Centres universitaris adscrits a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Disseny i Producció de Videojocs

Análisis y estudio de las metodologías ágiles: Propuesta híper-ágil para estudios indie

Adrián Cañete Paterna
TUTOR: Dr. Adso Fernández Baena
2018 - 2019

TecnoCampus Mataró-Maresme

Abstract

Agile methodologies are one of the most important tools to make into account in the development of video games. Having an aid that resolves team conflicts and makes it go in the same direction is essential to achieve the goal. Therefore, the purpose of this work is to know the different existing methodologies and with the acquired knowledge to propose a new one that fits the needs of the smallest studies.

Resumen

Las metodologías ágiles son unas de las herramientas más importantes a tener en cuenta en el desarrollo de videojuegos. Disponer de una ayuda que resuelva los conflictos del equipo y haga que éste vaya en un mismo sentido es primordial para conseguir el objetivo. Por ello la finalidad de este trabajo es conocer las diversas metodologías existentes y con el conocimiento adquirido proponer una nueva que se ajuste a las necesidades de los estudios más pequeños.

Resum

Les metodologies àgils són unes de les eines més importants a tenir en compte en el desenvolupament de videojocs. Disposar d'una ajuda que resolgui els conflictes de l'equip i faci que aquest vagi en un mateix sentit és primordial per a aconseguir l'objectiu. Per això la finalitat d'aquest treball és conèixer les diverses metodologies existents i amb el coneixement adquirit proposar una nova que s'ajusti a les necessitats dels estudis més petits.

Índice

1. Introducción

Durante los años 80 un nuevo tipo de entretenimiento había surgido (Kent, 2016). Millones de jóvenes alrededor del mundo se agolpaban en bares y centros recreativos con sus bolsillos llenos de monedas para poder jugar unas partidas a los nuevos videojuegos que estaban triunfando, Pong (Atari, 1972), Galaga (Namco, 1981), Donkey Kong (Nintendo, 1981), Pacman (Namco, 1980), son ejemplos de cómo una nueva industria había nacido, y con ella, necesidades que nunca se habían planteado.

Crear un videojuego se fue complicando. Desde aquellas "arcades" donde solo un programador se encargaba de todo: código, gráficos, audio hasta ahora donde las súper producciones más complejas con títulos que ya son sagas multimillonarias como GTA (Rockstar, 1997), Assassin's Creed (Ubisoft, 2007) o Call of Duty (Activision, 2003), pueden ocupar a equipos multidisciplinares de cientos de personas repartidas por todo el mundo.

Actualmente, la creación de un videojuego necesita de un proceso de maduración que puede oscilar de los entre 6 meses a 5 años, dependiendo del producto final al que se quiere llegar. Este proceso se podría realizar de una manera más sencilla si en la organización se implementase correctamente una metodología específica.

Los equipos de desarrollo son multidisciplinares y los componen personas de muy diferentes perfiles como programadores, artistas, diseñadores, etc. Todas estas personas con visiones muy diferentes del producto tienen que tener en cuenta el objetivo final, saber en cada momento qué tarea se ha de realizar y de cuánto tiempo se dispone para realizarla.

Es por ello que las metodologías ágiles son herramientas imprescindibles para que los equipos consigan elaborar el videojuego deseado. Con esa finalidad, en este proyecto se propone el análisis de tres de las metodologías ágiles principales: Scrum, Kanban y Lean. Han sido escogidas porque son las metodologías que no son híbridas, es decir, son metodologías puras, que no se ven afectadas por otros tipos como se puede observar en el gráfico (ver Figura.1). La primera, Scrum, es la el más utilizada y de ella se derivan varias modificaciones para ajustarse al proyecto a realizar.

Pero no siempre estas metodologías han sido tan relevantes como ahora. Encontramos sus orígenes en los inicios del desarrollo del software para los primeros ordenadores. En ese momento, el software se hacía de manera lineal, todas las tecnologías surgían de manera más lenta y no se necesitaba una reacción rápida durante su desarrollo.

Por ello el desarrollo se organizaba con estructuras *Waterfall*. Las estructuras Waterfall consistían en conocer las demandas del usuario desde el principio, sin tener en cuenta posibles necesidades futuras, siguiendo una estructura de diseño, implementación y entrega. Esta metodología no representaba recursividad alguna ni planteaba cambios durante el desarrollo del videojuego.

Las necesidades y gustos de los usuarios comienzan a cambiar rápidamente, la tecnología cada vez avanza a un ritmo mayor y las empresas necesitan ser más productivas y eficientes. Es en este entorno en el que surgen las metodologías ágiles.

En este trabajo se pretende mostrar cómo las metodologías ágiles son actualmente una herramienta imprescindible en cualquier desarrollo de software y, más concretamente, en el desarrollo de videojuegos, para lograr alcanzar con éxito el propósito final del producto a realizar.

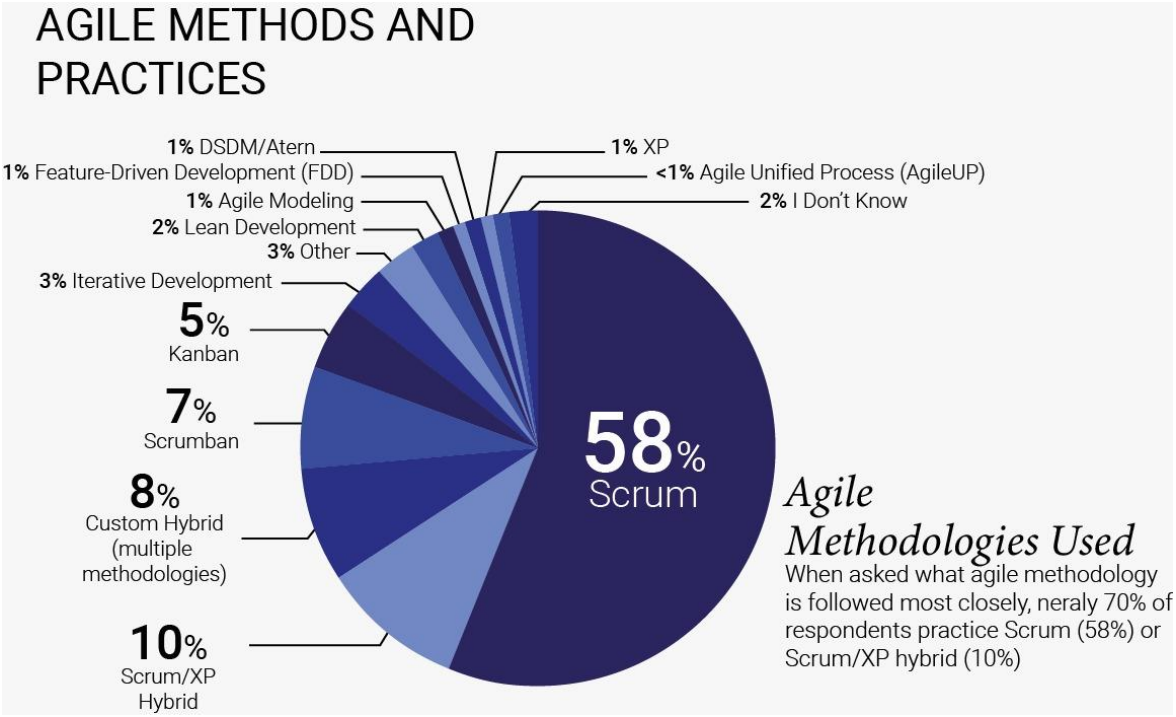

Figura 1. Gráfico uso metodologías ágiles. Fuente:(Gómez, García, y Dedo, 2017).

Una vez estudiadas las tres metodologías principales y extraídas sus diferentes características se definirá una nueva metodología (que podríamos considerar híbrida) con la experiencia obtenida en el estudio teórico previo. Con la metodología definida se propondrán una serie de entrevistas a diferentes profesionales del sector para poder conocer qué demandan de la metodología que están utilizando en ese momento y cómo la aplican a sus equipos, así como las carencias que han detectado y las mejoras que proponen.

Esta investigación se propone con la finalidad de profundizar en las metodologías ágiles más utilizadas en el desarrollo de videojuegos, analizando los puntos clave de cada una de ellas para aumentar el conocimiento sobre la gestión de un proyecto. Para poder aplicar estos conocimientos, se definirá una nueva metodología que nos mostrará cómo se han utilizado los diferentes conceptos teóricos y con las entrevistas se les intentará dar validez práctica.

2. Antecedentes de la investigación

Para poner en contexto el proyecto, a continuación se muestran tres ejemplos de metodologías que otros autores han ideado para poder responder a las necesidades que sus proyectos requerían. De esta manera se obtendrá conocimiento previo de cómo otras personas han resuelto el problema en la organización durante el desarrollo de un software.

2.1 Metodología para el desarrollo de aplicaciones móviles. (Mantilla, Ariza, & Delgado, 2013).

En el artículo de Mantilla, Ariza y Delgado se crea una metodología ágil pensada exclusivamente para la creación de aplicaciones móviles. La propuesta está dividida en cinco fases y en cada una de ellas se tienen que realizar las acciones necesarias para poder llevar a cabo el desarrollo con éxito.

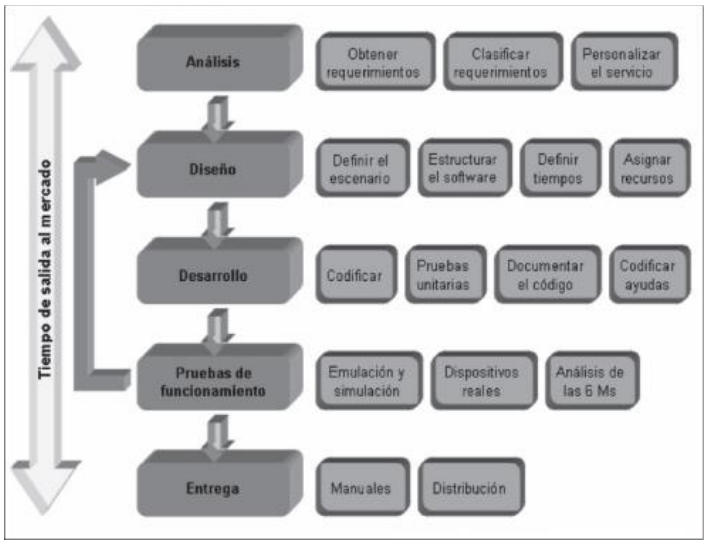

Figura 2.Estructura metodología ágil para aplicaciones móviles. Fuente: (Mantilla, Ariza, y Delgado, 2013).

Las cinco fases de la metodología son:

- Análisis.
- Diseño.
- Desarrollo.
- Pruebas de funcionamiento.
- Entrega.

Lo que convierte esta metodología en ágil, es su recursividad. Entre las tareas de diseño y las pruebas de funcionamiento se puede ir corrigiendo y adaptando el producto que se está creando, siendo éste el aspecto primordial de una metodología ágil.

Dentro de la etapa de análisis se deben obtener los requisitos del cliente y clasificarlos para poder identificar a qué parte de la aplicación estamos apuntando. Los autores diferencian los requisitos en: el entorno, el mundo, los requisitos funcionales y los no funcionales. El entorno estaría formado por las características que rodean al producto, como podrían ser la plataforma o el lenguaje de programación. El mundo, formado por los elementos con los que el usuario actuará. Los requisitos funcionales, son el producto básico con las diferentes funciones del mismo, y los no funcionales que podríamos definir también como intangibles, tales como la estabilidad, la ausencia de bugs o el rendimiento.

El siguiente punto a tratar es el diseño. Este es el primer apartado donde los autores proporcionan recursividad a la metodología, en el que se deben de definir el escenario y los posibles entornos donde se ejecutara la aplicación. También se ha de estructurar el software, identificando el código que se va a utilizar para la aplicación, y ordenarlo según su funcionalidad. Se han de definir los tiempos que se tardarán en realizar las diferentes tareas durante el desarrollo y finalmente han de asignarse los recursos necesarios a las diferentes tareas que se ejecutarán.

El desarrollo es la fase donde se deben programar todas las funcionalidades requeridas previamente diseñadas. Funcionalidades como realizar pruebas unitarias para comprobar que cada una de las funciones se ejecuten de manera correcta, documentar todo el código que se esté realizando para poder evitar problemas en un futuro, remarcando su uso y funcionamiento, y por último, programar todas aquellas ayudas que puedan ser relevantes para ayudar al usuario a realizar un uso correcto de la aplicación.

En la fase de pruebas de funcionamiento se emulará el funcionamiento de la app en diferentes entornos para verificar su correcto funcionamiento. También se debe de realizar un análisis de las 6M’s con la ayuda de expertos del sector para averiguar la capacidad de éxito de la aplicación.

Pasado este punto, si todo funciona correctamente, se pasaría a la fase de lanzamiento. Si no fuera así, es en este punto donde deberíamos volver al punto de diseño.

Según el artículo, los resultados obtenidos por los autores son buenos, pero no mencionan la existencia de error alguno ni tampoco parece que el desarrollo del proyecto haya sido muy largo, aproximadamente cuatro meses. Ambos aspectos son poco probables que ocurran en el sector de los videojuegos lo que nos lleva a concluir que una metodología para el desarrollo de videojuegos ha de ser mucho más modular y ágil.

2.2 Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje -MESOVA- (Castrillón, 2011).

En este artículo, el autor propone una metodología para la creación de objetos virtuales para educación. El autor difiere entre dos tipos de metodologías, las que se denominan de métodos pesados y la de métodos ágiles, siendo esta segunda la que nos interesa en este caso. Las diferencias entre estos dos tipos es la orientación en su concepción. Los métodos pesados se basan en el orden y la documentación. Un ejemplo podría ser la metodología Waterfall. Por otro lado, los métodos ligeros se crean en base a la comunicación entre los diferentes roles dentro del proceso de creación. La metodología Scrum sería un modelo de método ligero.

En cuanto a la metodología creada por el autor encontramos que, como la mayoría de las metodologías utilizadas actualmente, es una aglomeración de distintas metodologías ágiles como pueden ser XP (Extreme Programming), RUP (Rational Unified Process) y UP (Unified Process). Las tres 3 metodologías citadas están orientadas al desarrollo de software más convencional, donde la mayor parte del contenido es programación., Por este motivo no las trataremos en este trabajo.

En el caso que se está analizando, la metodología propuesta es una combinación entre metodología con métodos pesados y métodos agiles, formando así una metodología secuencial en la que hay saltos para poder crear acciones paralelas. En la imagen siguiente lo podemos observar esquemáticamente.

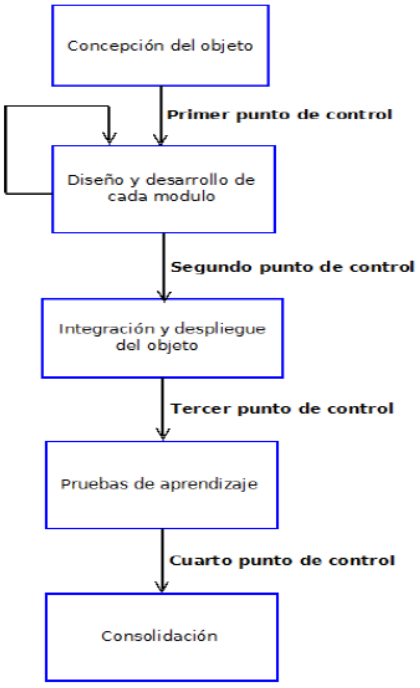

Figura 3.Fases de la metodología MESOVA. Fuente: (Castrillón, 2011).

Esta metodología se dividiría en seis fases diferentes, cada una de ellas con las actividades correspondientes y su punto de control. Los puntos de control están establecidos de manera que en cada uno de ellos se deben revisar las características de la actividad anterior para poder así validarla.

