PROCESOS DE DESARROLLO PARA VIDEOJUEGOS

Gerardo Abraham Morales Urrutia, Claudia Esther Nava López, Luis Felipe Fernández Martínez, y Mirsha Aarón Rey Corral

Instituto de Ingeniería y Tecnología. Universidad Autónoma de Ciudad Juárez

Resumen

Los procesos en el desarrollo de software son importantes, imponen consistencia y estructura sobre el conjunto de actividades necesarias en un proyecto. Este trabajo trata sobre el uso de procesos para el desarrollo de videojuegos. Se analizaron distintos procesos viables para el desarrollo de videojuegos y se compararon las ventajas del Scrum Framework sobre el Waterfall Process. Asimismo, se hace una propuesta de un proceso de desarrollo para videojuegos orientado a desarrolladores independientes, denominado Huddle. Esta propuesta incluye detalles del proceso, diagramas e información de las plantillas y herramientas que se crearon para apoyarlo.

Palabras Clave: procesos de desarrollo; software; diseño; videojuegos.

1. Introducción

Un videojuego es un medio de entretenimiento que involucra a un usuario, denominado jugador, en una interacción constante entre una interfaz y un dispositivo de video. Los videojuegos recrean entornos y situaciones virtuales en los que el jugador puede controlar uno o varios personajes para alcanzar objetivos por medio de determinadas reglas.

La interacción se lleva a cabo mediante dispositivos de salida de video: monitor de PC, televisión, proyector, etc., sin embargo, también intervienen dispositivos como un teclado, mouse, *joystick*, *gamepad* y dispositivos que detectan movimiento.

El videojuego como medio de entretenimiento ha evolucionado de manera increíble en muy poco tiempo, se han convertido en una industria multimillonaria casi a la par con la industria del cine (Otter, 2008). Los videojuegos eran considerados juguetes para niños y desarrollados por uno o dos programadores novatos en los sótanos de sus casas. Actualmente las empresas, editoriales y grupos de desarrolladores independientes involucran a profesionales en campos como diseño, informática, música, marketing e incluso leyes entre otros, para lograr publicar un solo videojuego.

Los videojuegos arte, ciencia y tecnología; involucran una plétora de habilidades y conocimientos en distintas disciplinas, desde ciencias formales hasta ciencias sociales que van más allá del típico proyecto de software e implican al mismo tiempo la creatividad y la imaginación. Un videojuego combina elementos de narración, música, animación y deporte. El código es como una partitura musical la cual es tocada por una computadora y los juegos se vuelven a veces tan competitivos que se juegan como deporte.

En la era que (Dille & Zuur, 2007) definieron como primitiva, cualquier persona podía aprender a programar un juego de computadora si se disponía de una Computadora Personal (PC) y conocimientos del lenguaje Ensamblador. Producir un juego no necesitaba más de dos personas y/o más de 3 meses. La competencia en las eras subsecuentes, la postura de la sociedad y el lugar en el que se encuentra la industria actualmente, implica equipos de desarrollo de 25 o más miembros para lograr publicar un solo videojuego comercial y los proyectos pueden durar más de 3 años y un presupuesto de millones de dólares. Un videojuego no sólo es un producto artístico, debe de pasar por varias fases desde que es concebido hasta que es olvidado, es decir, que como todo software, tiene un ciclo de vida. El ciclo de vida da la pauta a lo que hay que obtener a lo largo del desarrollo del juego más no el cómo desarrollarlo, de eso se encarga el proceso. Dada la complejidad de un videojuego, dicho proyecto debe de manejarse a través de un proceso, involucrando así una serie de pasos organizados que guiarán cada actividad hasta el producto final.

2. Procesos Viables al Desarrollo de Videojuegos

Se realizó un estudio y análisis de procesos de desarrollo de videojuegos y dicha investigación nos llevó a concluir que la industria se aferró muchos

años a utilizar la metodología cascada (Keith, Waterfall Game Development, 2009) y muchas compañías siguen creando productos de esta manera; del estudio mencionado se desprende que no existen procesos específicos para el desarrollo de videojuegos que sean públicos; posiblemente existan algunos procesos cerrados en el sentido de que su información, modelos, plantillas y herramientas no están disponibles al público en general y por lo tanto puede considerarse que existe la necesidad de modelos de procesos que puedan ser utilizados por la industria en general y que en su momento contribuyan al desarrollo de este tipo de aplicaciones.

En principio hay que considerar que un videojuego es una aplicación de software, esto orilló a buscar procesos ya existentes para el desarrollo de software; se analizaron en torno a sus actividades, roles y artefactos para ver que tan plausibles eran para guiar proyectos de videojuegos, la tabla 1 muestra los procesos considerados para este estudio.

Tabla 1. Procesos Viables al Desarrollo de Videojuegos

Proceso	Descripción	Valoración
Waterfall Process	Proceso de desarrollo de software especializado secuencial en el cual el desarrollo se basa en el modelo Cascada a través de las fases de concepción, iniciación, análisis, diseño, construcción y pruebas (Flood, 2003).	El producto final se demora más de lo esperado ya que cualquier problema que se presente en una de las etapas se tiene que regresar a una anterior para corregirlo. Requiere hacer muchos cambios a los documentos y regresarse a etapas anteriores propicia a que se vuelva un proceso muy desordenado. Sin embargo es uno de los procesos más populares en la industria de los videojuegos.
Rational Unified Process	Es un proceso de desarrollo de software iterativo, es adaptable y entallable para satisfacer las necesidades del equipo del proyecto (Kruchten, 2004). Comúnmente sigue una metodología pesada.	Un proceso muy completo pero no está enfocado al desarrollo de videojuegos. Demasiada documentación permite realizar un buen producto con lo que ello conlleva (tiempo, dinero y personas involucradas). Concluimos que no es necesario tener tanta documentación para el desarrollo de un videojuego.
Essential Unified Process	Consiste en integrar las prácticas acertadas que son recursos de los tres campos principales del proceso: el campo del proceso unificado, métodos ágiles y el campo de madurez del proceso. Cada uno de ellos apoya a capacidades diferentes. EssUP se centra en las prácticas esenciales que se creen deben tener todos los proyectos de desarrollo de software (Jacobson, 2009).	EssUP promueve un buen trabajo en equipo y está preparado para separar el trabajo creativo del mecánico. Separa los artefactos en alfa y beta, siendo alfa los más importantes en el proyecto. Es posible separar las prácticas del proceso y adaptarlas al desarrollo de videojuegos.
OpenUP	OpenUP es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. La mayoría de los elementos de OpenUP están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.	La descripción del proceso insiste mucho en la colaboración en equipo y la inclusión de *stakeholders* como parte del proceso. Es de cierta manera similar a Scrum, a diferencia que en Scrum los *stakeholders* no participan en las reuniones diarias para comentar el estado del proyecto. Sin embargo hay muy poca información del proceso como para realmente separarlo de las demás variaciones de Unified Process, pero la filosofía del arduo trabajo en equipo es esencial para el desarrollo de videojuegos.
Team Software Process	Es un conjunto de procesos estructurados que indican qué hacer en cada fase del desarrollo del proyecto y muestra cómo conectar cada fase para construir un producto completo (Humphrey, 2000). El objetivo principal de TSP es completar con éxito, a través de varios ciclos de desarrollo incremental, un pequeño proyecto de software con calidad, siguiendo fielmente el proceso y manteniendo durante cada ciclo de desarrollo un equipo eficiente y colaborativo.	Proporciona un balance entre proceso, producto y equipo de trabajo. Sus fases y tareas están bien definidas. Contiene todas las formas, guiones y estándares necesarios para poder registrar y seguir el proceso. Nos enseña los procedimientos para iniciar un proyecto, los pasos para poder guiarlo y nos muestra como analizar y reportar los datos obtenidos. Lo más interesante de este proceso es el documento Postmortem, artefacto que comparte con MSF, pero en TSP esa retroalimentación ocurre por ciclo, similitud que comparte con Scrum. Es común realizar un documento similar al final del ciclo de vida del desarrollo de un videojuego.
Microsoft Solution Framework	Microsoft Solution Framework es una serie de principios, modelos, disciplinas, conceptos y guías para diseñar aplicaciones de Microsoft (Keeton, 2006). Consiste en una serie de ciclos pequeños e iteraciones. Este modelo permite el desarrollo rápido con aprendizaje y refinación continua debido al entendimiento progresivo de los requerimientos de los clientes. Utiliza una metodología pesada y ágil.	La filosofía de MSF es que no hay una sola estructura o proceso que se aplica óptimamente a los requerimientos y ambientes de todo tipo de proyectos, por lo tanto se puede adaptar y soportar cualquier proyecto sin importar el tamaño o complejidad y reteniendo una serie de principios y perspectivas que podrían ser adaptables al proceso de desarrollo de un videojuego. Actualmente no existe una aplicación de MSF para el desarrollo de videojuegos. MSF insiste en la importancia de la constante retroalimentación

