlighted.
 - Check that the scout character description is displayed correctly.
5. Scroll to the left using the LEFT analog stick.
 - Check that the Male Scoundrel picture and title are highlighted.
 - Check that the scoundrel character description is displayed correctly.
6. Scroll to the right using the RIGHT analog stick.
 - Check that the Male Scoundrel picture and title are unchanged.
 - Check that the scoundrel character description is unchanged.
7. Press the X button.
 - Check that the Male Scoundrel picture and title are unchanged.
 - Check that the scoundrel character description is unchanged.
8. Press the Y button.
 - Check that the Male Scoundrel picture and title are unchanged.
 - Check that the scoundrel character description is unchanged.
9. Press the B button.
 - Check that the Main Menu screen is displayed with “New Game” highlighted.



Figura 5.4: Menú de selección de personajes de Knights of the Old Republic para Xbox

Severidad de los errores

Aunque cada proyecto puede definir sus propias categorías, esta es una de las categorías de severidad de errores más frecuentemente usada.

- **Crash bug:** el juego se congela o se cierra. Son extremadamente graves y son los primeros en arreglar.
- **Critical bug:** el juego avanza, pero es un error en alguna funcionalidad o característica importante. Ej: Faltan las texturas de un nivel.
- **Minor bug:** error notable por el jugador pero que no afectan a la experiencia del juego. Ej: errores ortográficos en textos del juego.
- **Feature request:** no es un error, sino funcionalidad que estaría bien que estuviera, pero que no estaba en el documento de diseño original. Ej: que el HUD se pueda activar/desactivar (si no estaba planificado en el GDD original).

Proceso de pruebas

Antes de comenzar el proceso de pruebas, el equipo de QA ha de definir la herramienta de gestión de errores a utilizar, así como el *pipeline* o estados por el que van a pasar los errores encontrados durante el proceso y las categorías de los errores.

Las herramientas o bases de datos de gestión de errores han de estar **centralizadas** y accesibles para todos los miembros implicados en ella. Algunas de las herramientas de gestión de errores (o *bugtrackers*) son:

- Mantis BT
- Bugzilla
- Los issues de Github
- Jira
- ...

En líneas generales, el proceso de pruebas consiste en:

1. El probador descubre un bug en el juego siguiendo el plan de pruebas.
2. El probador comprueba que el bug no ha sido introducido ya en la herramienta (o base de datos) de gestión de errores.
3. El probador describe el bug y lo reporta usando la herramienta de gestión de errores.
4. Dependiendo de la organización, el Lead revisa que el error no esté duplicado y comprueba que está bien cumplimentado.

5. Se solicita a los desarrolladores que lo resuelvan. Una vez resuelto, se marca como tal en la herramienta de gestión de errores.
6. El Lead lo asigna a un probador (o el propio probador que lo reportó se lo autoasigna) para volver a probarlo y comprobar que ha sido reparado.

Los estados por los que generalmente pasa un error son:

- Abierto (*found*): se ha encontrado y registrado un bug.
- Resuelto (*fixed*): el error ha sido analizado y, supuestamente, resuelto.
- Verificado (*verified*): la corrección del error ha sido verificada.
- Cerrado (*closed*): tras la verificación, el lead de QA los cierra (*housekeeping*).

También se pueden definir otros posibles estados como:

- “Need more info”: el desarrollador considera que el error y cómo reproducirlo no está suficientemente especificado. Esto obliga al probador a dar más detalles del error. Este estado es especialmente dramático si el probador y el desarrollador trabajan en diferentes usos horarios (típico en grandes empresas como EA) ya que se pierde mucho tiempo desde que el error se encuentra hasta que se arregla.
- “Reabierto” (si vuelve a surgir el bug)
- “Duplicado”: otro probador ya introdujo este error.
- “Not a bug” o “As design”: Diseño confirma que es el comportamiento esperado y no un error.
- “Will not fix”: errores conocidos que no pueden ser corregidos por parte del equipo de desarrollo

Descripción de errores

Como van a ser muchos los probadores que van a estar introduciendo errores en la base de datos es fundamental especificar la forma en la que se van a describir los errores. La descripción, en todo caso, ha de ser *exhaustiva* y formal, dando el máximo de detalles para reproducir el error.

Una empresa puede usar una plantilla de descripción de errores encontrados. Generalmente, dicha plantilla incluye campos como los que detallamos a continuación:

- Quién lo ha reportado (generalmente se recoge automáticamente en la herramienta).
- Versión de la build o changelist donde se produce.
- Categoría del error. Ej: arte, diseño, programación...
- Componente: subcategoría del error. Ej: IA, interfaz, física, carga y guardado, texturas, modelos...

- Resumen: “Asunto” o descripción en una frase del error. Suele ser conveniente usar un convenio para poder buscarlos fácilmente. Por ejemplo, si empiezan por “M01 - Arte: ...” sabemos que son errores de arte en la misión 1.
- Descripción: explicación del error. Es muy importante destacar cuál ha sido el comportamiento *encontrado* frente a qué comportamiento era el *esperado*.
- Severidad: De acuerdo a las categorías vistas en la sección .
- Prioridad: para ordenar los bugs con la misma severidad.
- Pasos para reproducirlo (si es reproducible).
- Capturas de pantalla o de vídeo.
- Ficheros de log (si los desarrolladores han creado un ejecutable que genera este tipo de ficheros).