2.3 Designing an agile methodology for mobile software development: A hybrid method engineering approach (Rahimian & Ramsin, 2008).

En este artículo los autores marcan unas pautas muy estrictas para que su metodología sea totalmente ágil y no se incline hacia un desarrollo más lineal como había ocurrido con las anteriores. Para ello, los autores remarcan diferentes puntos.

El primer punto sería la agilidad durante el proyecto, algo totalmente imprescindible en cualquier metodología para que sea lo más rápida y ágil, posible adoptando los cambios que el proyecto necesita en cada momento.

Como segundo punto tendríamos la conciencia del mercado, es decir, saber en todo momento qué es lo que el cliente o el mercado necesitan para así poder crear el producto más afín a sus necesidades y por otro lado conocer cuáles son las plataformas más populares para desarrollar y cuáles son las tecnologías de vanguardia.

En tercer lugar, la reutilización y adaptación de tecnologías anteriores para diferentes proyectos. Con ello una metodología ágil pretende minimizar los tiempos y costes del software ya desarrollado para poder crear uno de nuevo.

Finalmente, la integración de días en el proyecto para revisar el trabajo realizado, poder aprender cómo mejorar durante todo el proceso y cómo se ha realizado el desarrollo del mismo, para analizar puntos de perfeccionamiento.

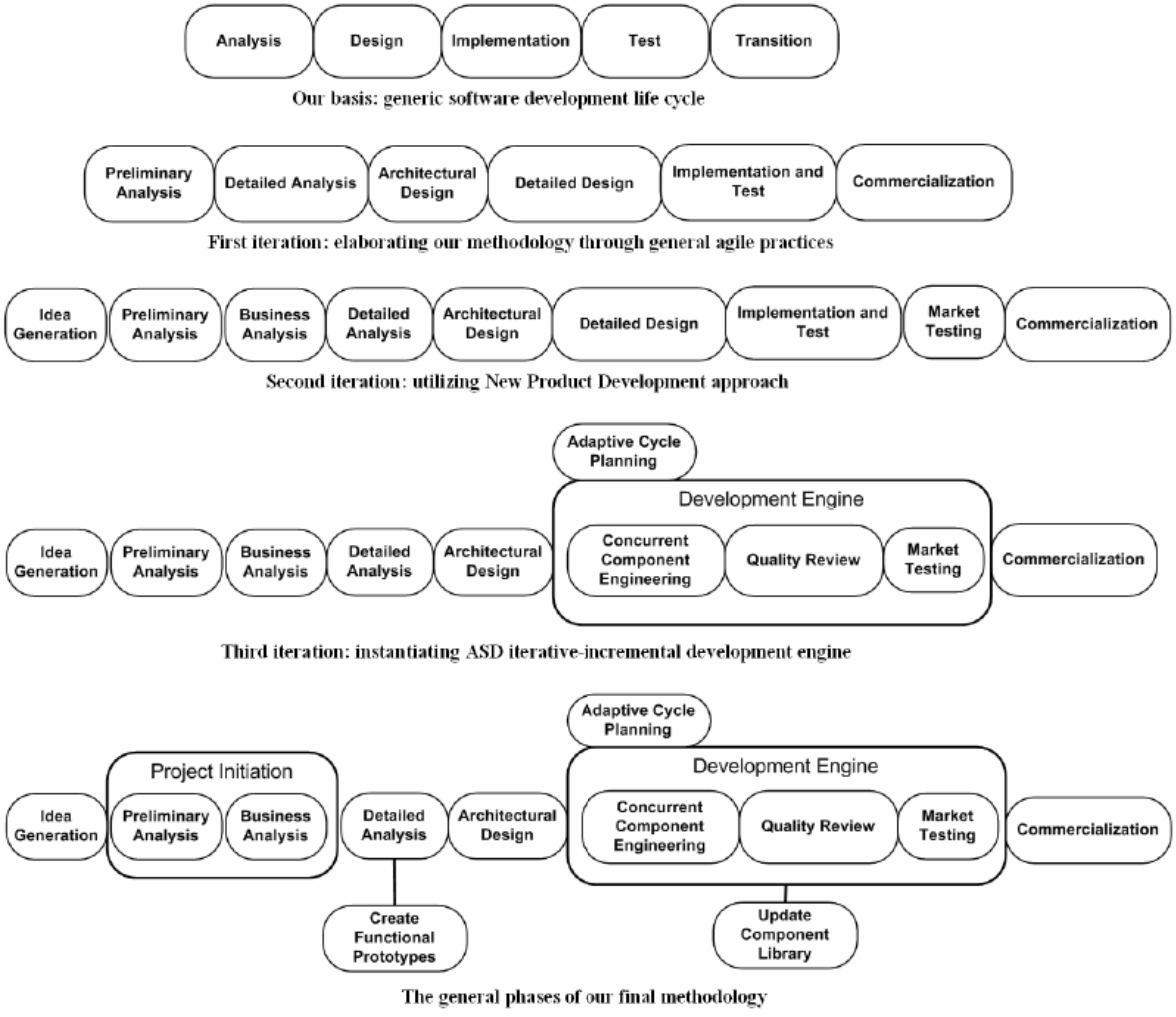

Figura 4.Evolución metodología ágil durante su desarrollo. Fuente: (Rahimian y Ramsin, 2008)

3. Marco teórico

En cuanto a la fundamentación teórica, en el transcurso del presente trabajo veremos cómo se definen y se estructuran las metodologías de Scrum, Kanban y Lean, tomando como referencia diversos autores. Del mismo modo, se explicará cómo el desarrollo de software genera la necesidad de organizar los proyectos para que progresen adecuadamente mientras se están elaborando con múltiples desarrolladores al mismo tiempo. No hemos de olvidar aspectos fundamentales como la priorización de tareas, las diferentes fases de desarrollo y el concepto de videojuego indie. Todo ello nos permitirá posteriormente elaborar nuestra propia metodología.

3.1 Los videojuegos indie

Según Pérez Latorre (2016), definir el concepto de videojuego indie es muy complicado y nadie tiene una respuesta clara. Esta dificultad en obtener una definición se extiende también a las personas que realizan los juegos y forman los diferentes estudios.

Los orígenes del videojuego indie se remontan a los años 80, cuando jóvenes fans estaban impulsando la industria del videojuego británico haciendo juegos desde sus habitaciones. Sin embargo, hasta el 2008 los videojuegos indie no alcanzaron su mayor éxito. Los juegos pioneros que impulsaron este fenómeno fueron: *Braid* (Number None Inc, 2008), *Castle Crashers* (The Behemoth, 2008) o *World of Goo* (2D Boy, 2008).

En este momento la fisura entre un videojuego indie y uno *mainstream* es cada vez menor. Por lo que, podríamos considerar como videojuegos indie aquellos que expresan una singularidad utilizando géneros populares tales como plataformas, *roge-likes*, o *walking simulators*.

Atendiendo a Pérez Rufí (2016), la situación de los estudios indie en España es muy relevante. Más de la mitad del total de las empresas españolas están formadas por cinco o menos empleados, dedicándose básicamente a la creación de videojuegos para Smartphone y PC. Todos ellos utilizan la distribución digital para sus productos y apuestan por el desarrollo de *IPs* propias, con modelos de negocio *pay to play.*

3.2 Etapas de desarrollo

En el desarrollo de un videojuego no se puede realizar todo al mismo tiempo y realizar cada una de las partes como si fuera el resultado final. Las mecánicas y el diseño del juego tienen que estar establecidos desde el primer momento. Sin embargo, las partículas que den el toque de calidad al juego serán lo último que se implemente.

Figura 5. Etapas del desarrollo. Fuente: Elaboración propia.

Por este motivo, habitualmente los desarrolladores dividen el proceso de creación en tres grandes partes: *Alpha*, *Beta* y *Gold*. Cada una de estas etapas se consideran como un estado específico del estado en el que el producto se encuentra. Esta manera de denominar las etapas está estandarizada en la industria y es la manera más sencilla de transmitir en qué punto del desarrollo se encuentra.

Antes de llegar a la fase *Alpha* se tiene que realizar lo que Bethke (2003) denomina llama *First playable.* En este estado el juego ya tiene que ser divertido y la *build* tiene que resultar atractiva divertida a todos los miembros del equipo y debería incitarlos a transmitir ideas para ser incluidas dentro del juego.

Alpha es la segunda *milestone* más importante durante el desarrollo del proyecto. Es en este momento cuando el equipo tiene que analizar todo el potencial que puede llegar a tener el producto final. La idea de esta *build* es que los programadores dejen de incluir nuevas características al juego y empiecen a hacerlo más eficiente, y limpien el código fuente. En este punto es necesario comparar la *build* con el documento inicial de diseño y comprobar que todas las características están implementadas.

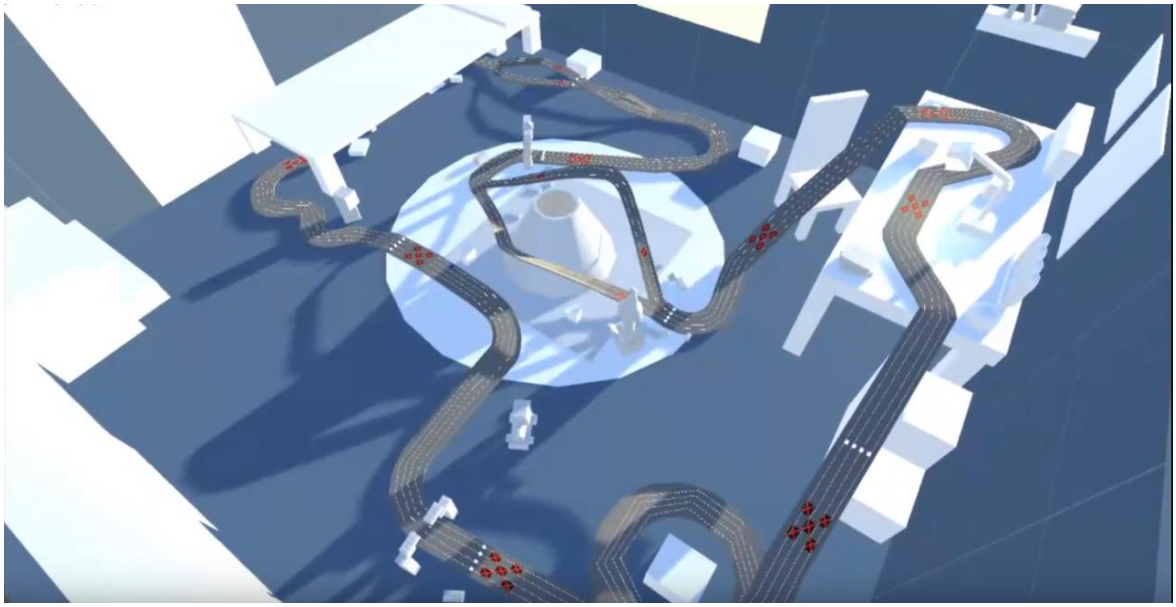

Figura 6. Escenario versión *alpha* proyecto tercer curso. Fuente: Elaboración propia.

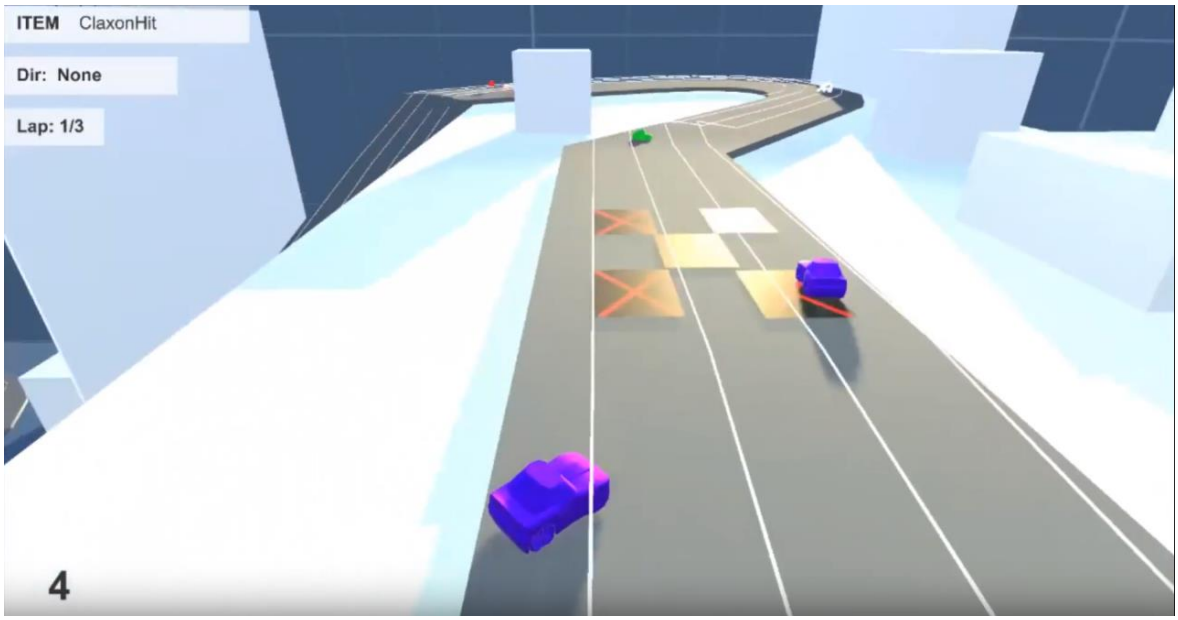

Figura 7. *Gameplay* proyecto tercer curso versión *Alpha*. Fuente: Elaboración Propia.

La siguiente fase es la *Beta*. En ella el equipo debe centrarse en hacer el juego estable, eliminar todos los *bugs* que sean posibles y balancear las mecánicas. Un método para comprobar que el producto está finalmente en Beta es que se pueda completar sin encontrar ningún tipo de *bug* no conocido. Si esto sucede, el producto debería poder ser distribuido.

Figura 8. Escenario versión *beta* proyecto tercer curso. Fuente: Elaboración Propia.

Figura 9. *Gameplay* versión *beta* proyecto tercer curso. Fuente: Elaboración propia.

Finalizada la fase la *Beta*, lo único que quedará por realizar será testar y arreglar el juego hasta hacerlo lo más robusto posible. El equipo de QA deberá volver a comprobar todos los *bugs* para corroborar que siguen solucionados. En este momento se deberá tomar una gran decisión: analizar si se va a realizar un parche para los primeros días del juego o no. Si la respuesta fuese afirmativa, se deberá empezar a planear con el equipo la nueva tarea a realizar.

Figura 10. *Gameplay* versión *gold* proyecto tercer curso. Fuente: Elaboración propia.

3.3 Las metodologías de desarrollo de software

Una de las primeras metodologías de desarrollo de software fue *Waterfall*, documentada y creada por Cusumano y Smith (1995). Se originó en los años 60 para el desarrollo de software de ordenadores gigantes del departamento de defensa e industria espacial de los Estados Unidos, así como para el desarrollo de aplicaciones con fin comercial.

Esta metodología, se utilizaba para desarrollos de software donde se conocían muy bien las necesidades del propio cliente y en aquellas situaciones donde el equipo no estaba expuesto a hacer grandes cambios en el transcurso del desarrollo. A pesar de todo ello, en esta metodología sí se contemplaba el hecho de introducir nuevas funcionalidades una vez las necesidades previas estaban cubiertas.

A continuación, y siguiendo con la idea de Cusumano y Smith (1995), en la siguiente figura se especifican detalladamente las fases de desarrollo de la metodología *Waterfall*.