		de experiencias (buenas y malas) de proyectos pasados.
Scrum Framework	Scrum es un framework de desarrollo de software iterativo-incremental utilizado en el desarrollo de software ágil. El trabajo está estructurado en ciclos conocidos como sprints. Durante cada sprint los equipos toman los requerimientos de una lista ordenada por prioridades conocidas como historias de usuario. Al terminar cada sprint, se tiene una versión potencialmente final del producto (Scrum Alliance, 2009).	Scrum facilita la iteración, permite a los equipos entregar características pulidas para probar la calidad del juego a lo largo de su desarrollo y así incorporar la retroalimentación de jugadores. Scrum no es solo para programadores, involucra a muchas personas a un solo proyecto. Es útil debido a que los videojuegos hoy en día se vuelven más complejos e involucran a personas multidisciplinarias. Por estas razones, consideramos Scrum ideal para el desarrollo de videojuegos.

Pese a la gran variedad de procesos que podrían ser susceptibles de adecuarse para el desarrollo de videojuegos, se optó por diseñar proceso caracterizado desde el principio a este tipo de software y tomando como base algunos elementos de los procesos mencionados en la Tabla 1.

Nos enfocamos principalmente en dos: *"Waterfall Process"*, y debido a su valoración positiva en la investigación, el *Scrum Framework*

Tabla 2. Waterfall Vs. Scrum

VENTAJAS	
WATERFALL	**SCRUM**
La planificación es sencilla.	Incremento en la productividad.
Si se conocen el total de los requerimientos desde el principio, la calidad del videojuego resultante es alta.	Mejoras constantes.
Permite trabajar con personal poco calificado.	El producto total se convierte en una serie de pequeños pedazos manejables.
Se tiene todo bien organizado.	Existe un progreso, inclusive si los requerimientos no están bien definidos.
No se mezclan las fases.	Todo es visible para todos.
Es perfecto para aquellos videojuegos que son rígidos donde se especifiquen muy bien los requerimientos y se conozcan muy bien las herramientas a utilizar.	Existe una gran comunicación en el equipo.
	El equipo comparte los éxitos desde el principio hasta el final.
	El cliente se mantiene informado en cada mejora del producto.
	Entrega de un producto funcional al finalizar cada *sprint*.
	Posibilidad de ajustar la funcionalidad en base a las exigencias de los jugadores.
	Visualización del videojuego día a día.
	Alcance acotado y viable.
	Equipos integrados y comprometidos con el desarrollo del videojuego, toda vez que ellos definieron el alcance y se auto-administran.
	Capacidad para aceptar modificaciones sobre la marcha sin influir en el desarrollo.
	Prioridades a características del videojuego gracias al *Product Backlog*.
DESVENTAJAS	
La duración de todo el ciclo es muy larga.	No genera toda la evidencia o documentación de otras metodologías.
Probabilidad alta de fracaso dado que existe poca comunicación con el usuario final.	Tal vez sea necesario complementarlo con otros procesos ágiles como XP.
El mantenimiento se realiza en el código fuente.	Un mal uso de la metodología puede dar lugar a un desarrollo sin final.
Las revisiones de videojuegos de gran complejidad son muy difíciles.	Si no se tiene experiencia en seguir procesos de desarrollo, puede ser caótica su uso
Impone una estructura de gestión de proyectos.	
Para que el videojuego tenga éxito deben desarrollarse todas las iteraciones.	
Si se cambia el orden de las fases el videojuego final será de menor calidad.	
Se retrasa la localización y corrección de errores.	
Puede producir videojuegos poco llamativos para los jugadores ya que no se le pueden hacer muchas modificaciones según la marcha.	
Inflexibilidad del modelo: dificultad para responder a cambios en los requerimientos.	

3. Un proceso para desarrollo de videjuegos: Huddle

Se tomó la decisión de llamar Huddle a este proceso siguiendo la analogía con SCRUM, se llama Huddle a la reunión que se realiza en el juego antes de cada jugada en el futbol americano; la filosofía es que mediante breves reuniones de planeación a corto plazo, se planee cada "jugada" que se inicie; con esto se da un seguimiento más estrecho al avance del proyecto y es posible hacer correcciones tempranas a posibles desviaciones.

Huddle es un proceso específico para desarrollo de videojuegos con las siguientes características: ágil, óptimo para equipos multidisciplinarios de 5 a 10 personas, iterativo, incremental y evolutivo. Huddle, sin embargo, puede utilizarse en equipos de menos de cinco elementos.

Todo el proceso se divide en 3 fases:

- Preproducción.
- Producción.
- Postmortem.

Figura 3. Las tres fases del proceso Huddle.

Adicionalmente a los roles propuestos en SCRUM, en Huddle existen dos roles importantes a considerar: *Game Designer* y *Project Manager*.

3.1 Preproducción

La planeación de un proyecto es clave para obtener un producto de calidad y que sea desarrollado dentro del tiempo y costo estimados. Preproducción tiene como objetivo migrar la idea del diseñador al *Feature Log* y al *Sprint Plan*; estos documentos darán la pauta a la planeación y producción del videojuego.