Betatesting

Las pruebas de betatesting se realizan con una versión del juego *virtualmente* terminada, con assets definitivos y con los errores críticos corregidos. Existen dos tipos de pruebas de betatesting:

- Cerradas o internas: es una prueba privada realizada por personal interno de la compañía. Sirven para pulir el gameplay y las mecánicas.
- Abiertas o externas: es una prueba pública, que usa probadores externos a la empresa. De esta forma, la empresa recoge feedback de una gran cantidad de probadores simultáneamente (que seguramente no tiene por qué tener en plantilla). El juego se juega por gente real en entornos reales, de modo que da una idea de cómo se usa el juego en realidad y cuáles son los problemas encontrados más comunes. En caso de juegos multijugador, sirven también como pruebas de estrés *reales* para los servidores (que deberían estar probados con anterioridad).

En general suelen ser pruebas más libres que las del departamento de QA pero en muchas ocasiones también puede consistir en realizar tareas específicas. Es común que la fase de betatesting (sobre todo si es externa) se confunda (y se mezcle) con el playtesting, de modo que se pida feedback a los usuarios del producto final mediante surveys y analíticas.

Non-Disclosure Agreements

En general, los betatesters (y otros miembros del equipo de desarrollo) han de firmar un documento de confidencialidad conocido como *non-disclosure agreement o (NDA)*. Es un contrato de negocio que protege la confidencialidad de la información del videojuego

que los testers están probando. Básicamente, el probador se compromete a no revelar ninguna información relacionada con el juego que han probado: no hablar de ello en redes sociales, no tomar fotos o vídeos del producto ni enseñar documentos relacionados con el mismo. En general se obliga a no hablar sobre ninguna de las tareas realizadas (ya sean o no de testeo) hasta que no se haya concluido el desarrollo del mismo.

Es muy común poner marcas de agua en las versiones beta protegidas por un NDA, con el fin de saber quién ha compartido el juego ilegalmente. Actualmente, esto también se está haciendo en las versiones usadas por el departamento de QA dentro de la propia empresa, incorporando en la marca de agua un identificador del probador, para evitar fugas de información.

Pruebas de Compatibilidad

Es el proceso por el cual se prueba el juego en múltiples configuraciones de hardware, con el objetivo de asegurar que el videojuego es compatible con el mayor número de dispositivos y periféricos que hay en el mercado. Implica probar:

- Cómo se comporta el juego (velocidad, lag, flickering, crashes...)
- Si se visualiza correctamente en distintas pantallas (tamaño de pantalla, densidad de píxeles, colores, drivers...)
- Impacto del juego, sobre todo en móviles (uso de batería, red y datos...)

Los tests de compatibilidad se realizan para:

- PC: distintos sistemas operativos, con distintos periféricos (monitores, tarjetas gráficas, mandos...)
- Consolas
- Móviles
- Navegadores
- Red: IPv4 vs. IPv6, qué ocurre si la conexión falla o se cae.
- Compatibilidad entre versiones del videojuego

Debido al coste de estas pruebas (imaginemos el número de móviles que deberíamos tener para probar un juego de móvil en un número aceptable de dispositivos, **teniendo en cuenta que en 2015 se contabilizaron más de 24000 modelos distintos** de dispositivos Android) es muy común externalizarlas a empresas que disponen de “granjas” de dispositivos sobre los que se ejecutan baterías de casos de pruebas automáticas.



Figura 5.5: Granjas de móviles en Meta (Fuente: [The mobile device lab at the Prineville data center, Engineering at Meta, 2016](#))

Soaking tests

Son tests en los que los probadores prácticamente no interactúan con el juego. Consisten en dejar un juego funcionando durante un largo periodo de tiempo (pueden ser días) en un equipo (consola, móvil...) para comprobar que el videojuego funciona correctamente. Son necesarios ya que algunos problemas relacionados con *leaks de memoria* o redondeos solo aparecen tras una ejecución muy larga, por lo que son indetectables por los test normales. Algunos de estos tests son fundamentales para poder pasar las pruebas de certificación, de las que hablaremos a continuación.

Certificación o *Compliance*

La *code release candidate (CRC)* se considera aún candidata ya que antes de ser lanzada ha de pasar las pruebas de calidad impuestas por la empresa de la plataforma para la que va a ser desarrollada. Cada plataforma puede predefinir una serie de requisitos técnicos que el juego ha de cumplir. Las pruebas que sirven para verificar que se cumplen estos requisitos técnicos se conocen como **pruebas de certificación o compliance**. No solo las CRC pasan las pruebas de certificación sino que también los parches y las DLCs (*DownLoadable Content*).

Para realizar estas pruebas las empresas suelen definir una lista con todos los requisitos técnicos que ha de pasar el videojuego para una plataforma específica antes de ser

lanzado en dicha plataforma. Por ejemplo:

- Sony define los TRCs (del inglés, *Technical Requirements Checklist*).
- Microsoft tiene los Xbox Requirements.
- Nintendo define las guidelines (*lotcheck*).
- Steam y otras tiendas online también imponen sus propios requisitos.

Estos documentos están bajo NDA (*No disclosure agreement*) por lo que no se suelen conocer. Algunos son muy técnicos y no se consideran como pruebas del juego como tal (por ejemplo, cuánto tiempo puede permanecer una pantalla en negro, qué tipos de avisos hay que poner durante el guardado de una partida, si los datos se han corrompido o si se ha de formatear una unidad, comprobaciones del tipo “el videojuego ha de pausarse cuando se desenchufa el mando”...).

Si no se pasan los test impuestos por la plataforma entonces se pone en riesgo el juego. Generalmente, estas certificaciones cuestan dinero y tiempo (tanto de desarrollo como del momento en el que van a volver a probar nuestro juego si hacemos una *resubmission*). Por tanto, suele ser preferible designar a una persona experta en ellos que los conozca en profundidad y los compruebe. Así mismo, existen empresas encargadas exclusivamente a realizar este tipo de pruebas. Si se encuentran, todos los errores relacionados con *compliance* son registrados en el *bugtracker* con una severidad crítica.

Nota final

Esta obra está bajo una Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional