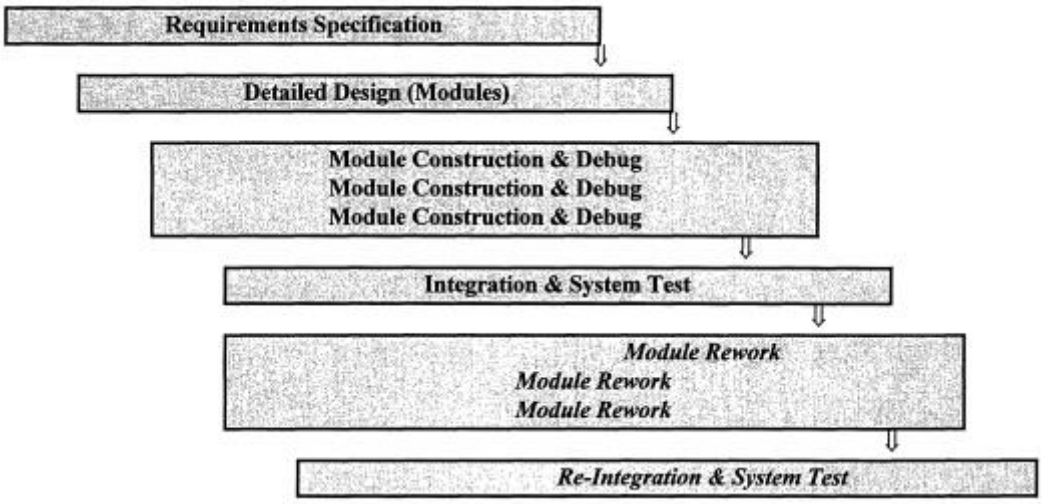

Figura 11.Esquema desarrollo con la metodología *Waterfall*. Fuente: (Cusumano y Smith, 1995).

A pesar de su gran utilidad durante los años 70 y 80, y como consecuencia de la continua expansión de los ordenadores personales y mucho más delante de los teléfonos inteligentes, esta metodología ha quedado obsoleta, al no permitir al equipo de desarrollo adaptarse a las posibles situaciones de cambio que puedan surgir durante el propio proceso de desarrollo.

3.4 Las metodologías ágiles

En 2001, un grupo de 17 personas se dieron cuenta que todos estaban utilizando diferentes enfoques para la organización del desarrollo de software. Por ello, decidieron reunirse para averiguar qué puntos tenían en común los métodos que utilizaban durante el desarrollo de sus productos, siendo el objetivo optimizar y agilizar el proceso (Agile Alliance, 2001).

Tras finalizar esta reunión consolidaron lo que hoy en día se conoce como Manifiesto Ágil. Este documento incluye las cuatro reglas básicas que toda metodología ágil debe cumplir:

"**Individuos e interacciones** sobre procesos y herramientas,

Software funcionando sobre documentación extensiva,

Colaboración con el cliente sobre negociación contractual

Respuesta ante el cambio sobre seguir un plan"

(Beck et al., 2001).

Meses más tarde añadieron doce principios que complementarían estas cuatro reglas para formalizar de esa manera lo que se calificaría como metodología ágil. A continuación, expondremos los doce principios.

3.4.1 Principios de las metodologías ágiles

En base a lo mencionado anteriormente y siguiendo con estos autores, todas las metodologías ágiles deben cumplir los siguientes doce principios.

“

1. La máxima prioridad es satisfacer al cliente a través de la entrega temprana y continua de software valioso.

2. Aceptar los requisitos cambiantes incluso tarde en el desarrollo. Las metodologías ágiles aprovechan esos cambios como ventajas competitivas delante del cliente.

3. Entrega de software funcional desde una semana a unos meses con la preferencia del mínimo tiempo.

4. La gente de negocios y el equipo desarrollador deben trabajar diariamente juntos en el proyecto.

5. Construir los productos con individuos motivados. Proporcionar el entorno y el soporte que necesitan confiando en que tendrán el trabajo listo.

6. El método más eficaz para transmitir información y comunicarse entre los desarrolladores es cara a cara.

7. El software funcional es la medida primaria del progreso.

8. Las metodologías ágiles promueven un desarrollo sostenible. El equipo debe de ser capaz de mantener un ritmo constante de manera indefinida.

9. La continua atención a la excelencia técnica y el buen diseño incrementan la agilidad.

10. La simplicidad es esencial.

11. Las mejores arquitecturas, requisitos y diseños nacen de un equipo auto-organizado.

12. En intervalos regulares el equipo refleja su mayor eficiencia, modifica y ajusta el comportamiento de manera acorde.

” (Agile Alliance, 2001).

Estos doce principios mencionados reflejan cuáles deben de ser las prácticas más relevantes para maximizar la potencia de las metodologías ágiles.

3.5 Scrum

Según los autores de *"Métodos ágiles. Scrum, Kanban, Lean."* (Gómez, García, y Dedo, 2017) los primeros en utilizar el término Scrum en el ámbito del desarrollo fueron Hirotaka Takeduchi e Ikujiro Nonaka en el año 1986, utilizándolo para describir una nueva forma de producción más acelerada, por la necesidad de ser cada vez más competitivo en cualquier mercado.

Posteriormente, Keith (2010) hace referencia a Ken Schwaber y Mike Beeblde, ambos coautores del manifiesto ágil quienes escribieron un libro en 2002 sobre Scrum, que estaba enfocado en el desarrollo de software. La publicación de este libro fue la que finalmente acabó popularizando esta herramienta, permitiendo una mayor eficacia y eficiencia a nivel organizativo, aumentando la productividad y éxito dentro de la empresa.

3.5.1 Puntos clave

Atendiendo a Gómez, García y Dedo (2017), los puntos clave de la metodología Scrum son cuatro siendo imprescindibles para aplicar correctamente la metodología y no fracasar en el proceso de creación del producto.

Inspección y adaptación. En este punto se trata la estructura de tiempos que tiene Scrum. Esta metodología se divide en *sprints* que se realizan entre una 1 y 4 cuatro semanas, dependiendo de las tareas a realizar y de cómo el equipo se organiza. En cada uno de estos *sprints* tiene que haber un producto entregable y funcional. Tras cada una de las entregas, el equipo se reúne para poder analizar qué se ha hecho y cómo para poder optimizar el siguiente sprint.

Auto-organización y colaboración. Scrum apuesta por un sistema donde el equipo es el que se gestiona. De esta manera, la libertad que los individuos obtienen hace que su responsabilidad y su compromiso aumenten. Asimismo, se promueve la colaboración y el espíritu de equipo para lograr el objetivo final.

Priorizar. Este punto es crucial, ya que si los diferentes elementos a desarrollar no están priorizados correctamente es muy probable que se acabe derrochando mucho tiempo y dinero.

Mantener un latido. Mantener un ritmo constante es fundamental para que el desarrollo continúe avanzando sin sobresaltos. Una buena organización tanto de los *sprints* como del ritmo diario es muy importante, para poder saber cuál es el volumen de trabajo que el equipo puede soportar. Es muy favorable tener fechas clave durante la creación del producto y que estas sean muy claras dentro del equipo.

3.5.2 Organización y uso

Scrum es la metodología más completa. Combinando diversos elementos, logrando una gran versatilidad en su aplicación. Por ello es la más utilizada en el desarrollo de software. Se divide en tres bloques diferentes: los roles dentro del equipo de desarrollo, la gestión de las tareas y la comunicación entre el equipo.

Todos estos elementos involucrados en la realización de la metodología son claves para poder lograr un resultado óptimo en la ejecución (Keith, 2010). A continuación se muestran todos estos elementos y se describen sus funciones en el proyecto.

Roles dentro del equipo de desarrollo.

Los roles son tres: *Scrum master*, *Product owner* y el equipo de desarrollo. Estos roles están muy bien diferenciados y cada uno de ellos tiene una tarea muy marcada. Incluso dependiendo del rol, éste deberá estar presente en unas reuniones u otras.

El *Scrum master* es el responsable de dar soporte a todo el equipo sobre Scrum. Ha de asegurarse que todo el personal conozca y siga las pautas que marca Scrum y facilitar la resolución de los problemas que puedan surgir durante el desarrollo, aportando diferentes herramientas al equipo para ello.

El *Product owner* es el encargado de la visión de negocio. Su función es conseguir obtener el máximo ROI (Retorno de inversión) estableciendo y priorizando todas las características necesarias para ello en el *product backlog*.

El equipo de desarrollo es cualquier persona que esté implicada directamente en el desarrollo del producto. En el caso del desarrollo de videojuegos serían los programadores, artistas, diseñadores, guionistas, QA testers, …

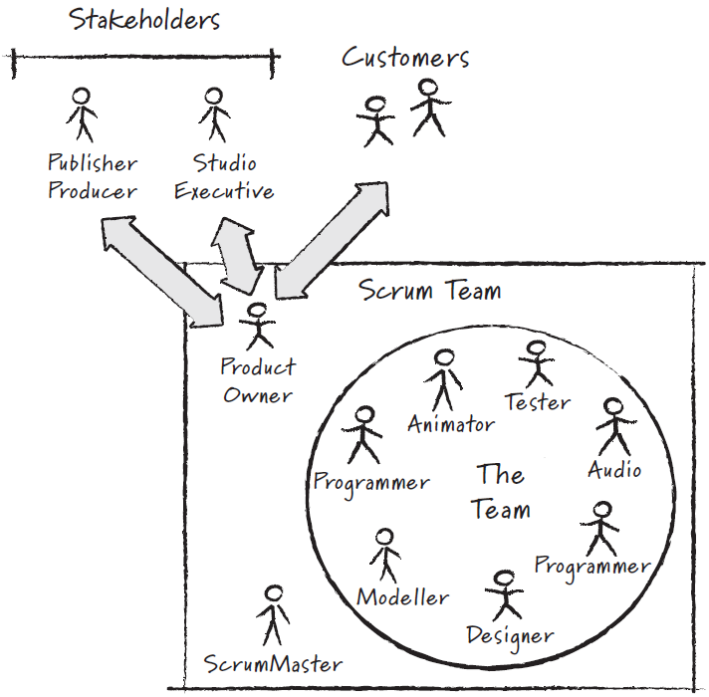

Figura 12. Esquema de los roles implicados dentro de Scrum. Fuente: (Keith, 2010).

Herramientas de gestión

Las herramientas de gestión utilizadas para controlar la situación del desarrollo, conocer las tareas que se están realizando y cuáles deberían realizarse son el *Product backlog*, el *sprint Backlog* y el *Burndown chart*.

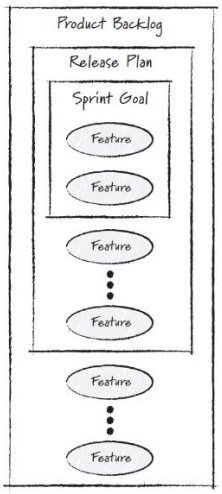

Figura 13. Esquema de las herramientas de gestión de Scrum. Fuente: (Keith, 2010).

El *product backlog*, contiene todas las características que el cliente demanda del producto, previamente clasificadas y priorizadas por el *Product owner* que es el encargado de su gestión. Las tareas aquí están redactadas en un lenguaje comprensible entendible para todo el mundo, sin ningún tipo de tecnicismo.

El *sprint backlog* almacena todas las tareas que han de realizarse en ese sprint. Estas tareas están separadas en los pasos necesarios para cumplir los requisitos finales contenidos en el *product backlog*. Estas tareas las gestiona el equipo y por ello sí que incluyen tecnicismos para describir con la mayor fidelidad posible estos requisitos.

El *Burndown chart*, es el lugar donde se pueden visualizar todas las tareas que están siendo realizadas por el equipo o faltan por iniciar. Normalmente existen dos clasificaciones: las que están relacionadas con el sprint actual y las que relacionan el proyecto en su totalidad.

Finalmente, la comunicación entre el equipo se realizan el *Sprint planning*, el *daily meeting*, el *Sprint review* y el *Sprint retrospective*. Son reuniones que Scrum contempla para que todos los pasos sean conocidos por todas las partes implicadas en el proyecto y que de esta manera surjan los mínimos errores posibles durante la comunicación.

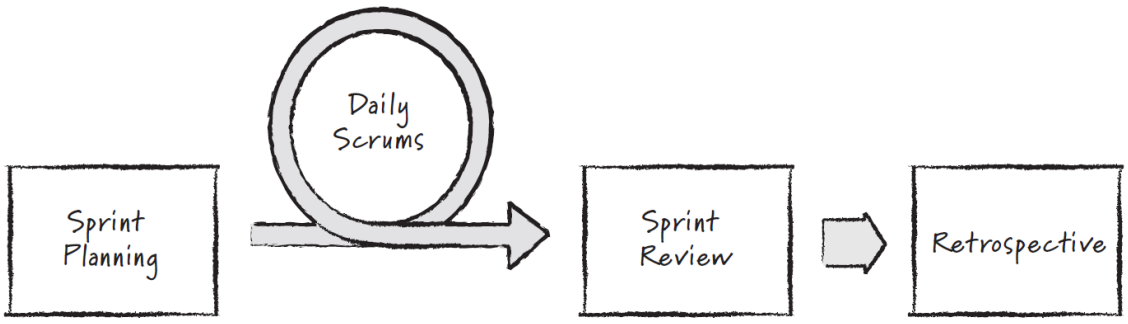

Figura 14. Esquema de las herramientas de comunicación de Scrum. Fuente: (Keith, 2010).

Reuniones entre los diferentes roles

El *sprint planning* tiene un objetivo claro y es planificar todas las características que el sprint va a tener. En esta reunión el equipo decide qué tareas el *product backlog* se puede comprometer a tener finalizadas al final del sprint y también se dividen las tareas seleccionando qué individuo se encargará de realizarlas.

El *daily meeting* o *daily Scrum* es una pequeña reunión diaria que el equipo realiza para ver cómo fue el día anterior, en qué estado está su tarea y si está teniendo algún problema para

llevarla a cabo. Esta reunión tiene que ser espontánea y no debe de durar más de 10 o 15 minutos.

En el *sprint review* están presentes tanto el *product owner* como los *managers* de sección y se analizará el sprint realizado. El objetivo de esta reunión será estudiar y definir todo lo que hay que mejorar para que el equipo rinda mejor en el siguiente sprint.

El *sprint retrospective* se realiza entre los miembros del equipo de desarrollo. Los desarrolladores deberán encontrar cómo mejorar el proceso con el que se han realizado las tareas con el fin de corregir cualquier error en el proceso y así poder evitar errores durante el sprint.

3.6 Kanban

Los autores Anderson y Carmichael (2016) hablan sobre el término Kanban. Es una palabra japonesa que significa señal o tarjeta de señalización. Esta palabra empezó a ser utilizada en el ámbito de la gestión en la década de los 60, cuando Toyota nombró al sistema que había desarrollado para limitar el flujo de trabajo que se realizaba en el mismo momento "sistemas Kanban". Este es el origen de la metodología Kanban que se utiliza en el desarrollo de software.

En la página web donde se encuentra toda la información relativa a las metodologías ágiles (Agile Alliance, 2001) se explica como esta metodología es aplicable en cualquier ámbito de trabajo, no solo en el desarrollo de software, y gracias a su aplicación tan visual, es muy útil en los casos donde el trabajo surge de manera imprevista, ya que es muy fácil de implementar.

3.6.1 Puntos clave

De nuevo los puntos clave de esta metodología (Gómez, García, y Dedo, 2017) son cuatro, siendo imprescindibles para determinar si dicha metodología se está aplicando de forma correcta o no. Veamos en detalle esos cuatro puntos:

Visualizar el flujo de todo el trabajo: Se diseña un panel organizado en columnas en el que ha de estar visible todo el ciclo de desarrollo de las partes del producto. Cada una de las tarjetas que aparecen en las columnas representa una actividad diferente. El panel ha de ser visible para todo el equipo y resultar accesible para realizar cualquier cambio.