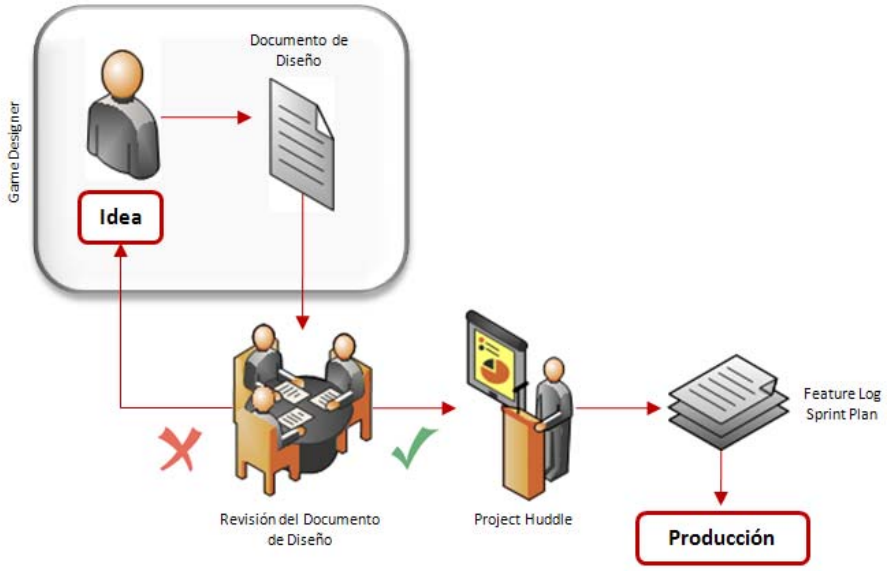

Figura 4. Modelo de Preproducción.

Entre sus propósitos principales se encuentra, el análisis del proyecto, fase en la que se revisará y se aceptará, en su caso, la idea inicial y se realizará la planeación completa de la fase de producción, en esta fase es necesaria la participación de todos los miembros involucrados en el proyecto.

Inicialmente se parte de un documento de diseño que expresa formalmente la idea principal y detalles de la propuesta de videojuego. Este documento es revisado con la finalidad de saber si es factible, en caso contrario se modificará el documento de diseño hasta que sea aprobado o en su caso rechazado definitivamente. Si el proyecto es aprobado se pasará al *Project Huddle*, en el cual se hará la planeación completa del proyecto para poder pasar a la fase de producción, una vez terminada la actividad se deberá contar con el *Feature Log* y con un *Sprint Plan*.

En el *Project Huddle* es necesaria la participación de todos los integrantes del equipo así como del *Game Designer* ya que aquí se decidirá el rumbo que tomará el proyecto durante la etapa de producción, desde las características del videojuego hasta los tiempos estimados de cada sprint y del proyecto en general.

3.2 Producción

La segunda etapa, la más importante y la más larga es la de Producción. Ésta, se apoya en herramientas del Scrum Framework, como son los *Daily Meetings*, los *Sprints* y *Sprint Reviews*, y en artefactos como el *Sprint Backlog* y *Burn-down Charts*.

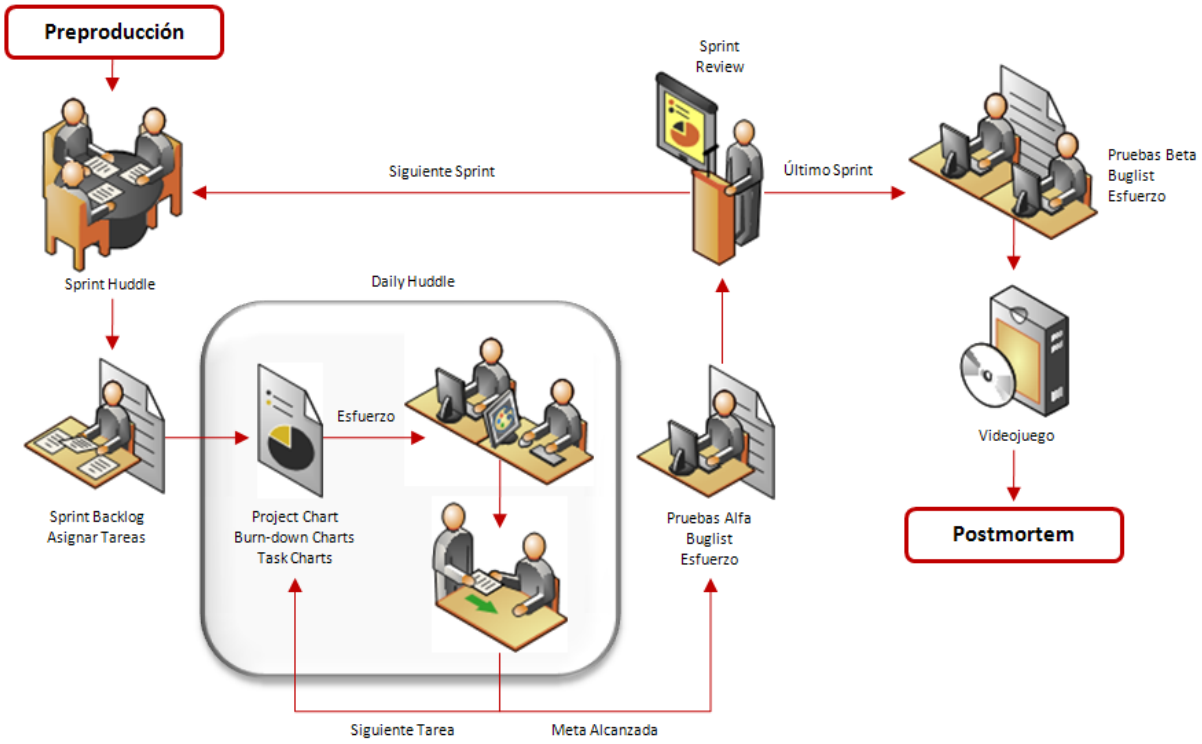

Figura 5. Modelo de Producción.

Huddle sin embargo, no se apoya en los roles de *Scrum*. El rol de *Project Manager* al igual que el de un *ScrumMaster* es asegurarse que el proceso se siga al pie de la letra. Sin embargo en Huddle también se encarga de que los artefactos estén al día y de que el equipo trabaje de manera eficiente.

El *Game Designer* juega un papel muy importante en esta fase, ya que estará pendiente de cómo se van generando los diferentes requerimientos especificados en la etapa de preproducción y de las características nuevas que se puedan agregar durante los *sprints* para planearlas y calendarizarlas.

Esta etapa comienza con el primer *sprint* entrando de lleno al desarrollo del videojuego. Una vez que se ha comenzado el *sprint*, se lleva a cabo el *Sprint Huddle*, en la cual se reúne todo el equipo y se analizan los requerimientos que se definieron anteriormente en el *Feature Log*, generando entonces un *Sprint Backlog* que contiene las tareas a realizar para poder lograr la meta del sprint.

Las tareas, al igual que en *Scrum*, son seleccionadas por los miembros del equipo, ellos se administran, deciden el tiempo necesario para su desarrollo y si necesitan más personas involucradas. En esta parte del proceso se hace evidente el trabajo en equipo ya que será necesario que todos los integrantes obtengan y/o proporcionen la ayuda necesaria para lograr la meta del *sprint*.

Una vez asignadas las tareas los miembros se reúnen diariamente (*Daily Huddle*) y discuten su progreso y/u obstáculos presentados. El trabajo se registra en un artefacto conocido como *Burn-down Chart*, tomado de *Scrum*.

Las *Burn-down Charts* son una herramienta importante en Huddle, proporcionan a los miembros una representación visual del trabajo realizado y el que queda por hacer del *sprint*, permitiendo así a los miembros organizarse mejor en caso de que el progreso sea lento.

Al alcanzar la meta del *sprint*, se procede a la etapa de pruebas *Alfa*. Durante esta actividad, los miembros dedicados a pruebas analizan cada característica que se implementó durante el *sprint*, se aseguran que efectivamente la meta del *sprint* se haya alcanzado y que no haya errores en la codificación e integración de recursos. Se mantiene un control de errores en un artefacto conocido como *Buglist*. El *sprint* no puede terminar si este proceso de depuración no concluye con la solución de los errores en el *Buglist*.

Al término de todos los *sprints*, se alcanza el hito de una versión beta del videojuego, el cual será probado por personas que no sean miembros del

equipo de desarrolladores. En esta etapa se manejan dos tipos de estrategias de pruebas, *Closed Beta* y *Open Beta*.

Al finalizar las pruebas *Beta*, se obtiene el producto final o *Gold Master*. Una vez obtenido, se deberá pasar a la etapa final que recibe el nombre de Postmortem.

3.3 Postmortem

La etapa del postmortem consiste en generar un reporte cuyo propósito es describir a detalle las actividades específicas que fueron más efectivas para el proyecto desde el inicio del proceso hasta la entrega del producto; de igual manera, describe las actividades que llegaron a perjudicar el desarrollo junto con sugerencias para corregir dichos problemas con la finalidad de no acarrearlos al siguiente proyecto.

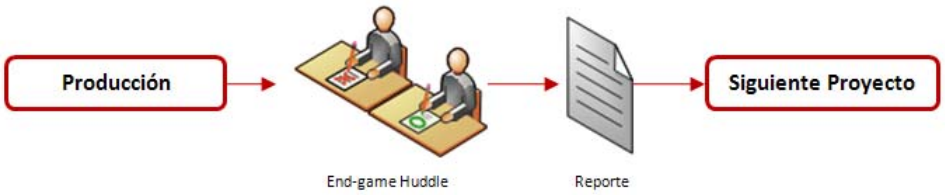

Figura 6. Modelo de Postmortem.

Para realizar esta fase, el equipo debe realizar la última actividad llamada *End-game Huddle* en la cual se analizarán los aspectos positivos y negativos del proyecto. Una manera de llevar estas reuniones es respondiendo entre todos, las siguientes preguntas:

¿Qué salió bien?
¿Qué salió mal?
¿Qué obstáculos se presentaron?

Del *End-game Huddle* saldrán sugerencias que deberán ser analizadas y filtradas con la intención de generar un reporte que incluya todas aquellas propuestas de mejora al proceso con el objetivo de que sean incorporadas en el próximo proyecto.

Generar el reporte postmortem es importante debido a que resulta más sencillo comenzar a un proyecto basándose en los resultados obtenidos de otros; es de esta manera como se tiene un desarrollo más confiable y eficiente; como se mencionó anteriormente, el conocimiento y la experiencia son los factores más importantes cuando se desarrolla un videojuego.

4. Plantillas y Herramientas

Una de las características de cualquier proceso de desarrollo de software es que debe de ser soportable (Sommerville, 2002), es decir, las actividades deben de poder desarrollarse en herramientas que ayuden a la aplicación del proceso.

Huddle cumple con esto, proporcionando plantillas para el Documento de Diseño, el Reporte Postmortem y una herramienta desarrollada en Microsoft Excel 2007 que engloba todos los artefactos que se generan durante la fase de Producción.

4.1 El Documento de Diseño

El Documento de Diseño es el artefacto más importante del proceso, se acarrea a lo largo del proyecto y deberá estar dispuesto a sufrir varios cambios desde la etapa de revisión. Contiene todas las especificaciones necesarias para comenzar el proyecto, éstas van desde el tema principal del videojuego hasta el número de niveles que tendrá.

El diseñador del juego asienta la idea a este documento con información detallada del proyecto, desde el título del juego, el género, una visión general y mecánica, aspectos de jugabilidad, modos de juego, plataforma, software que se utilizará, etc.

Los videojuegos comúnmente tienen conceptos muy originales, y estandarizar este documento sería como estandarizar la creatividad, hay secciones y campos que pueden o no aplicar al proyecto, sin embargo, hay cierta información que es esencial en todos los documentos de diseño de videojuegos.

A lo largo del proyecto, este documento sufrirá varios cambios, es importante que la primera versión de este documento no contenga información tan específica del videojuego, no se puede saber todo desde el principio, tal vez el juego requiera más niveles o menos, la mecánica puede cambiar a algo más entretenido o se pueden agregar otros modos de juego, conocimiento y experiencia.

El Documento de Diseño es la guía del proceso Huddle, es el que define si el proyecto merece entrar a una etapa de Producción y dicta las características que se registran en el *Feature Log*.

El documento será sometido a revisión con escrutinio por parte de desarrolladores independientes que conformarían parte del equipo o por medio de un representante o ejecutivos de una empresa que publica videojuegos. Este paso dicta el futuro del proyecto, es probable que los ejecutivos decidan obstaculizar el proyecto, después de todo, ellos lo publicarán, puede que la idea nunca llegue a la etapa de Producción y deberá cambiarse de manera mínima, severa o incluso abandonarla.

En un grupo con desarrolladores independientes cambia el caso, comúnmente es parte del rol de *Game Designer* escoger su equipo de trabajo a través de las habilidades que el proyecto necesita, el equipo entonces opina que partes del documento y del videojuego exceden capacidades o que otras características se podrían agregar, es sin embargo el diseñador el que tiene la última palabra.