Dividir el trabajo en ítems pequeños: Las diferentes tareas tienen que estar divididas en pequeñas acciones y han de mantener una magnitud similar. De esta manera resulta muy visual la concepción del tiempo que una tarea conlleva.

Limitar el trabajo en curso: Se tienen que limitar el número de tareas por columna para así evitar cuellos de botella, lo que permitirá que si se detecta un problema durante el desarrollo se puede solventar mucho más rápido con la colaboración del conjunto del equipo.

Medir el tiempo empleado en completar un ciclo completo: Es imprescindible calcular el tiempo que se requiere para completar la creación de una de las partes del producto ya que eso nos dará una idea de la capacidad que tendrá el equipo para desarrollar las siguientes actividades.

3.6.2 Organización y uso

Para llevar a cabo todos los puntos clave mencionados anteriormente los Anderson y Carmichael (2016) explican cómo la metodología Kanban utiliza un panel donde los trabajadores organizan todas las tareas y estas son visibles para todo el equipo, siendo ésta la herramienta principal de la metodología.

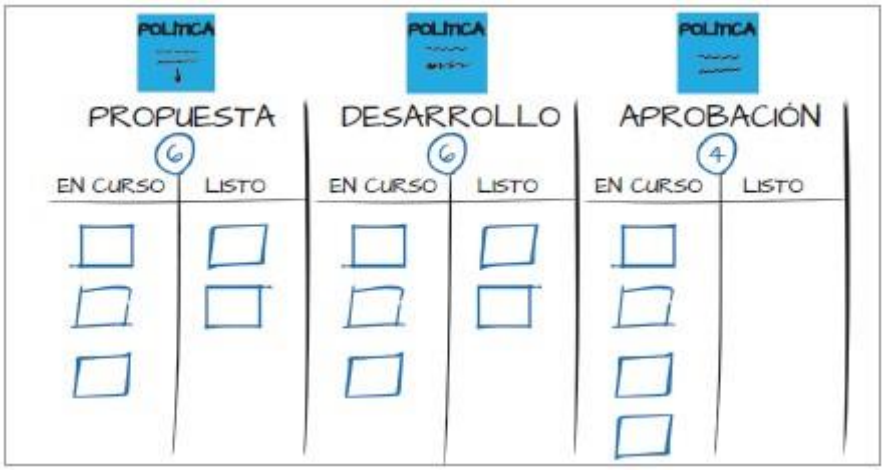

Figura 15. Panel Kanban con sus diferentes elementos. Fuente: (Anderson y Carmichael, 2016).

En la figura anterior podemos ver cuáles serán los diferentes puntos importantes a tener en cuenta cuando se integre esta metodología en un desarrollo.

Las fases del desarrollo están divididas en las diferentes columnas, cada una de las columnas contiene las tareas que están realizando en ese momento cada una de las diferentes partes. Cada una de estas columnas, en Kanban, son denominadas flujo.

Cada uno de los flujos que están representados en la tabla tienen definida una política exclusiva. Las políticas definen las normas y restricciones que ese flujo va a conllevar y han de cumplirse siempre, pudiendo ser modificadas en cualquier momento para adaptarlas a las necesidades puntuales de cada flujo.

Dentro de cada flujo, se especifican el número de tareas máximas que una columna puede soportar. De esta manera podemos observar en cualquier momento si en alguna de las secciones del desarrollo se presenta un cuello de botella que debamos solucionar, para evitar problemas durante la creación del proyecto. Esta limitación de tareas en el flujo es adaptativa y el equipo es quien decidirá cuántas son las que puede sostener a la vez. Como en las políticas, este tamaño puede variar dependiendo de las necesidades del equipo o de las características del desarrollo.

Cada uno de los flujos se divide en dos: una parte del flujo representa las tareas que se están llevando a cabo en ese momento y el otro es el apartado de las tareas que se han finalizado y que, si fuera necesario, están a la espera de pasar al siguiente flujo.

3.7 Lean Software Development

Esta metodología tiene su origen (Gómez, García, y Dedo, 2017) como lo tiene Kanban, en las fábricas japonesas del fabricante de automóviles Toyota y de cómo utilizaban Lean dentro de su filosofía de creación. El origen como tal del uso de Lean dentro del ámbito de desarrollo de software surge de los autores Mary y Tom Popendieck, que fueron los primeros en nombrar una serie de principios y técnicas que procedían de Lean.

3.7.1 Puntos clave

Esta vez son 7 los puntos clave que se enumeran como imprescindibles para poder llevar a cabo un buen uso de la metodología durante un desarrollo (Gómez, García y Dedo, 2017).

Eliminar el desperdicio: Este principio busca eliminar todo aquello que no sea valioso para el desarrollo, se ha de mantener el foco en lo esencial e intentar evitar todo aquello que no sea imprescindible en el proyecto final. También se ha de evitar la desinformación durante el proyecto y las interrupciones durante el trabajo.

Optimizar todo: Siempre pensar en el aspecto de manera global ya que cualquier pequeña optimización en uno solo de los apartados puede afectar de manera negativa a todo el conjunto.

Calidad integrada: Se tiene que buscar siempre un alto grado de calidad desde el primer momento, evitando cualquier tipo de mala práctica desde el inicio del desarrollo y realizar pruebas constantes para encontrar los defectos desde el momento inicial.

Aprender constantemente: Durante el desarrollo se tiene que ser abierto al cambio y concebirlo de forma normal. Se ha de entender en todo momento como ventaja para mejorar a medida que se avanza en el desarrollo.

Reaccionar rápido: Es la mayor de las ventajas, implementar rápidamente y ser capaz de adaptarse en todo momento a las necesidades requeridas durante el proceso de creación.

Mejora continua: Es imprescindible buscar la mejora constante, no solo en el proceso de creación sino también en el conjunto de personas que lo hacen posible. Así se mejorará el producto actual y se arraigará el sistema para que en el futuro funcione mejor.

Cuidar al equipo de trabajo: El equipo tiene que estar motivado. Esto se conseguirá dotando de autonomía a cada uno de los miembros y otorgándoles la posibilidad de aprender y mejorar constantemente para que sientan que su labor es valiosa en todo momento.

3.7.2 Organización y uso

Lean software development no tiene herramientas o reuniones marcadas como las otras dos metodologías disponen. Por ello, Lean es muy usado junto a Kanban, aportando Lean los principios fundamentales de la metodología y logrando con el uso de Kanban la gestión visual de las tareas.

Lean es considerada más una filosofía de trabajo que una metodología ágil como tal, pero se ha considerado muy importante para este proyecto ya que incluye valores que la hacen muy relevante para incluirla en la metodología que más adelante se conceptualizará.

Algunos expertos (Gómez, García, y Dedo, 2017) realizan durante el uso de la metodología Lean métricas de tiempo de producción. Estas métricas consisten en recoger todos los datos referentes al tiempo de duración de cada una de las acciones realizadas. De la recopilación de datos se obtiene un gran número de información con la que se puede trabajar para ajustar más los tiempos en la siguiente fase del proceso, consiguiendo así ser más realista a la hora de estimar los plazos de entrega.

3.8 Estimación

Una técnica muy eficiente para estimar cuánto tiempo le va a costar al equipo realizar una tarea es la del *planning poker*.

Atendiendo al proceso que Cohn (2006) especifica, se empieza creando un conjunto de cartas que mostraran una serie de números, por ejemplo 0,1,2,3,4,5,6,7,15,30 y 100. Los números indicarán el grado de complejidad que se puede asignar a cada tarea. Las cartas se deben de preparar antes de que la sesión de *poker* comience y el número tiene que ser tan grande como sea necesario para que todas las personas en la sala sean capaces de verlo.

Figura 16. Representación de *planning poker*. Fuente: (Cohn, 2006).

El *planning poker* incluye a todos los miembros que forman el equipo de desarrollo, programadores, artistas, diseñadores, … Todos ellos recibirán un set de cartas cuando la sesión empiece. Por otro lado, debe de haber un rol que sea el del moderador, quien se encargará de describir cada una de las tareas que se debe de estimar y llevará el control de la sesión en todo momento. En un proyecto que se esté realizando de manera ágil los miembros de la sesión no deberían de superar las 10 personas. Si eso ocurriera, lo más recomendable sería dividir en dos el grupo para reducir el número de personas.

Cuando la sesión empieza, el moderador lee la descripción de la tarea y los miembros del equipo pueden realizar todas las preguntas que sean necesarias. Una vez resueltas todas las dudas, los miembros del equipo seleccionarán el número de la carta que crean pertinente en función del grado de dificultad que cada uno considere que presente esa tarea y la mostrarán todos a la vez.

En este momento es muy probable que los números difieran mucho unos de otros, lo que es positivo ya que nos ayudará más tarde a concretar mejor el valor real a asignar.

En este momento, se pregunta a la persona que haya clasificado la tarea con el número más alto por qué lo ha creído así y se le pedirá que exponga sus argumentos. Lo mismo se hará con la persona que ha asignado un número menor. En el momento de las argumentaciones el moderador tiene que evitar cualquier tipo de ataque y recordar que de esta manera se conseguirá un resultado mejor de estimación. Una vez escuchadas las dos argumentaciones, se vuelve a realizar el proceso de selección de valor.

Si los valores siguen difiriendo mucho, el moderador volverá a preguntar a las personas que hayan clasificado con el número más alto y el más bajo y nuevamente se volverá realizar el proceso de selección del valor. El proceso finaliza en el momento que la mayoría de los números del equipo convergen y se considerará ése número como valor final de la tarea. Aunque parezca un proceso largo, no suele durar más de tres rondas.

El autor indica que hay tres motivos de peso por los cuales el *planning poker* funciona tan bien en los equipos para priorizar las tareas.

La primera, es que se juntan a muchos expertos de diferentes áreas a deliberar y eso hace que la estimación sea realmente precisa, ya que se está valorando la experiencia de cada una de ellos.

La segunda, hace referencia al diálogo durante el *planning*, y es que la justificación de cada uno de los miembros del equipo hace que cada vez que se realice el proceso se llegue a un valor cada vez más exacto.

Y el tercer motivo es que el proceso resulta ameno y divertido.

4. Objetivos de la investigación

Este proyecto tiene como meta lograr los dos objetivos primarios y los dos objetivos secundarios que a continuación se mostrarán. Estos objetivos son el propósito y la motivación para la realización del proyecto. Los objetivos se enmarcan en objetivos sobre la realización del proyecto y otros sobre la ejecución de este.

Objetivos sobre la realización del proyecto.

- El objetivo primario en la realización del proyecto es la conceptualización de una metodología ágil que sea más óptima para aplicarla en el desarrollo de estudios indie.

Este objetivo hace referencia al motivo por el cual se está realizando este el proyecto. Las metodologías actuales demandan muchos requisitos para ser usadas. Por ello muchas empresas, sobre todo las indie, tienen que realizar cambios para adaptarlas a sus necesidades. Cada equipo y estudio son diferentes por lo que el objetivo ha de ser crear una metodología más aplicable en cualquier tipo de situación sin necesidad de ser adaptada.

    - El objetivo secundario sería conocer con más detalle las diferentes metodologías ágiles que se utilizan actualmente en el desarrollo de software.

Con este objetivo se espera obtener un alto conocimiento sobre las metodologías ágiles actuales, dónde surgieron y qué les ha llevado a formar un punto clave en el desarrollo de cualquier producto, pero más especialmente en los videojuegos. Por otro lado, también se espera averiguar por qué aún implementándose de manera tan diferente en el desarrollo, las tres metodologías seleccionadas son consideradas ágiles.

Objetivos sobre la ejecución del proyecto.

- Corroborar mediante diferentes entrevistas a profesionales del sector que la metodología ideada es útil para los diferentes estudios, sería el objetivo primario de la fase de ejecución.

Mediante el uso de entrevistas se pretende obtener la información necesaria para poder aceptar o denegar la metodología planteada y, en su caso, modificarla para obtener una segunda versión que satisfaga las necesidades de los estudios a los que dirige.

- Como objetivo secundario de la fase de ejecución, nos fijamos conocer más de cerca la implementación real de las metodologías en los estudios de videojuegos en España.

De nuevo mediante entrevistas se pretende averiguar cómo los estudios aplican en el desarrollo las diferentes metodologías y si se aplican como los autores las describen, o bien deben realizarse modificaciones para adaptarlas al equipo de trabajo.

5. Diseño metodológico y cronograma

Para poder organizar el desarrollo del proyecto y estructurar todos los puntos que éste contiene, se mostrarán a continuación el orden de creación (diseño del método) y el proceso necesario para lograr los objetivos definidos de este trabajo (cronograma).

5.1 Diseño metodológico

Para lograr el objetivo principal, conceptualizar una metodología ágil más óptima para los estudios indie, seguidamente se explican todos los elementos y procesos que se van a llevar a cabo para conseguir que este objetivo se logre finalmente. Algunos de los elementos que se utilizarán son la realización de esquemas, la ejecución de entrevistas a profesionales del sector o documentos que se usarán para mostrar las respuestas.

5.1.1 Conceptualización metodología ágil

Las metodologías ágiles no se crearon siguiendo un patrón específico ya que surgieron de una necesidad de organización cuando se estaba desarrollando el producto. El ejemplo más claro está en las fábricas japonesas de Toyota dónde después de la guerra, con la expansión de los mercados en los años 60 la demanda de automóviles era cada vez más grande por parte de los japoneses y los demás países del mundo.

Sin embargo, actualmente existen diversas metodologías que se han expuesto a lo largo del proyecto. De tal manera se analizarán estas tres metodologías escogidas con el objetivo de extraer toda la información posible, con los puntos fuertes de cada una de ellas para de esta manera lograr entender y poder aplicar estos conocimientos a la metodología que se propondrá.

Para cumplir el objetivo principal de este trabajo se conceptualizará una metodología con todo el conocimiento teórico adquirido, se realizarán diversos esquemas que expliquen cual será el proceso para llevar a cabo correctamente el uso de la metodología propuesta, y se redactarán las diferentes normas y roles que serán necesarios para lograr una organización óptima.

5.1.2 Validación de la metodología ágil

Otro punto en el cuál se trabajará es en la preparación de una entrevista semi-estructurada, que se compone de preguntas que se realizarán o no dependiendo de las respuestas de cada entrevistado. Más adelante se realizará a tres productores en la industria del videojuego para conocer sus opiniones, sobre todo lo referente al uso que ellos hacen de estas herramientas en su día a día. También se harán preguntas sobre la metodología que se conceptualizará para conocer la opinión de estos expertos.

Cuando la entrevista esté formalizada se citarán a los diferentes entrevistados ya sea de manera presencial o por videoconferencia, para realizarles las preguntas que se les plantearán. Las entrevistas estarán gravadas en audio para que estén disponibles una vez acabado el proyecto y también para que puedan ser transcritas y tener las respuestas en formato papel.

Para analizar las entrevistas se seguirá una metodología concreta, primero se generará un documento donde se unificarán todas las respuestas dependiendo de la pregunta contestada, de esta manera se podrá visualizar mejor toda la información respectiva a esa pregunta.

Seguidamente se creará un segundo documento donde se anotarán diferentes puntos clave que se hayan obtenido de leer las respuestas dadas por los entrevistado y se relacionarán esos puntos clave con la metodología que se está realizando. De esta manera se comprobarán los puntos fuertes y débiles de la metodología.

Con ese análisis de los puntos clave, se realizarán los cambios en la metodología para generar una segunda versión más adaptada a las necesidades reales de los estudios. Cuando se estén justificando estos cambios se utilizarán citas explícitas de los entrevistados para convencer de la importancia de los mismos.

Finalmente, cuando la metodología haya sido modificada, se concluirá sobre la misma con el fin corroborar si el resultado final es óptimo para cumplir las necesidades que un estudio indie puede tener durante la creación de un proyecto.

5.2 Cronograma

En este apartado se muestra el proceso a lo largo del tiempo, indicando las fechas de cada uno de los hitos que se deben lograr. Este esquema con las fechas se complementa con un gráfico de Gantt en el cual se pueden visualizar cada una de las entregas y la duración de cada uno de los procesos que anteriormente en el diseño metodológico se han expuesto.

1 – 21 octubre: Preparación de la propuesta.

22 octubre – 11 febrero: Creación del pre-proyecto, análisis de las diferentes metodologías ágiles seleccionadas.

11 febrero: Entrega pre-proyecto.

11 febrero – 23 abril: Corrección y mejora del pre-proyecto, preparación memoria intermedia, primera conceptualización de la metodología, preparación de las entrevistas y contacto con los entrevistados.

24 abril: Entrega memoria intermedia.

25 abril – 12 junio: Corrección y mejora de la memoria intermedia, realización de las entrevistas, revisión final de la metodología propuesta y conclusión del resultado.

12 – 14 junio: Entrega memoria final.

15 – 24 junio: Preparación defensa oral.

25 junio – 12 Julio: Defensa del trabajo final de grado.

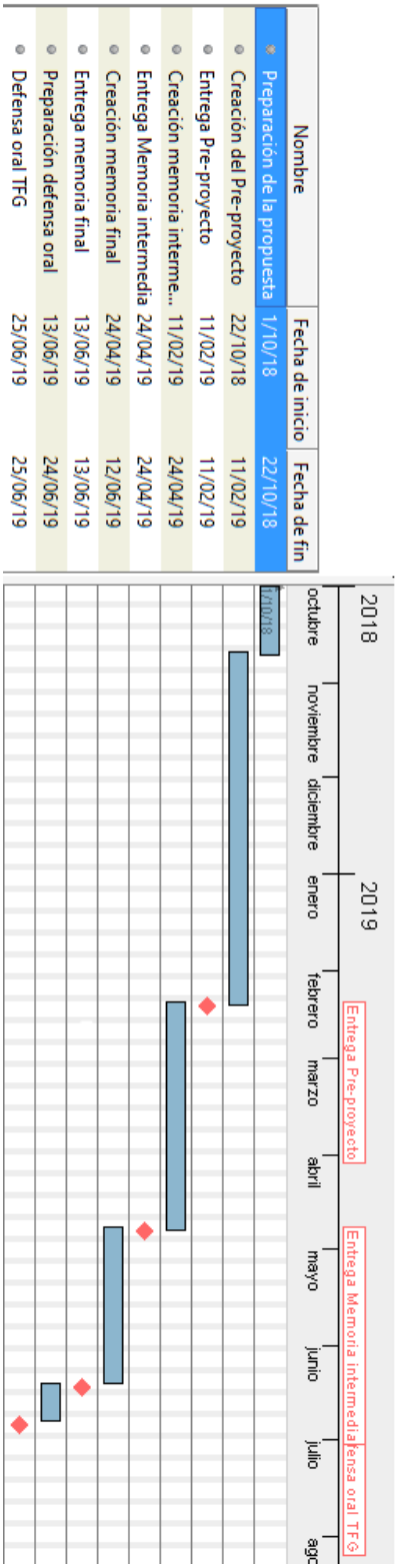

Figura 17. Cronograma de gantt. Fuente: Elaboración propia.

6. Análisis y resultados

Para cumplir todos los puntos expuestos en el diseño de la metodología de trabajo, a continuación, se muestran todos los puntos estudiados y toda la información que se ha recopilado durante la creación del proyecto, para poder lograr todos los objetivos que se ha considerado para este trabajo.

6.1 Análisis puntos fuertes para el contexto indie

Después de analizar cada una de las metodologías estudiadas y comprobar uno a uno los puntos fuertes de cada una, se han escogido nueve puntos que se creen imprescindibles para que la metodología que se está creando se adapte de manera óptima a los estudios indie.

Los puntos se han seleccionado de Scrum, Kanban y Lean. De Scrum escogemos la inspección y la adaptación, la auto-organización y colaboración, y priorizar. De Kanban seleccionamos visualizar el flujo de trabajo y dividir el trabajo en ítems pequeños. Finalmente, de Lean eliminar el desperdicio, calidad, aprender constantemente y cuidar el equipo de trabajo.

En este apartado se justifica el motivo de la elección de éstos y porqué se consideran adecuados para una metodología enfocada a estudios indie.

De la metodología Scrum:

1.La inspección y la adaptación

- Saber adaptarse de manera rápida y conocer en todo momento los cambios en el mercado es un punto esencial para cualquier estudio, pero aún más para uno donde los recursos son aún más limitados. Para los estudios indie este es un punto que les puede permitir salir al mercado con un producto que sea mucho más afín a las necesidades del cliente.

2.Auto-organización y colaboración

- Dar libertad a los desarrolladores y que colaboren entre ellos es imprescindible y muy beneficioso para el proyecto ya que la motivación y la implicación aumentan. Es muy conveniente este punto, ya que al ser equipos pequeños, hace que los desarrolladores se sientan una parte indispensable del proyecto y logrará que éste avance más rápido y con mayor calidad.

3.Priorizar

- Establecer las tareas que van a convertir el producto en un MVP (*Minimum viable product*). El producto tiene que mutar siempre teniendo un valor por sí mismo. Por ello priorizar qué tareas se van a realizar primero puede lograr el éxito en un estudio pequeño donde lo más importante es el resultado final del producto.

De Kanban:

4.Visualizar el flujo de trabajo

- Este punto se ha considerado esencial ya que se entiende como extremadamente necesario para el equipo. Que éste conozca en qué punto está el proyecto, hasta dónde se ha realizado y qué falta por realizar. Para el equipo, ver el panel con los avances consigue que sean conscientes de la velocidad a la que van y hasta donde han llegado. La visualización de todas las tareas finalizadas es esencial para mantener el pulso de trabajo.

5.Dividir el trabajo en ítems pequeños

- La visualización del proyecto en pequeñas tareas hace el trabajo mucho más sencillo y manejable para cualquier estudio. Por ello se ha considerado conocimiento de las tareas como pequeños logros, haciendo que el equipo se mantenga motivado durante más tiempo y que la línea de trabajo sea más clara para lograr el objetivo final.

De Lean Software Development:

6.Eliminar el desperdicio

- Evitar desarrollar elementos que finalmente no se vayan a incluir en el proyecto puede implicar al estudio grandes pérdidas, elemento que no se puede permitir un estudio indie. Por esto es muy importante conocer que tareas van a hacer avanzar realmente el proyecto y no perder el tiempo en pequeños detalles o en aspectos que no son realmente importantes para el producto final.

7.Calidad

- Realizar cada una de las tareas intentando lograr la máxima calidad posible. De esta forma se reduce la probabilidad de tener que volver a realizar la misma tarea de nuevo más adelante y finalmente conseguir un producto más pulido. Un producto de mejor calidad implica menos costes de testeo a largo plazo y mejores ventas en el momento de salida del juego.

8.Aprender constantemente

- Como el mundo de la tecnología evoluciona tan rápido es esencial que el equipo este al día. Por eso tomarse un tiempo para estudiar cómo implementar una *feature.* Conocer nuevas tecnologías o formas de implementar una característica pueden ser imprescindible para que más adelante en el desarrollo se avance de una manera mucho más rápida.

9.Cuidar el equipo de trabajo

- Ayudar al equipo a poder sobrellevar todos los problemas que el desarrollo pueda comportar o facilitarles herramientas para que puedan expresar como se sienten se ha considerado necesario. En un equipo pequeño, donde cada una de las personas es sumamente importante, hacer que éstas puedan rendir al máximo durante la mayor parte del tiempo se ha considerado esencial como punto fuerte.

6.2 Metodología híper-ágil versión 1

Después del estudio realizado y la elección de los puntos clave, considerados como esenciales para una metodología apropiada para los estudios indie, se determinan para esta propuesta tres pilares imprescindibles, los roles que las personas involucradas en el desarrollo efectuarán, las reuniones que se realizarán durante el periodo de desarrollo y por último la organización visual que las tareas tendrán en el sitio de trabajo.

6.2.1 Explicación

A continuación, se muestran todos los elementos de la metodología para poderla implementar dentro de un proyecto. Este proceso que se explica a continuación se realizará de manera recursiva durante todo el desarrollo, es decir, se utilizará para lograr una versión Gold desde el mismo prototipo.

6.2.1.1 Organización del equipo

El **equipo** constará de todos los programadores, artistas y diseñadores de los que el estudio disponga.

El ***Project manager*** es la persona encargada de gestionar el uso correcto de la metodología y conseguir que el equipo la aplique correctamente durante el desarrollo. También, tendrá que realizar la labor de mantenerse en contacto con el cliente (Publisher o mercado) para saber en todo momento cómo tiene que evolucionar el proyecto.

Por último, el ***Team manager***. Este rol lo realizará una persona que esté completamente fuera del desarrollo. El responsable se encargará de mantener al equipo con un estado de ánimo alto y de resolver cualquier incidencia de carácter personal que los miembros del equipo puedan requerir.

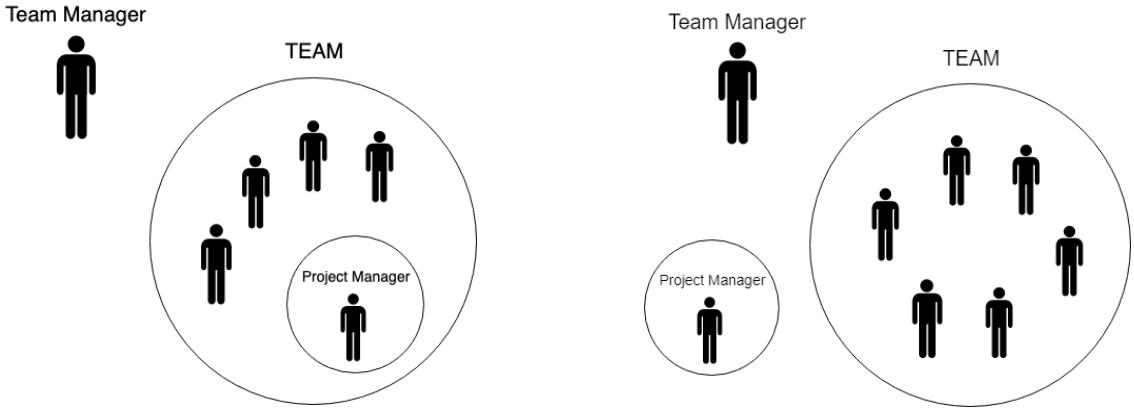

Figura 18. Ejemplos estructura personal. Fuente: Elaboración propia.

6.2.1.2 Creación de las tareas

Primero, se debe especificar cuál va ser el objetivo que se plantea como *milestone*, por ejemplo, un objetivo para una *milestone* puede ser transformar el prototipo del juego a una *alpha* jugable.

En este punto se realiza la ***Milestone planning*** (reunión) donde se generan las tareas que se van a llevar a cabo para lograr el objetivo marcado, estas tareas son las consideradas por el equipo como esenciales para lograr el objetivo de la *milestone*. Esta reunión implica tanto al equipo como al *Project manager* y no tendrá una duración concreta. Se debe de consumir todo el tiempo necesario para que las tareas a realizar sean las correctas y necesarias para lograr el objetivo.

Una tarea tiene que ser indivisible, es decir, una tarea tiene que ser la mínima unidad de trabajo de una persona. Un ejemplo sería, "Modelar el personaje principal", otra sería "Riggear el personaje principal". En ningún caso, una tarea puede ser "Hacer el personaje principal" o "Balancear daño enemigos".

De esta reunión, se obtiene un total de tareas (***Task pool***) para lograr el objetivo de la *milestone*. Con estas tareas acordadas entre el equipo y el ***Project manager*** se realiza un *poker* (método de estimación y priorización) que nos dará una puntuación del 1 al 10 que representará la dificultad estimada por el equipo.

De todas las tareas ya clasificadas se escoge el 1/3 de tareas más complicado, estas tareas representan ahora el ***Bronze Stage***. Las tareas restantes (***Task pool***) se reservan para realizarse en un futuro.

Las tareas seleccionadas, se intentará que involucren a todos los departamentos por igual para poder trabajar en paralelo y avanzar de una manera más rápida, pero no tienen por qué ser equitativas para cada uno de estos.

No se propondrá un límite de tiempo para realizar estas tareas (***Bronze Stage***), siempre se intentará obtener la mejor calidad del juego y máximo bienestar del equipo.

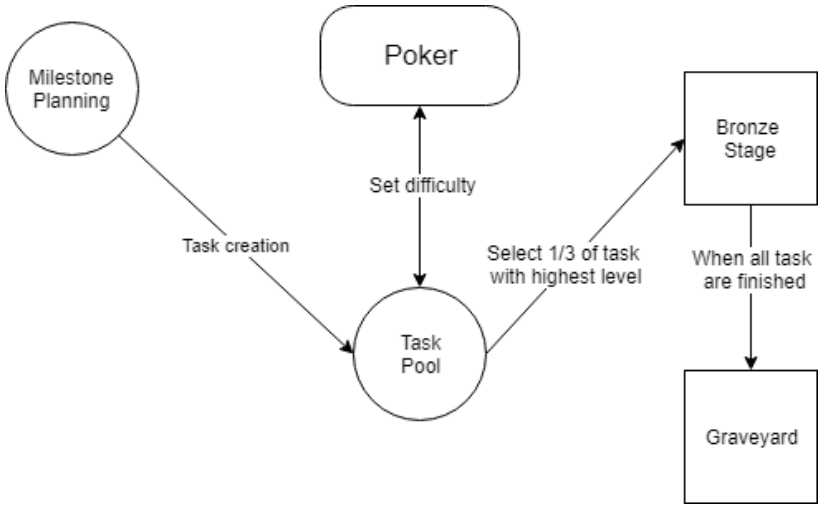

Figura 19. Flujo de trabajo en la metodología Híper-ágilv1. Fuente: Elaboración propia.

6.2.1.3 Representación de las tareas

Con las tareas ya acordadas y clasificadas, estas se deben colocar en el panel donde nos mostrará el estado de cada tarea. El panel tiene seis columnas:

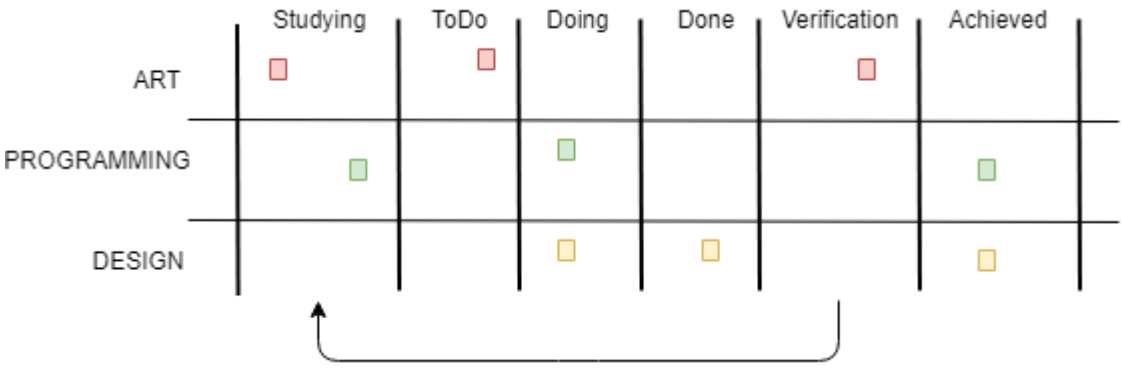

Figura 20. Panel para la metodología Híper-ágil. Fuente: Elaboración propia.