Tabla 3. Plantilla del Documento de Diseño

CAMPO	DESCRIPCIÓN
CONCEPTO	
Título	El título del juego, debe ser un nombre que capte la atención del jugador y del lector del documento. A grandes rasgos, debe de incluir el concepto del juego. El titulo debe ser algo memorable.
Estudio/Diseñadores	El nombre del estudio y/o del diseñador o diseñadores del documento.
Género	El género abarca que tipo de juego será. Simulación, FPS, Rol, etc.
Plataforma	Qué hardware se requiere para jugarlo. Computadora Personal, Xbox 360, PS3, etc.
Versión	La versión del documento. Debe ser un número y no una fecha. (Ver el campo de Historial de Versiones)
Sinopsis de Jugabilidad y Contenido	En uno o dos párrafos, describir la esencia de jugar el juego. Incluir un poco del contenido que tendrá, historia, personajes, objetivo, etc.
Categoría	Comparar con uno o varios juegos existentes y enfatizar en las diferencias y características principales de este juego.
Licencia	Describir si el juego está basado en un libro o en una película. Si es original, se puede omitir este campo o describir el por qué puede convertirse en una franquicia.
Mecánica	Describir la jugabilidad y el control del juego. ¿Qué hace el jugador? ¿Qué usa para lograr sus objetivos?
Tecnología	Enlistar que hardware y software se requiere para producir el juego. Desde lenguaje de programación hasta editor de sonidos.
Público	¿A quién va dirigido el juego? ¿Quién lo jugará? Se puede describir una demografía como niños o adolescentes, sin embargo, es más sencillo describir un tipo de jugador, ya sea casual, competitivo o veterano.
HISTORIAL DE VERSIONES	
El Documento de Diseño es un artefacto que siempre estará sujeto a cambios, por lo tanto, un control para las diferentes versiones del documento y de los cambios que se han hecho es esencial. El número de versión varía de acuerdo a si es un cambio mínimo o uno muy radical. El historial no se incluye en un documento que se somete a revisión por una empresa o grupo de desarrolladores debido a que incluye fechas, esto para evitar que juzguen la idea del juego como un concepto viejo.	
VISIÓN GENERAL DEL JUEGO	
Debe de establecer la visión y el enfoque del juego que guiará al proyecto hasta el final del proceso. El resumen debe mencionar lo más interesante, las ventajas y lo original del juego. ¿Por qué las personas jugarían este juego? La estructura de los párrafos es similar a un ensayo, una introducción debe de abarcar todos los aspectos importantes mientras que los párrafos subsecuentes deben detallar lo mencionado en la introducción. Al final, la conclusión debe dejar al lector entusiasmado y emocionado por jugar el juego.	
MECÁNICA DEL JUEGO	
Esta sección esencialmente describe lo que el jugador puede hacer y cómo puede hacerlo. Describir las acciones del jugador, de preferencia en secuencia a cómo será en el juego.	
Cámara	Describir el tipo de cámara que se utilizará. Es decir, qué perspectiva tiene el jugador ante lo que está viendo en el juego, si es 3D o 2D, vista isométrica, en primera persona, etc.
Periféricos	¿Qué periféricos utilizará el jugador para lograr los objetivos mencionados? Incluir todos los que apliquen: teclado, mouse, *gamepad*, micrófono, etc.
Controles	Describir los botones y teclas que invoquen las acciones mencionadas en la sección de Mecánica del Juego.
Puntaje	Explicar de qué manera el juego se mantiene al tanto de los logros del jugador. Incluir también si existe una tabla de puntajes que compare los mismos entre los jugadores, ya sea de manera local o en línea.
Guardar/Cargar	Describir cómo el jugador guarda su progreso de los objetivos logrados en el juego y cómo puede continuar los objetivos pendientes. De igual manera, describir los dispositivos de almacenamiento que se usarán o si el juego tiene un sistema de contraseñas.
ESTADOS DEL JUEGO	
Un estado del juego se refiere al lugar en donde se encuentra el jugador durante el juego, es decir, si el jugador está en el Menú Principal, está jugando un Juego Multijugador, está en el Menú de Pausa, etc. Los diagramas deben representar visualmente las relaciones entre los estados, si del Menú Principal se puede ir al Menú de Opciones, ¿Cómo lo hace? ¿Qué se ejecuta? ¿Qué interfaz muestra?	
INTERFACES	
Las interfaces dan la pauta a la interactividad que tiene el jugador con el juego, en esta sección se debe de describir la apariencia del juego, es decir, colores y temática. Es importante dejar una impresión visual en el jugador y obviamente debe de estar relacionada con el concepto del juego.	

Nombre de la Pantalla	El nombre de la pantalla, si es el Menú Principal o el H.U.D. (Heads-up Display).
Descripción de la Pantalla	¿Para qué sirve esta interface?
Estados del Juego	Enlistar todos los estados de juego que invoquen esta pantalla así como también los estados que se puedan invocar en ella.
Imagen	Una imagen que muestre en concepto cómo se vería la pantalla.
NIVELES	
Los juegos comúnmente se dividen en niveles o en mapas secuenciales dentro de los cuales se debe cumplir con ciertos objetivos para progresar en el juego. Existen juegos en los cuales los niveles solo cambian a razón de la dificultad y los objetivos siguen siendo los mismos, de igual manera se deben describir esos cambios en esta sección.	
Título del Nivel	El nombre del nivel.
Encuentro	Describir si es el primer nivel, un tutorial o un bonus, en otras palabras, ¿Cuándo es que el jugador llega a este nivel?
Descripción	Una descripción detallada del nivel.
Objetivos	¿Qué debe de hacer el jugador para terminar el nivel? Este campo también debe incluir si el jugador tiene que resolver ciertos acertijos o derrotar a cierto enemigo para progresar.
Progreso	Describir que ocurre cuando el jugador termina el nivel.
Enemigos	Si el nivel tiene enemigos que el jugador debe enfrentar, éstos se enlistan en este campo, de lo contrario este campo puede ser omitido.
Items	Enlistar los objetos que el jugador o los enemigos pueden usar y que aparecen en este nivel, este campo se puede omitir si no existen dichos objetos.
Personajes	Los personajes que aparecen en el nivel, de igual manera, este campo puede ser omitido si no existen personajes en el juego.
Música y Efectos de Sonido	Describir la música de este nivel al igual que los efectos de sonido de ambiente que contiene.
Referencias de BGM y SFX	Escribir todas las referencias que apliquen con respecto a la música de fondo y efectos de sonido descritos en la sección de Música y Sonidos.
PROGRESO DEL JUEGO	
Enlistar de manera secuencial o por medio de un diagrama de flujo los eventos o niveles que el jugador debe de pasar para progresar en el juego. Existen juegos que tienen distintos modos de juego, en ese caso se requieren varias listas y/o diagramas.	
PERSONAJES	
Los personajes principales y secundarios que aparecerán en el juego. Esta sección se puede omitir si el juego no tiene personajes.	
Nombre del Personaje	El nombre del personaje.
Descripción	Describir detalladamente el físico del personaje, si es humano o extraterrestre, su vestimenta, etc.
Imagen	Fotografía o dibujo conceptual del personaje.
Concepto	Describir la conducta y comportamiento, al igual que los motivos del personaje. Mencionar también si es el enemigo principal o el protagonista. El concepto también puede relatarse como una historia del personaje, detallando en las relaciones con otros personajes del juego.
Encuentro	¿Cuándo aparece este personaje en el juego?
Habilidades	Enlistar las habilidades del personaje.
Armas	Enlistar las armas del personaje.
Items	Enlistar los objetos del personaje.
Personaje No-Jugable	Si el personaje no es controlable por el jugador, describir su propósito para el juego y/o para el jugador.
ENEMIGOS	
Los enemigos obstaculizan el progreso del jugador, pueden ser máquinas, otros personajes, monstruos, etc.	
Nombre	El nombre del enemigo.
Descripción	Describir detalladamente el físico del enemigo así como también su comportamiento.
Encuentro	¿Cuándo aparece este enemigo en el juego?
Imagen	Fotografía o dibujo conceptual del enemigo.
Habilidades	Enlistar las habilidades del enemigo.
Armas	Enlistar las armas del enemigo.
Items	Enlistar los objetos del enemigo.
HABILIDADES	
Los personajes y los enemigos llegan a tener ciertas habilidades fuera de las acciones comunes, en esta sección se describen cada una de ellas.	
ARMAS	
En esta sección se describen las armas que aparecerán en el juego.	
ITEMS	
Todos los objetos especiales que ayudan al jugador a realizar los objetivos y progresar en el juego se mencionan aquí.	
GUIÓN	
En esta sección se incluyen todos los diálogos del juego. Estos pueden ser muy variantes o inexistentes dependiendo de la naturaleza del juego. El guión debe de incluir encabezados, nombres, diálogo, acción y transiciones.	
LOGROS	
Describir los varios logros o hitos que el jugador obtiene mientras progresa en el juego. Estos pueden otorgar medallas, personajes secretos o puntos extra.	
CÓDIGOS SECRETOS	
Describir los códigos secretos que el jugador puede ingresar, lo que hacen y cómo son ingresados.	
MÚSICA Y SONIDOS	

La música y/o sonidos que se usarán en el juego, nombre, descripción junto con un número de referencia. Si es música de fondo, la referencia debe de empezar con una 'M' seguida de un número en secuencia. Si es un efecto de sonido, empezar con 'S'.	
IMÁGENES DE CONCEPTO	
Todas las imágenes que muestren algún posible nivel, personaje, objeto, etc., deben ser incluidas en esta sección y deben estar enumeradas y con título.	
MIEMBROS DEL EQUIPO	
Información de las personas que trabajarán en el proyecto, incluye su nombre, el rol o roles que desempeñan y medios por los cuales se les puede contactar.	
DETALLES DE PRODUCCIÓN	
Antes de entrar a la etapa de Producción, se definen en el documento algunos detalles del proyecto.	
Fecha de Inicio	¿Cuándo empieza la etapa de Producción del proyecto?
Fecha de Terminación	¿Cuándo termina la etapa de Producción del proyecto?
Presupuesto	Una estimación aproximada del presupuesto del juego.

4.2 Documentos de Producción

Utilizando *Microsoft Excel 2007*, se elaboró una herramienta compilando las diferentes plantillas que se utilizan durante la fase de Producción que incluye: *Feature Log*, *Sprint Plan*, *Sprint Backlog*, *Project Charts*, *Burn-down Charts*, *Task Charts* y *Buglist*.

La herramienta ordena las prioridades de las tareas, cambia el formato según su estatus, genera gráficas, entre otras funciones útiles; todo por medio de botones pre-programados.

Tabla 4. Plantillas de los Documentos de Producción

CAMPO	DESCRIPCIÓN
DETALLES DEL PROYECTO	
Título	El título del juego especificado en el Documento de Diseño.
Estudio/Diseñadores	El nombre del estudio y/o del diseñador o diseñadores del videojuego.
Género	El género del juego.
Plataforma	El hardware que se requiere para jugarlo.
Fecha de Inicio	Fecha en la que se iniciará el proyecto.
Fecha de Término	Fecha de término para el proyecto.
Planeado	Porcentaje de metas del *Sprint Plan* que ya fueron planeadas pero aún no se encuentran en desarrollo.
En Desarrollo	Porcentaje de metas del *Sprint Plan* que ya fueron planeadas y se encuentran en desarrollo.
Sin Planear	Porcentaje de metas del *Sprint Plan* que no han sido planeadas.
Terminado	Porcentaje de metas del *Sprint Plan* que ya han sido terminadas.
SPRINT PLAN	
SprintID	Identificador numérico del *sprint*.
Inicio	Fecha de inicio del *sprint*.
Días	Número de días que tomará el *sprint*.
Fin	Fecha de término del *sprint*.
Meta	Meta a alcanzar en el *sprint*.
%	Porcentaje total del *sprint* dentro del proyecto.
Project Chart (Botón)	Crea la grafica del proyecto en caso de que ésta no exista.
PROJECT CHART	
Grafica del avance total del proyecto según los *sprints* planeados, terminados, sin planear y en desarrollo.	
FEATURE LOG	
FeatureID	Identificador numérico del *feature*.
Nombre	Nombre que recibe el *feature*.
Estado	Estado en el que se encuentra el *feature*, puede ser: Planeado, Sin planear, En desarrollo, Terminado, *Feature Creep* o Eliminado.
Días	Días que tomará desarrollar el *feature*.
SprintID	Número del *sprint* al que pertenece el *feature*.
Comentarios	Comentarios sobre el *feature*.
Sort Features (Botón)	Organiza los *features* según su estado y el *sprint* al que pertenecen.
SPRINT BACKLOG	
Sprint # Backlog	Título que incluye el número de *sprint* del cual se está creando el *backlog*.
Días	Número de días que tomará el *sprint*.
Tareas	Número de tareas que contiene el *Sprint Backlog*.
Tendencia	Promedio del esfuerzo que se ha realizado desde el inicio del *sprint*.
Esfuerzo Restante	Campos que contienen el esfuerzo restante según el número de día en que se encuentre.
Tendencia Actual	Campos que contienen el promedio del esfuerzo restante según como se ha trabajado desde el inicio del *sprint* respecto al día en que se encuentre.
Progreso Ideal	Campos que contienen el esfuerzo ideal restante según el día en que se encuentre y el número de días con que se cuenta para terminar el *sprint*.
Tareas Restantes	Campos que contienen el número de tareas restantes según el día en que se encuentre.
Nombre de la Tarea	Nombre que recibe la tarea.

FeatureID	Identificador numérico del *feature* al que pertenece la tarea.
Miembro	Miembro del equipo designado para realizar dicha tarea.
Rol	Rol que asume el miembro que realizará la tarea.
Estado	Estado en el que se encuentra la tarea, puede ser: Planeado, En desarrollo o Terminado.
Esfuerzo	Esfuerzo total de la tarea.
#'s	Campos que contienen el número del día en que se encuentra o que se usará como referencia para los cálculos de los demás campos.
Burn-down Chart (Botón)	Crea la *Burn-down Chart* del *sprint* en el que se encuentre, esto en caso de que no exista.
Task Chart	Crea la *Task Chart* del *sprint* en el que se encuentre, esto en caso de que no exista.
Organizar Tareas	Organiza las tareas según el *feature* al que pertenecen y su estado.
Task Slips	Genera tarjetas de cada una de las tareas enlistadas en el *Sprint Backlog* para entregar a los miembros.
TASK CHART – SPRINT #	
Gráfica que muestra el número de las tareas restantes respecto a los días.	
BURN-DOWN CHART – SPRINT #	
Gráfica que muestra el esfuerzo diario realizado desde el inicio del *sprint*, el progreso ideal que debería de haber y el promedio del esfuerzo realizado.	
BUGLIST	
BugID	Identificador numérico del error encontrado en alguna fase de pruebas.
Descripción	Descripción del error dada por el *tester*.
Descripción Técnica	Descripción del error dada por el miembro del equipo designado a resolverlo, incluye especificaciones técnicas como línea de código, clase, recurso, etc.
Autor	Nombre del *tester* que encontró dicho error.
Estado	Estado en que se encuentra el error.
FeatureID	Número del *feature* al que pertenece.
Organizar Buglist (Botón)	Organiza los errores según el *feature*.
TASK SLIPS	
FeatureID	Número del *feature* al que pertenece.
Nombre	Nombre del *feature*.
Tarea	Descripción de la tarea de la que se crea la tarjeta.
Miembro	Nombre del miembro del equipo designado para realizar dicha tarea.
Esfuerzo Estimado	Estimado del total del esfuerzo que se necesitara para realizar dicha tarea.
Terminado	Esfuerzo realizado desde el inicio del *sprint*.
Restante	Esfuerzo restante respecto al estimado total y al realizado.

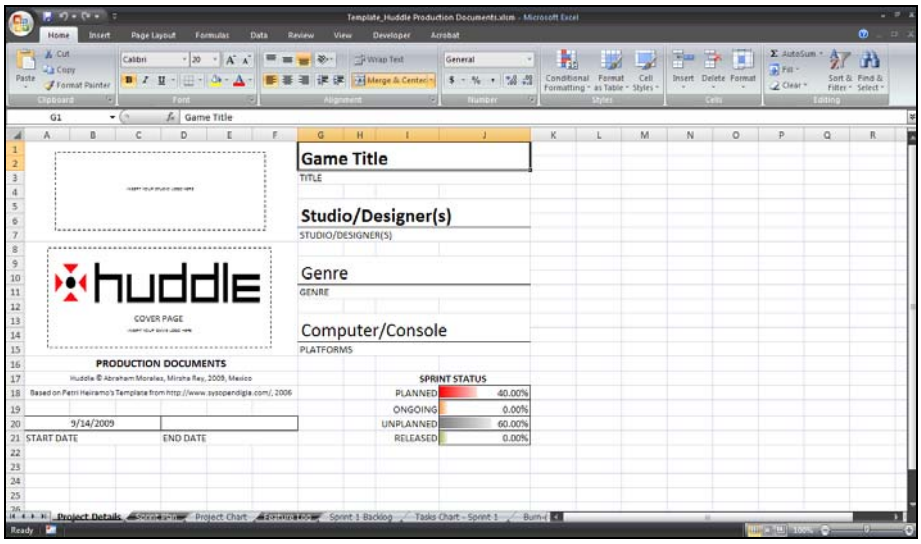

Figura 7. Detalles del Proyecto.