Todas estas tareas serán representadas dentro de un post-it. La información de la tarea es representada de la siguiente manera:

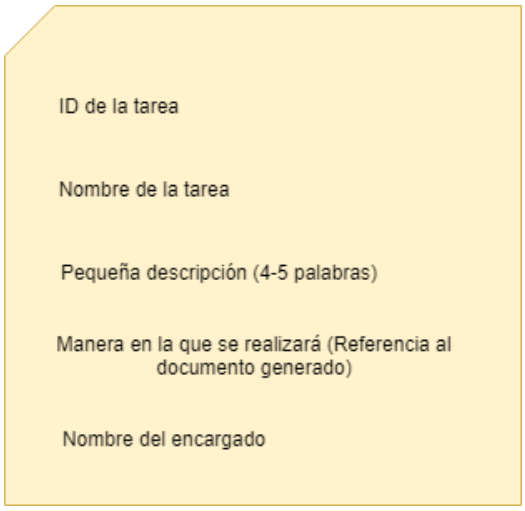

Figura 21. Representación de la tarea en post-it. Fuente: Elaboración propia.

Una vez completados todos los post-its necesarios con las tareas, se colocan en el departamento correspondiente en la columna de estudio.

6.2.1.4 Proceso de trabajo

Cuando una tarea este en ***Studiying***, se estudiará entre todos los miembros del departamento, siendo **mínimo dos personas** en un estudio si esto fuera posible. Cuando este finalice se generará un documento (máximo una página) con la información necesaria para realizar esa tarea, siempre de una manera escueta y visual para obtener la mejor vía posible para realizar esa acción.

Una vez realizado el estudio se asignará esa tarea a la persona que la realizará, que será también la encargada de ir actualizando su tarea en la tabla.

Se irán realizando las tareas necesarias para ir pasando de una columna a otra. Cada dos días se hará una pequeña reunión entre todos los miembros del equipo para ver que se está realizando, qué problemas se están encontrando y como se van a solucionar. Esta reunión se llama ***BiDaily.*** Esta se realizará lunes, miércoles y viernes para tratar de abarcar toda la semana. La duración será de aproximadamente de unos diez minutos y todos los miembros del equipo y el *Project manager* estarán presentes. En esta reunión los integrantes del equipo deberán explicar cómo llevan las tareas y que problemas están teniendo mientras las realizan.

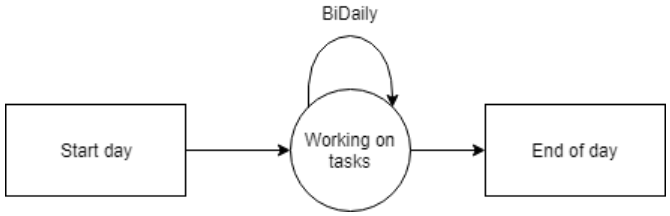

Figura 22. Flujo de trabajo diario. Fuente: Elaboración propia.

Una vez la tarea llegue a la parte de ***Verification*** de nuevo el departamento responsable realizará la verificación de manera conjunta con el *Project manager* de esa manera se asegura la mayor calidad posible dentro de todas las etapas. En este momento se determinará si se considera valida la tarea o no. Si no se determina como válida deberá volver al estudio y se tendrá que encontrar una forma mejor de implementar la *feature*.

Finalmente, si esta es aceptada pasara al estado de *Achieved*.

Una vez logradas todas las tareas se eliminarán del panel y se guardaran de manera simbólica en el ***graveyard*** (Zona donde se guarden los post-its) y se realizara una ***stage review*** (reunión) para ver cómo ha ido todo el proceso y que puede mejorarse en el próximo *Stage*. Nuevamente esta reunión involucra al equipo y el *Project manager*. Esta reunión que tendrá una duración de una hora, el equipo tiene que discutir cada uno de los puntos que considere precarios mientras se ha desarrollado el stage, de esta manera se pueden tomar medidas para que se resuelvan los problemas encontrados y se afronte con más facilidad el siguiente stage.

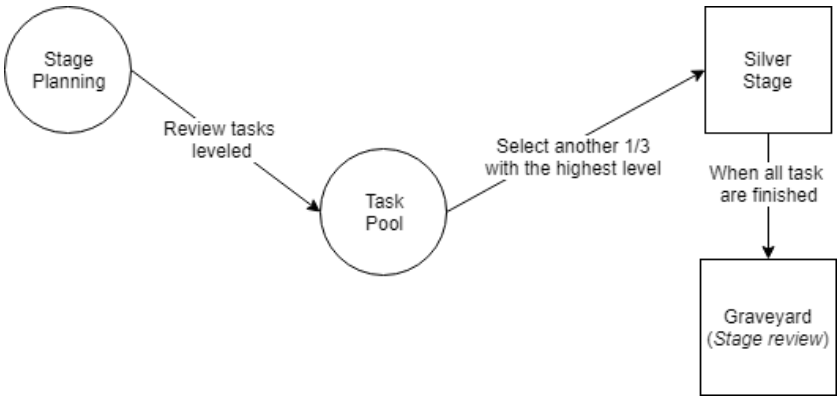

Figura 23. Flujo de trabajo *silver stage*. Fuente: Elaboración propia

Después de esto se realizará otra reunión, llamada ***stage planning*** donde de las tareas que continúan en la ***Task pool***, se escogerá el 1/3 de las tareas más difíciles de nuevo, se revisará que la dificultada sea la adecuada y se volverá a realizar el proceso para esta ***silver stage*** y finalmente se realizará para la ***gold stage***.

Finalmente, estas tareas se realizarán de manera recursiva desde el inicio al final del desarrollo del proyecto. Cada una de las *milestones* contará con las tres subdivisiones por *stage*s que son *bronze*, *silver* y *gold*.

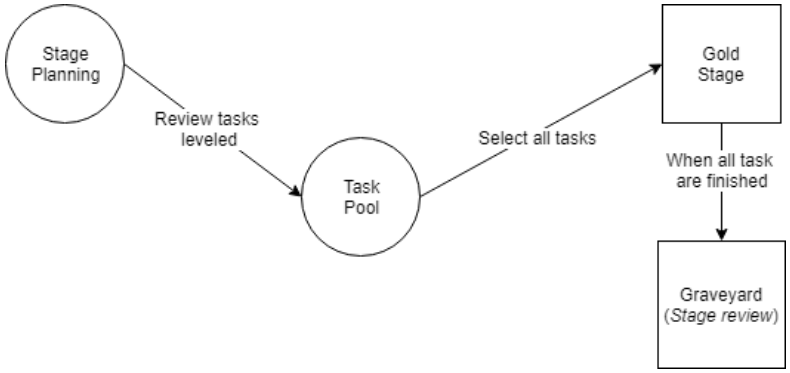

Figura 24. Flujo de trabajo *gold stage*. Fuente: Elaboración propia.

6.2.2 Justificación

A continuación, se exponen y justifican los tres pilares fundamentales que forman la metodología conceptualizada. Estos pilares son: los roles involucrados, las reuniones y la organización visual. Cada uno de estos tres apartados son los elementos expuestos en la explicación anterior.

Roles involucrados

Primero se definen unos roles dentro de la empresa que se tendrán que encargar de diversas tareas durante el tiempo de desarrollo.

El primer rol creado ha sido el *Project manager* que tendrá dos tareas al mismo tiempo durante todo el periodo de desarrollo.

Que esta figura se encargue de dos tareas a la vez cuando en otras metodologías como Scrum están separadas viene dada por la limitación que un estudio indie tiene normalmente en número de personal. Si el equipo consta de seis personas es muy difícil que dos se puedan encargar únicamente de las tareas de gestión y además fuera del equipo de desarrollo como Scrum propone.

Por ello, dependiendo del número de participantes en el desarrollo, esta persona estará dentro o fuera del equipo. La forma más óptima de implementar este rol sería en la figura del productor y estando fuera del equipo de desarrollo, para que pudiera realizar este trabajo con una calidad óptima. Si no fuera este el caso y que el equipo se compusiese de un grupo de personas tan reducido que no tuviese un productor, la persona más indicada para realizar este rol sería el diseñador, teniendo en cuenta que no puede dejar de lado su tarea propia como diseñador.

Otro rol que se ha generado es el *Team Manager*. Esta es la persona encargada de que el equipo se mantenga motivado durante todo el desarrollo, y se la ha propuesto para cumplir con el punto fuerte escogido de Lean: cuidar el equipo de trabajo. Es realmente importante que el equipo completo esté motivado y a gusto en el trabajo, ya que en un equipo pequeño si tan sólo uno de sus componentes no rinde bien, puede frenar la evolución del proyecto.

Se ha concebido como un rol independiente para evitar que esta persona pueda involucrarse en el desarrollo de manera directa. Así podrá realizar sus tareas de manera óptima y el estado del equipo no podrá alterar su conducta ante los problemas de éste.

Reuniones

Para esta metodología se han considerado cuatro reuniones con las que poder fomentar la comunicación entre los miembros del equipo de una manera ordenada. Cada una de ellas tendrá una duración diferente y estarán enfocadas a solucionar diferentes problemas. Como la metodología está pensada para equipos pequeños de estudios indie, todos los miembros del equipo y el *Project manager* estarán presentes en todas ellas.

La primera y más importante de las reuniones, la *Milestone Planning,* deberá tener a todo el equipo presente. En ella se concibe el objetivo de la milestone, qué tareas se van a realizar y cuáles son las estimaciones para cada una de ellas.

Por ello reunión se ha concebido para que todo el equipo esté involucrado en el proyecto y pueda aportar todo su conocimiento para que éste sea un éxito. Se han tenido en cuenta para esta reunión que no exista un límite de tiempo, para poder concretar de la forma más fiel posible las necesidades de la *milestone*. En esta reunión ya se concentran gran parte de los puntos fuertes seleccionados como podría ser priorizar, dividir el trabajo en ítems pequeños o eliminar el desperdicio.

Durante el desarrollo se realizarán reuniones *BiDaily*. Estas reuniones servirán para ponerse al día con las tareas que se están realizando y comprobar si las claves de la metodología se están implementando o no. La reunión se realizará un día sí y otro no para evitar pérdidas de tiempo.

La tercera reunión, llamada *Stage Planning*, es una versión reducida de la *Milestone planning*, donde se tiene que concretar todo lo necesario para afrontar el *stage* siguiente, acordando con el equipo todo aquello a realizar. De nuevo esta reunión no tiene una duración muy larga para evitar pérdidas de tiempo, pero ha de ser el necesario para de nuevo ver cómo evoluciona el equipo y si este continúa realizando correctamente la metodología.

Finalmente se realizará una *Stage review*, donde se comentarán todos aquellos problemas que el equipo pueda haber tenido durante el desarrollo y cómo se pueden afrontar las nuevas tareas para evitar estos problemas. Esta reunión sí que se puede extender un poco más llegando a las dos horas. Todo este tiempo invertido ayudará a que más tarde, en el desarrollo, si el equipo se enfrenta a un problema similar, sean mucho más rápidos con la solución, pudiendo incluso evitarlos. Esta reunión está concebida para mantener la calidad y el aprendizaje constante.

Organización visual

La metodología propuesta considera unos paneles instalados en el estudio para que todo el equipo pueda reconocer el estado de desarrollo de un vistazo, cumpliendo así con los puntos fuertes que se han extraído de Kanban.

El panel tendrá que estar en un sitio visible y con toda la información disponible para cualquier miembro del equipo y mostrará cada uno de los post-its que formarán las tareas de la *milestone*.

De esta manera se pretende mejorar la calidad global del producto y aprender constantemente obedeciendo a los tres puntos clave seleccionados de Lean y también cumplir con los puntos fuertes extraídos de Kanban: visualizar el flujo de trabajo y dividir el trabajo en ítems pequeños.

6.2.3 Simulación

El motivo de la realización de estas simulaciones es demostrar cómo se comporta la metodología en una situación real de desarrollo, por ello se ha cogido como referencia el proyecto final realizado en el tercer curso. Este videojuego consistía en un juego de scalextric donde los jugadores controlaban unos coches que recorrían una habitación infantil (Ver Figuras de 6 a 10).

En esta simulación se ha tenido en cuenta las tareas que se realizaron para pasar del prototipo a la *alpha*, estas tareas se reformularan con la estructura creada y se llevara a cabo hasta finalizar el primer *stage* de la metodología.

6.2.2.1 Organización del equipo

El equipo constará de dos artistas, un programador y dos diseñadores.

El *Project manager* en este proyecto no se contempla un *Project manager* externo al equipo por esto, uno de los diseñadores toma el rol, siendo el encargado de organizar la metodología con el equipo y el contacto con el profesorado, en esta simulación el cliente.

Por último, el *Team manager*, este rol lo realizará uno de los profesores por lo que se considera como si estuviera subcontratado ya que no está dentro de la empresa que realiza el proyecto.

De esta manera el equipo estaría formado finalmente y la constitución de este está representada en el esquema que se muestra a continuación.

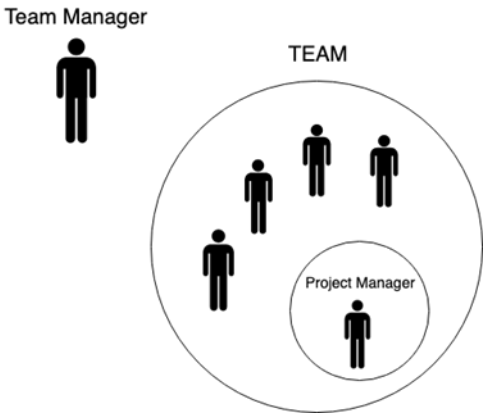

Figura 25. Representación estructura simulación versión 1. Fuente: Elaboración propia.

6.2.2.2 Creación de las tareas

Una vez el equipo está formado, se tiene que establecer un objetivo, este como se ha expuesto anteriormente es el de transformar el prototipo en una *alpha* jugable.

Ahora que el objetivo está claro, se realiza la Milestone planning donde todos los miembros del equipo van a participar para elaborar cada una de las tareas que se van a llevar a cabo para cumplir con el objetivo. (En el anexo número 6 se adjuntan las tareas reales que se realizaron para lograr este objetivo.)

A continuación, una vez decididas todas las tareas a realizar, el equipo realiza el *planning poker* donde se asignarán cada una de las dificultades de cada tarea. Las tareas las cuales se tienen que clasificar son las siguientes:

- Programar lógica de *teleport* -- 8
- *Debugar* "super sicker" -- 7
- Modelar coches -- 7
- Sistema de conteo de vueltas -- 6
- Estructurar comportamiento IA -- 6
- Diseñar trazado circuito -- 6
- Estudiar Luz -- 5
- *Concepts* y colores habitación -- 5
- Establecer valores ítems -- 5
- Programar menú básico -- 4
- Definir *landmarks* -- 4
- Colocar *placeholders* en la escena -- 4
- *Concepts* UI -- 4
- Colocar *colliders* curvas -- 3
- Programar "changers" -- 3
- Programar cajas ítems -- 3
- Crear pista con piezas sencillas -- 3
- Modelar piezas pista -- 3
- Boceto del escenario -- 2
- *Menu flow* -- 2
- Establecer railes en el circuito -- 2
- Ajustar escalas -- 1
- Colocar "changers" -- 1
- Colocar cajas ítems -- 1
- Actualizar GDD -- 1

Estas son las 25 tareas que finalmente forman el Task pool.

En este momento el equipo ya dispone de todas las tareas que harán lograr el objetivo marcada al inicio de la *Milestone planning.* Ahora todas estas tareas están en las *Task pool* y se escogerá el 1/3 de las tareas más complicadas que haya considerado el equipo. A continuación, se muestran las ocho tareas que finalmente forman el *Bronze stage.*

- Programar lógica de *teleport* -- 8
- *Debugar* "super sicker" -- 7
- Modelar coches -- 7
- Sistema de conteo de vueltas -- 6
- Estructurar comportamiento IA -- 6
- Diseñar trazado circuito -- 6

- Estudiar Luz -- 5
- *Concepts* y colores habitación -- 5

Una vez este seleccionado este 1/3 el *Bronze stage* estará creado y el resto de las tareas quedaran apartadas en el *Task pool* hasta que se finalice el *stage* constituido.

6.2.2.3 Representación de las tareas

A continuación, se muestra el ejemplo de una tarea ya completada en el post-it para mostrar cómo se representaría.

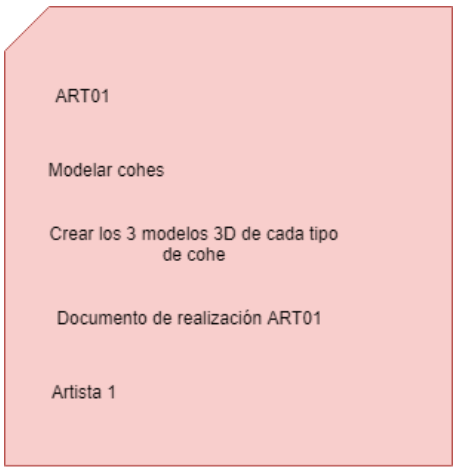

Figura 26. Representación post-it versión 1. Fuente: Elaboración propia.

Una vez completados todos los post-its necesarios con las tareas, se colocan en el departamento correspondiente en la columna de estudio.

6.2.2.4 Proceso de trabajo

En este momento cada uno de los departamentos estudiara la manera más óptima de realizar la tarea y realizaran un documento donde se especifique todo el procedimiento con las sugerencias que se hayan encontrado para realizar la tarea correspondiente. Este documento no podrá sobrepasar una hoja para que esta explicación sea lo más sintética y clara posible.

Una vez finalizado el proceso de estudio el proceso de elaboración comienza y los miembros del equipo realizaran las tareas que les correspondan. Durante este periodo de tiempo cada dos días se realizará la *BiDaily*. Se irán realizando las tareas necesarias para

Cuando el proceso de creación termine entra en el estado de *Verification*, para esta simulación se contempla que una de las ocho tareas no pasara la prueba de verificación así si podrán observar los dos caminos que una tarea puede seguir.

De las ocho tareas realizadas siete son correctas y por ello acaban el recorrido en la columna del panel de *Achievend*. Por otro lado, la tarea de programar la lógica del teleport no pasa la prueba de verificación, por ello esta tiene que volver a la columna de *Studying* en el panel y el programador tendrá que buscar una nueva forma de ejecutar esta tarea, se modificara el documento realizado anteriormente con las instrucciones y se volverá a realizar todo el proceso de creación.

Esta vez la tarea pasa el proceso de verificación y pasa al estado de *Achieved*. En este momento el proceso de desarrollo de la *bronze stage* ha finalizado. En este momento todas las tareas son retiradas del panel y guardadas en el *Graveyard.*

A continuación, se realizará con todo el equipo la Stage review para comprobar como ha ido el desarrollo y poder evitar problemas futuros en el siguiente stage.

Finalmente, este proceso se realizará dos veces más con el resto de las tareas que quedan en la *Task pool* y se iterara el proceso cambiando de objetivo en las *milestones* hasta conseguir entregar el producto final.

6.3 Entrevistas

Para poder verificar y corroborar la metodología hasta ahora propuesta se han realizado tres entrevistas a profesionales del sector que podían aportar su visión de lo hasta ahora conceptualizado y si esta era una buena aproximación. Las entrevistas son de tipo semi-estructuradas por lo que, dependiendo de las respuestas de los entrevistados, obtendríamos un tipo de información u otro que más tarde se analizará para extraer resultados y realizar la modificación a la metodología hiper-ágil.

6.3.1 Estructura

A continuación, se muestra la entrevista semi-estructurada que se ha utilizado. Dicha entrevista se organiza en preguntas primarias, que son utilizadas para obtener información básica del entrevistado y el estudio donde trabaja. Las preguntas sobre las demologías que el entrevistado utiliza en el momento de la entrevista con su equipo. Por ultimo las preguntas en relación a la metodología conceptualizada para conocer la opinión de los diferentes aspectos de esta.

Preguntas primarias

- ¿Cuántos empleados sois?

- ¿Estáis desarrollando algún proyecto actualmente?

- ¿En qué fase de desarrollo está?

Metodologías

- ¿Utilizáis algún tipo de metodología ágil?

Sí

- ¿Cuál es esa metodología?

- ¿Habéis hecho alguna modificación de ésta para adaptarla a vuestro equipo?

(Sí han hecho una modificación)

- ¿Cuál fue el motivo de la modificación?

- ¿En qué consiste esa modificación?

(Si no la han hecho)

- ¿Os habéis planteado hacer alguna para intentar mejorar la gestión del proyecto?

- ¿Si tuvieras que cambiar la metodología que utilizas que le pedirías?

No

- ¿Cuál es el motivo por el cual no utilizáis estas metodologías?

- ¿Lo habéis intentado con anterioridad?

Sí - ¿Cuál fue la experiencia?

No - ¿Que te ha hecho decidir no aplicarla?

- ¿Para empezar a utilizar una metodología ágil que crees que tendría que aportar?

Información para mi metodología

"Explicación de la metodología propuesta"

- ¿Consideras útil el rol del Team Manager?

- ¿Crees conveniente, programar un tiempo para el estudio de una tarea y un tiempo de revisión antes de considerarla finalizada?

- ¿Ves conveniente limitar las secciones de Estudio y Revisión en número de personas que la realizan?

- ¿Se ha subdividido una milestone en tres stages diferentes, consideras conveniente una división tan concreta o es mejor adaptarse a la longitud de la milestone con más o menos stages?

- ¿Consideras más óptimo hacer reuniones cada dos días en vez de una al día para conocer el progreso?

- ¿Consideras óptimo limitar las reuniones a días específicos?

- ¿Utilizarías esta metodología con tu equipo? ¿Por qué?

6.3.2 Entrevistados

Como entrevistados se han escogido a tres profesionales del sector de los videojuegos, de tres empresas diferentes de Barcelona.

Javier Flores[1]

El primer entrevistado pertenece a la empresa Tangelo games, una empresa indie dedicada a la creación de títulos de genero casino-casual. Disponen de su aplicación en web donde están ubicados todos sus juegos y actualmente están haciendo el port de ese entorno web a aplicación móvil para poder llegar a más público.

Actualmente, *Producer* a cargo del desarrollo de la aplicación para móvil, ha trabajado en diversas empresas del ámbito tecnológico trabajando incluso como *Lead Artist* en Ubisoft Barcelona. Actualmente en Tangelo games utilizan Scrum como metodología ágil y es Javier quien lo gestiona haciendo de *Scrum master*.

María Fernanda Díez[2]

La segunda entrevistada pertenece a la empresa Omnidrone situada en Barcelona, actualmente están trabajando en un juego que está en *soft-launch*. El titulo anterior de esta compañía fue Titan Brawl, un juego para móvil de estrategia-PVP a tiempo real.

[1] LinkedIn Javier Flores: https://www.linkedin.com/in/javierfloresduran/

[2] LinkedIn María Fernanda Díez: https://www.linkedin.com/in/mariafdiez/

Actualmente, en el proyecto que están realizando María es la *Producer*. Ella es especialista en la metodóloga Scrum y en gestión de proyectos. En Omnidrone actualmente Scrum es la metodología seleccionada para realizar sus proyectos.

Ivan Magrans[3]

El tercer y último entrevistado Ivan Magrans trabaja para la liga de videojuegos profesionales y también en The Breach Estudio, un estudio indie de Barcelona con 25 trabajadores. Actualmente están trabajando en dos títulos simultáneamente, un juego de móvil y un juego *multiplayer* para pc.

Actualmente Ivan trabaja como COO en la liga profesional de videojuegos, pero anteriormente también trabajo como *Product Owner* y *Stuido Manager*. Es especialista en gestión de proyectos, *game production* y juegos online.

6.3.3 Análisis entrevistas

Como se especificó en la metodología del trabajo una vez realizadas las entrevistas se ha realizado un análisis de las misma siguiendo unos pasos concretos para extraer la información necesaria para modificar la metodología que se está creando.

El análisis consiste en la creación de un documento donde se unificarán todas las respuestas dependiendo de la pregunta contestada, de esta manera se podrá visualizar mejor toda la información respectiva a esa pregunta.

Seguidamente se ha creado un segundo documento donde se han anotado los diferentes puntos clave obtenidos de las respuestas, se relacionarán esos puntos clave con la metodología para poder hacer una segunda versión con las mejoras de los expertos.

Después de todo el análisis se han extraído siete puntos que se han considerado clave de entre todas las respuestas obtenidas. Estos puntos se integrarán en la nueva metodología como cambios para poder mejorarla.

- No subdividir las *milestones* hacer estas variables siendo el objetivo más pequeño, pasando de una milestone a una "*feature creation*".
- Adaptar la metodología al equipo, no el equipo a la metodología.
- Comprobar las dependencias de cada tarea.

Estos puntos incitan a crear un sistema más orgánico que se adapte a la capacidad de trabajo que tenga el equipo de esta manera es más fácil ajustar los tiempos que se va a tardar en realizar la *feature* y calcular recursos es más sencillo. También resulta más sencillo ver

[3] LinkedIn Ivan Magrans: https://www.linkedin.com/in/ivanmagrans/

cuáles van a ser las dependencias de las diferentes tareas que son sumamente importantes cuando se está trabajando con equipos multidisciplinares.

"Propongo un enfoque por "*Feature"* y a partir de ahí mirar las dependencias entre equipos, y hacer un planning hasta que se termine dicha *Feature*."

"[…]no creo en las metodologías que son como instrucciones, sino de plantear una serie de ideas, de pautas, de posibilidades que se puedan adaptar al día a día del equipo […]"

- Hacer una reunión diaria y suprimir la reunión cada dos días.
- Promover la comunicación.

Estos dos puntos se complementan el uno al otro, los entrevistados ha considerado que la visión de la *BiDaily* para reducir el tiempo en reuniones era una visión correcta, pero eso entraba en conflicto directamente con promover la comunicación. Por lo que han recomendado para solucionar esto planear muy bien de lo que debe ser hablado en la reunión.

"Yo estoy de acuerdo con que se celebre un daily, es el momento en que todo el mundo está obligado a tomarse el tiempo para saber qué hace el resto y el poder explicarlo para generar esa cohesión."

- Suprimir el rol del *Team manager,* pero transferir la tarea al *Producer*.

El rol del Team manager ha parecido interesante para los tres entrevistado, sin embargo, los tres lo ven demasiado difícil de implementar en un equipo indie donde los recursos están tan limitados. Por otro lado, se considera mantenerlo, pero migrando las tareas al rol del *Producer*.

"Creo que es interesante este rol, pero creo que si estamos enfocados a estudios indie que son estudios con un numero de recursos limitados, creo que es un lujo tener una persona cien por cien dedicada a esto."

- Limitar los tiempos.

Por ultimo limitar los tiempos todos los entrevistados han coincidido en que hay que poner límites a las tareas, ya que si no se podría entrar en un bucle de mejora de la tarea del que no se podría salir.

"Sí, imprescindible para reducir la cantidad de *rework* y bugs que aparecen posteriormente."

Se han escogido estos puntos porque son los considerados como más afines para el objetivo de la metodología y la propuesta que se está haciendo. Con estos puntos se realizarán los cambios que hagan más viable la metodología dentro de un equipo real.

6.4 Metodo híper-ágil versión 2

Una vez analizadas las entrevistas y con los datos extraídos sobre las recomendaciones que los entrevistados han propuesto se han realizado una serie de cambios a la metodología generando una segunda versión. A continuación, se explica el porqué de los cambios y como se ha realizado dentro de la metodología.

6.4.1 Cambios nueva versión

Con los puntos que se han considerado de las entrevistas se han formulado diferentes cambios para conseguir hacer la metodología propuesta más interesante para los estudios indie. En esta segunda versión se han realizado cambios referentes a los roles involucrados en la metodología siguiendo las propuestas de los entrevistados.

Los cambios realizados son:

- La eliminación del rol de *Team manager*
- Supresión de los *stages*
- Modificación en el *study kick-off*
- Cambio de *BiDaily* a *Daily*
- Mayor control en las dependencias

A continuación, se detallan cada uno de estos cambios y se justifican estos cambios a partir de las respuestas de los entrevistados.

El rol de *Team manager* desaparece como un rol individual en una persona fuera del equipo, pero se ha querido mantener la función que realizaba ya que ha parecido interesante a todos los entrevistados y sí que veía potencial en esta figura.

A partir de ahora las tareas que realizaba el *Team manager* pasaran al *Project manager*, por ello, ahora sí, la persona que realice esta tarea estará fuera del equipo y no desarrollara a la vez que realiza estas funciones.

En la nueva versión se van a eliminar la subdivisión de las *milestones* (*stages*), por eso ahora el objetivo va a ser más pequeño, aumentando el número de iteraciones que se van a realizar durante el desarrollo, de esa manera la metodología también se vuelve más ágil pudiendo retocar parámetros de manera más continuada. Por este motivo se considera que la desviación de la dificultad asignada a las tareas será mucho menor, por lo tanto, se conseguirá una mejoría en el tiempo de trabajo.

Ahora los objetivos de las milestones serán *features* concretas que se quieran implementar en el proyecto. Este cambio también obedece a uno de los puntos fuertes de Kanban seleccionados anteriormente, que es dividir el trabajo en ítems pequeños.

Anteriormente, la metodología completaba el estudio como una sección donde los diferentes departamentos estudiaban la mejor manera de llevar a cabo una tarea para conseguir la máxima calidad desde el inicio. Pero este método podía acarrear problemas ya que, por ejemplo, la decisión de programación podía ser incompatible con la decisión de arte.

Por esto mismo y gracias a que la metodología está contemplada para un equipo pequeño se realizara lo que se ha llamado *study kick-off* esta sesión de trabajo involucrara a todo el equipo y se buscara la mejor alternativa para cada tarea. Gracias al conocimiento de cada uno de los miembros se conseguirá una mejor compatibilidad entre todas las propuestas de realización, lo que llevará a reducir el desperdicio y aumentar la calidad del producto final.

Otro punto que los entrevistados remarcaron y que se ha tomado en cuenta para realizar un cambio en la metodología es la propuesta de la *BiDaily,* esta va a ser eliminada y se va a sustituir por una reunión diaria con el mismo objetivo. Los entrevistados remarcaron la importancia de esta reunión y por eso se ha realizado este cambio. La realización de una *Daily* también implica un mejor flujo de comunicación que los entrevistados remarcaron como imprescindible para cualquier equipo de trabajo.

Finalmente, uno de los nuevos cambios implementados hace referencia a las dependencias entre las tareas que generan el trabajo, esto no se había tenido en cuenta en la metodología anterior, pero sin embargo es un problema que va a existir siempre.

Estas dependencias hacen referencia a tareas que dependen de un trabajo realizado anteriormente. Por ejemplo, no se puede texturizar un modelo 3D sin que exista el modelo, por ello primero el modelo 3D debe de estar acabado.

6.4.2 Explicación

Organización del equipo

El **equipo** constará de todos los programadores, artistas y diseñadores de los que el estudio disponga.

El ***Project manager*** es la persona encargada de gestionar toda la parte referente al proyecto que queda fuera del desarrollo, esta persona realizará tres tareas diferentes durante el tiempo de creación del proyecto. Este estará fuera de la estructura del equipo y no desarrollará mientras realice estas funciones principales.