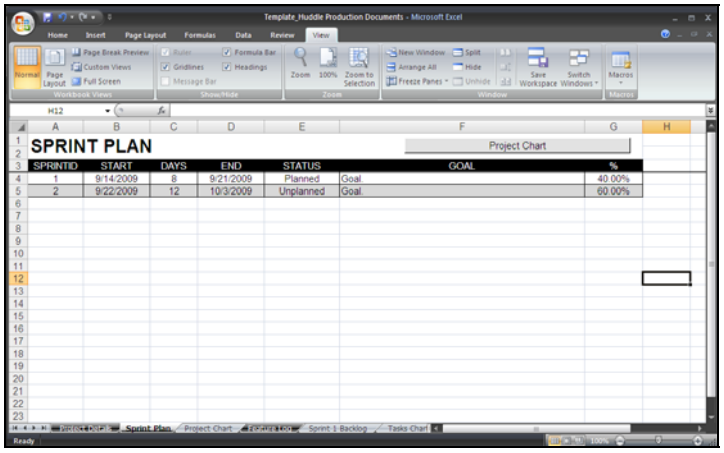

Figura 8. Sprint Plan.

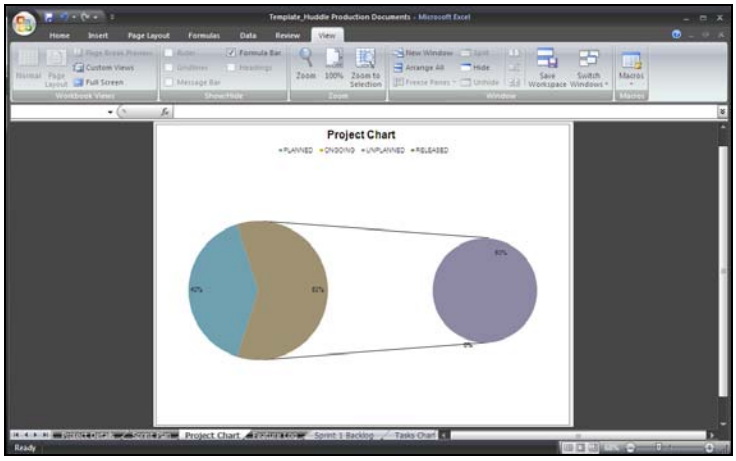

Figura 9. Project Chart.

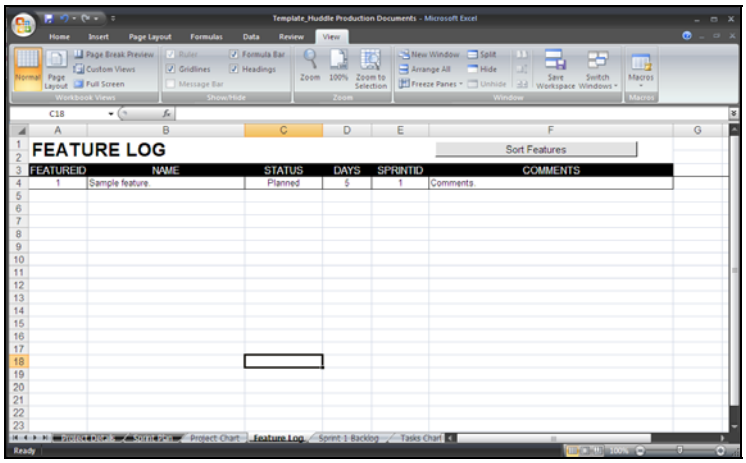

Figura 10. Feature Log.

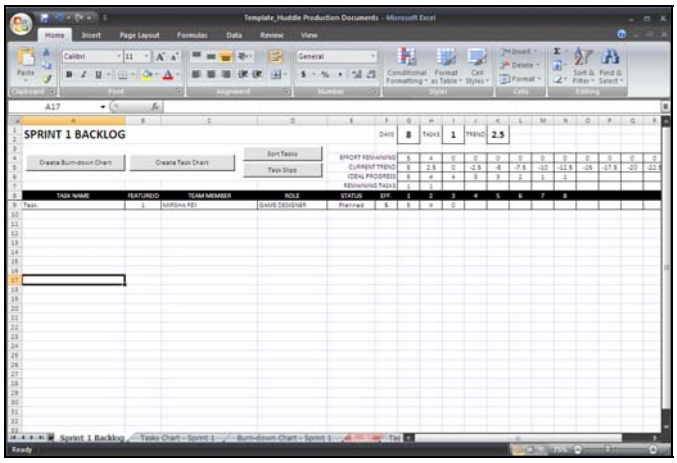

Figura 11. Sprint Backlog.

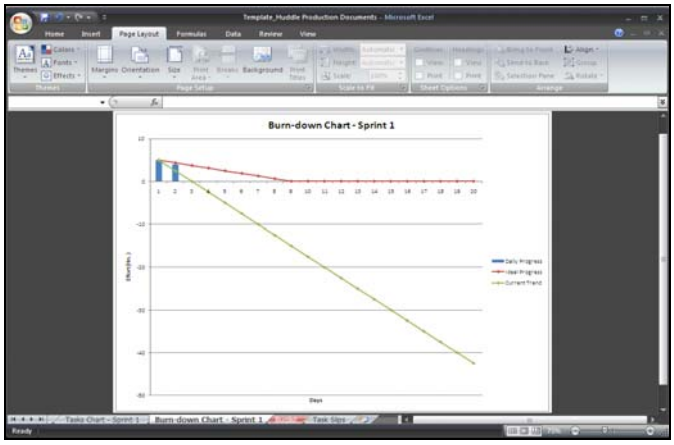

Figura 12. Burn-down Chart.

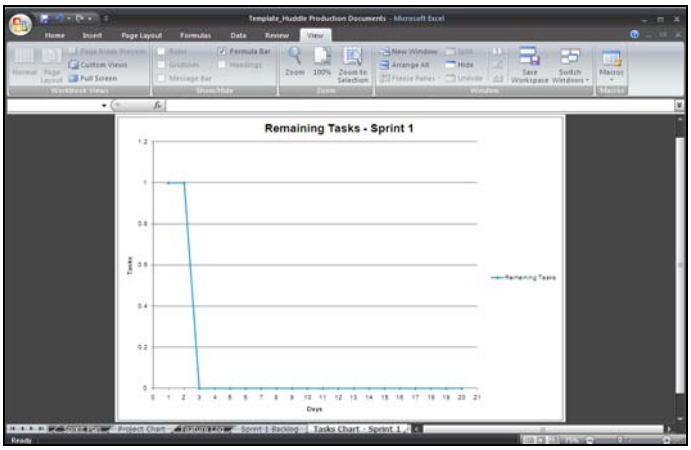

Figura 13. Task Chart.

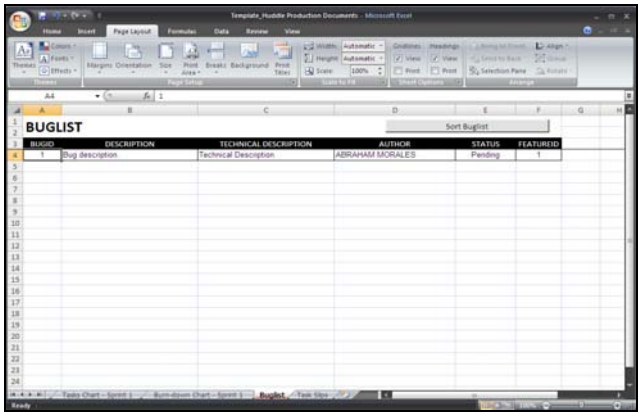

Figura 14. Buglist.

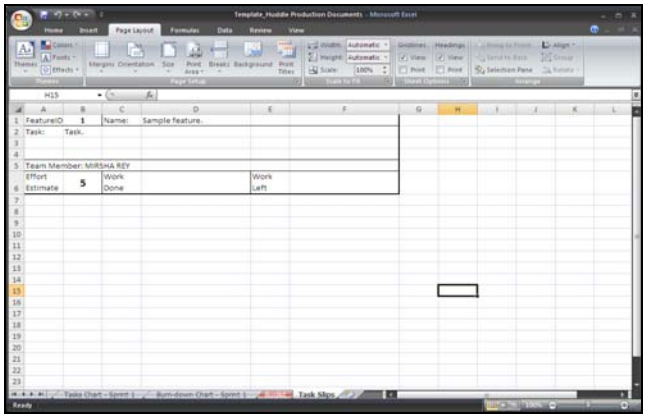

Figura 15. Task Slips.

4.3 Reporte Postmortem

El propósito del Reporte Postmortem es describir a detalle las actividades específicas que fueron más efectivas para el proyecto desde el inicio del proceso hasta la entrega del producto; de igual manera describe las actividades que llegaron a perjudicar el desarrollo junto con sugerencias para corregir dichos problemas para no acarrearlos al siguiente proyecto.

La meta del documento es entonces informar a los desarrolladores de proyectos futuros sobre los obstáculos a los que se enfrentó el equipo durante este proyecto. El reporte debe de enfocarse en identificar lo siguiente:

Tabla 5. Plantilla del Reporte Postmortem (Anexo 2)	
CAMPO	**DESCRIPCIÓN**
ANTECEDENTE	
Título	El título del juego especificado en el Documento de Diseño.
Estudio/Diseñadores	El nombre del estudio y/o del diseñador o diseñadores del videojuego.
Género	El género del juego.
Plataforma	El hardware que se requiere para jugarlo.
Antecedente	Una descripción breve pero detallada del proyecto que incluya los aspectos más importantes y característicos del videojuego.
MIEMBROS DEL EQUIPO	
Información de las personas que trabajaron en el proyecto, incluyendo su nombre, el rol o roles que desempeñaron y los medios para contactarlos.	
SIGN-OFFS	

Incluye los nombres, roles y firmas de los que realizaron el reporte, de igual manera incluye la fecha de cuándo se realizó el reporte.	
REFERENCIAS	
En esta sección, se enlistan numéricamente todos los documentos que se mencionarán en el reporte junto con la fecha de su elaboración.	
EFECTO POSITIVO	
Es una lista de aquellas actividades que generaron un efecto positivo durante el proceso de desarrollo.	
Referencia	La referencia del documento.
Categoría	La actividad que se llevó a cabo.
Votos	Los votos de los miembros del equipo que opinen que dicha actividad se llevó a cabo satisfactoriamente.
Declaraciones	Descripción de cómo es que dicha actividad generó un efecto positivo.
¿Cómo continuar así?	En este campo se enlistan por referencia las actividades adjuntando sugerencias de cómo seguir generando un efecto positivo.
EFECTO NEGATIVO	
Es una lista de aquellas actividades que generaron un efecto negativo durante el proceso de desarrollo.	
Referencia	La referencia del documento.
Categoría	La actividad que se llevó a cabo.
Votos	Los votos de los miembros del equipo que opinen que dicha actividad perjudicó el desarrollo.
Declaraciones	Descripción de cómo es que dicha actividad generó un efecto negativo.
¿Cómo corregir?	En este campo se enlistan por referencia las actividades adjuntando sugerencias de cómo corregir estos problemas.
CONCLUSIÓN	
Esta sección sintetiza lo positivo y lo negativo del proyecto, a su vez, menciona las lecciones que se aprendieron a lo largo del desarrollo del mismo.	

5. El Futuro de Huddle

El siguiente paso para Huddle es ponerlo a prueba, se tiene pensado capacitar a desarrolladores de videojuegos con el proceso y generar dos equipos para llevar a cabo dos proyectos simultáneos.

Se recibirá y recopilará retroalimentación de los equipos durante las tres fases del proceso para realizar posteriormente los cambios pertinentes.

En discusiones con desarrolladores independientes, se recibieron buenas críticas al presentarles el modelo del proceso.

Hoy en día, la industria de los videojuegos está creciendo de manera acelerada, es por ello que requiere de constantes innovaciones. Huddle pretende ayudar a los desarrolladores que carecen de muchos recursos y persiguen una carrera en la industria.

Buscamos ayudar a México, donde se consta de mentes muy ingeniosas y creativas fascinadas por los videojuegos pero carecen del conocimiento para desarrollarlos.

Estamos seguros que Huddle guiará a los desarrolladores durante todo el proyecto de manera eficiente, divertida y proporcionando las herramientas necesarias para el desarrollo de su videojuego.

6. Referencias

Dille, F., & Zuur, J. 2007. *The Ultimate Guide to Video Game Writing and Design.* Lone Eagle Publishing Company.

Flood, K. 2003. *Game Unified Process.* Obtenido de GameDev: http://www.gamedev.net/reference/articles/article1940.asp.

Humphrey, W. 2000. *The Team Software Process.* Estados Unidos: Carnegie Mellon.

Jacobson, I. 2009. *Essential Unified Process.* Obtenido de Ivar Jacobson International: http://www.ivarjacobson.com/process_improvement_technology/essential_unified_process_software/

Keeton, M. 2006. *Microsoft Solutions Framework (MSF): A Pocket Guide.* Van Haren Publishing.

Keith, C. 2009. *Waterfall Game Development.* Obtenido de Agile Game Development: http://www.agilegamedevelopment.com/2009/01/in-dawn-of-video-game-development.html

Kruchten, P. 2004. *The Rational Unified Process: An Introduction.* Estados Unidos: Addison-Wesley.

Otter. 2008. *The Movie Industry Vs. the Gaming Industry.* Obtenido de Associated Content: http://www.associatedcontent.com/article/1015720/the_movie_industry_vs_the_gaming_industry.html

Scrum Alliance. 2009. *What is Scrum?* Obtenido de Scrum Alliance: http://www.scrumalliance.org/pages/what_is_scrum

Sommerville, I. 2002. *Ingeniería de Software. Un enfoque práctico.* México: Pearson Educación.