Las tareas que el *Project manager* realizará son:

- Gestionar las tareas relativas a la metodología aplicada: Mantener al día el panel de tareas, comprobar que el equipo está realizando las tareas referentes a esta, organizar las reuniones y el tema sobre el cual se va a hablar.
- Mantener el contacto con el cliente, tanto si es un Publisher o el propio mercado, para conocer en todo momento como tiene que evolucionar el proyecto.
- Mantener al equipo con un estado de ánimo alto y de resolver cualquier incidencia de carácter personal que las personas del equipo puedan requerir.

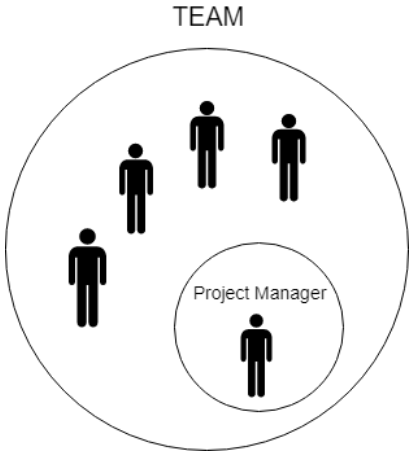

Figura 27. Estructura del personal versión 2. Fuente: Elaboración propia.

Creación de las tareas

Primero, se debe especificar cuál va ser el objetivo que se plantea como *milestone*, por ejemplo, hacer completamente funcional el personaje principal.

En este punto se realiza la ***Milestone planning*** (reunión) donde se generan las tareas que se van a llevar a cabo para lograr el objetivo marcado, estas tareas son las consideradas por el equipo como esenciales para lograr el objetivo de la *milestone*. Esta reunión implica tanto al equipo como al *Project manager* y no tendrá una duración concreta. Se debe de consumir todo el tiempo necesario para que las tareas a realizar sean las correctas y necesarias para lograr el objetivo.

Una tarea tiene que ser indivisible, es decir, una tarea tiene que ser la mínima unidad de trabajo de una persona. Un ejemplo sería, "Modelar el personaje principal", otra sería "Riggear el personaje principal". En ningún caso, una tarea puede ser "Hacer el personaje principal" o "Balancear daño enemigos".

De esta reunión, se obtiene un total de tareas (***Task pool***) para lograr el objetivo de la milestone. Con estas tareas acordadas entre el equipo y el ***Project manager*** se realiza un *poker* (método de estimación y priorización) que nos dará una puntuación del 1 al 10 que representará la dificultad estimada por el equipo.

Una vez ajustada la dificultad de las tareas se deben de analizar todas las tareas y buscar cual va a ser la dependencia de cada una. Se tiene que ordenar dependiendo que las tareas generen,

de esta manera se obtendrá el orden correcto de realización que más tarde se incluirá en la información relevante del post-it.

Representación de las tareas

Con las tareas ya acordadas y clasificadas, estas se deben colocar en el panel donde nos mostrará el estado de cada tarea. El panel tiene seis columnas:

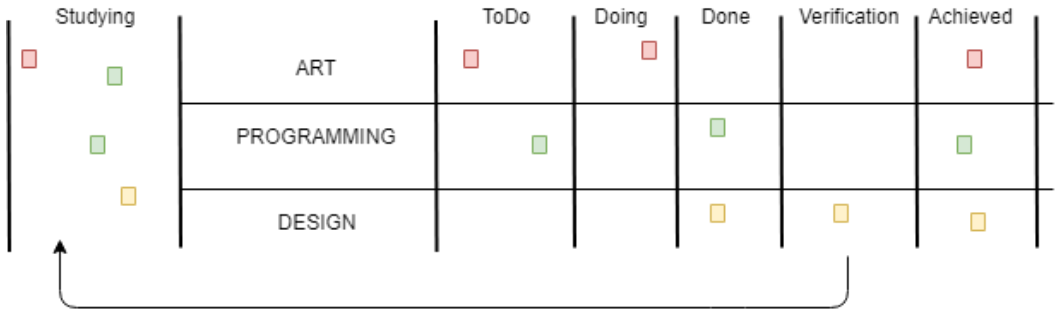

Figura 28. Panel para la metodología Híper-ágil versión 2. Fuente: Elaboración propia.

Todas estas tareas serán representadas dentro de un post-it. La información de la tarea es representada de la siguiente manera:

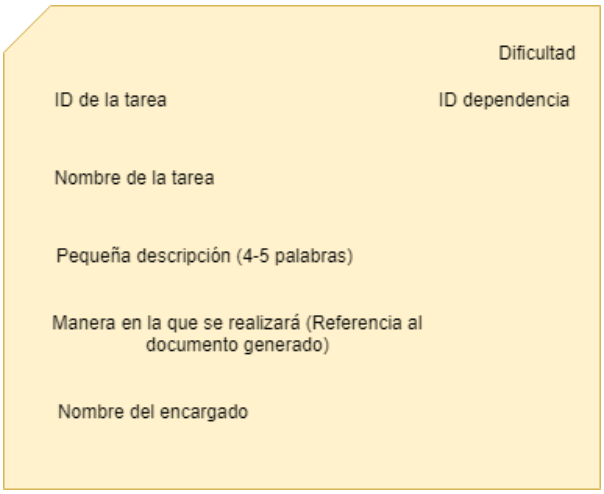

Figura 29. Representación de la tarea en post-it. Fuente: Elaboración propia.

Una vez estén creadas todas las tareas necesarias para la *milestone*, se realizará el *Study Kick-off*, donde una a una se irán analizando con todo el equipo presente para comprobar cuál será la mejor forma de implementar esa tarea en el juego, si el equipo tiene el conocimiento sobre la manera óptima, se realizará un pequeño documento (máximo una página) con toda la información relevante para esta tarea y se adjuntará a la tarea.

Si el equipo no tuviera claro cuál es la mejor forma o tienen dudas de cómo implementar esa tarea todos buscarán formas para implementarla, para finalmente decidir la que mejor se adapte al desarrollo y generaran el documento con la información necesaria para enlazarlo a la tarea.

De este modo, todo el equipo conoce como se van a realizar las tareas, se incrementa la auto-organización y la colaboración entre el equipo, y se incrementa también la calidad para poder evitar el desperdicio mientras se esté desarrollando.

Proceso de trabajo

Una vez finalizada la sesión de estudio, se inicia el proceso de trabajo realizando las tareas asignadas a cada departamento teniendo en cuenta las dependencias necesarias, para ir pasando de una columna a otra.

Cada día se hará una pequeña reunión entre todos los miembros del equipo para ver que se está realizando, qué problemas se están encontrando y como se van a solucionar. Esta reunión se llama ***Daily.*** La duración será de aproximadamente de unos diez minutos y todos los miembros del equipo y el *Project manager* estarán presentes. En esta reunión los integrantes del equipo deberán explicar cómo llevan las tareas y que problemas están teniendo mientras las realizan.

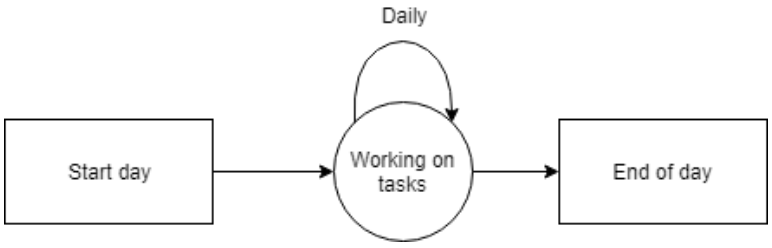

Figura 30. Flujo de trabajo diario versión 2. Fuente: Elaboración propia.

Una vez la tarea llegue a la parte de ***Verification*** el departamento responsable realizará la verificación de manera conjunta *Project manager* de esa manera se asegura la mayor calidad posible dentro de todas las etapas. En este momento se determinará si se considera valida la tarea o no. Si no se determina como válida deberá volver al estudio y se tendrá que encontrar una forma mejor de implementar la *feature*.

Finalmente, si esta es aceptada pasara al estado de *Achieved.*

Una vez logradas todas las tareas se eliminarán del panel y se guardaran de manera simbólica en el ***graveyard*** (Zona donde se guarden los post-its) y se realizara una ***milestone review*** (reunión) para ver cómo ha ido todo el proceso y que puede mejorarse en la próxima

iteración. Nuevamente esta reunión involucra al equipo y el *Project manager.* Esta reunión tendrá una duración variable, el equipo tiene que discutir cada uno de los puntos que considere precarios mientras se ha desarrollado, de esta manera se pueden tomar medidas para que se resuelvan los problemas encontrados y se afronte con más facilidad la siguiente *milestone*.

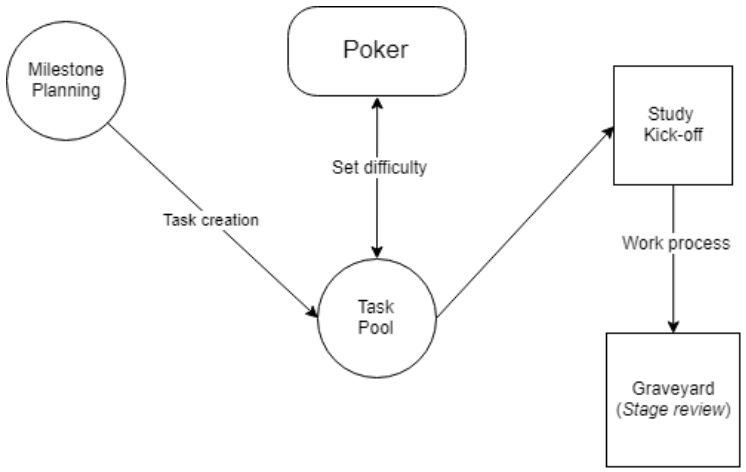

Figura 31. Flujo de trabajo metodología híper-ágil versión 2. Fuente: Elaboración propia.

6.4.3 Simulación

De nuevo en esta simulación se continúa utilizando como ejemplo el proyecto final realizado en el tercer curso. (Ver Figuras de 6 a 10).

En esta simulación se ha tenido en cuenta las tareas que se realizaron para pasar del prototipo a la *alpha*, sin embargo, el objetivo en esta versión de la metodología es distinto y la *milestone* consistirá en crear el escenario del juego.

6.4.3.1 Organización del equipo

El **equipo** constará de dos artistas, un programador y dos diseñadores.

El ***Project manager*** en este proyecto no se contempla un *Project manager* externo al equipo por esto, uno de los diseñadores toma el rol, siendo el encargado de organizar la metodología con el equipo, el contacto con el profesorado (en esta simulación el cliente) y el control del estado de motivacional y emocional del equipo.

De esta manera el equipo estaría formado finalmente, y la constitución de este se representa en el esquema que se muestra a continuación.

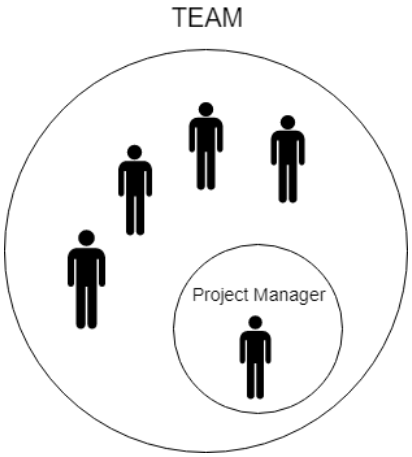

Figura 32. Estructura personal simulación versión 2. Fuente: Elaboración propia.

6.4.3.2 Creación de las tareas

Una vez el equipo está formado, se tiene que establecer un objetivo, este como se ha expuesto anteriormente es el de crear el escenario de juego.

Ahora que el objetivo está claro, se realiza la Milestone planning donde todos los miembros del equipo van a participar para elaborar cada una de las tareas que se van a llevar a cabo para cumplir con el objetivo. (En el anexo número 6 se adjuntan las tareas reales de las cuales se extraen cuales completarán el objetivo.)

A continuación, una vez decididas todas las tareas a realizar, el equipo realiza el *planning poker* donde se asignarán cada una de las dificultades de cada tarea y su dependencias con el resto. Las tareas las cuales se tienen que clasificar son las siguientes:

- 1.Diseñar trazado circuito -- 6
- 2.Estudiar Luz -- 5 -- Dep: 3,5
- 3.*Concepts* y colores habitación -- 5 -- Dep: 1
- 4.Programar menú básico -- 4
- 5.Definir *landmarks* -- 4 -- Dep: 1
- 6.Colocar *placeholders* en la escena -- 4 -- Dep: 10
- 7.Colocar *Colliders* curvas -- 3 -- Dep: 1,10
- 8.Programar “changers” -- 3
- 9.Programar cajas ítems -- 3
- 10.Modelar piezas pista -- 3 -- Dep: 1,11
- 11.Boceto del escenario -- 2 -- Dep: 1
- 12.Establecer railes en el circuito -- 2 -- Dep: 1,10
- 13.Ajustar escalas -- 1 -- Dep: 6,10
- 14.Colocar “changers” -- 1 -- Dep: 1,10,8
- 15.Colocar cajas ítems -- 1 -- Dep: 1,10,9

Estas son las 15 tareas que finalmente forman la *milestone* en esta simulación.

En este momento el equipo ya dispone de todas las tareas que harán lograr el objetivo marcado al inicio de la *Milestone planning.*

6.4.3.3 Representación de las tareas

Con las tareas ya acordadas y clasificadas, estas se deben colocar en el panel donde nos mostrará el estado de cada tarea.

Todas estas tareas serán representadas dentro de un post-it. A continuación se muestra un ejemplo de cómo quedaría la tarea representada en un pos-it una vez finaliza.

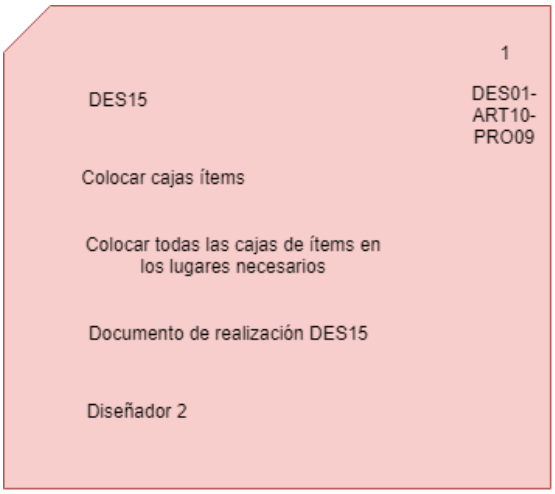

Figura 33. Representación post-it versión 2. Fuente: Elaboración propia.

Una vez estén creadas todas las tareas necesarias para la milestone, una a una se irán analizando con todo el equipo presente para comprobar cuál será la mejor forma de implementar esa tarea en el juego, si el equipo tiene el conocimiento sobre la manera óptima, se realizará un pequeño documento (máximo una página) con toda la información relevante para esta tarea y se adjuntará a la tarea.

Si el equipo no tuviera claro cuál es la mejor forma o tienen dudas de cómo implementar esa tarea todos buscarán formas para implementarla, para finalmente decidir la que mejor se adapte al desarrollo y generaran el documento con la información necesaria para enlazarlo a la tarea.

6.4.3.4 Proceso de trabajo

En este momento el equipo empieza el proceso de trabajo donde las tareas irán cambiando de estado.

Durante todo el proceso de desarrollo diariamente se realizará una Daily donde cada uno de los miembros del equipo explicará qué problemas está teniendo en la realización de la tarea y en qué estado se encuentra esta.

Una vez la tarea llegue a la parte de *Verification* el departamento responsable realizará la verificación de manera conjunta *Project manager* de esa manera se asegura la mayor calidad posible dentro de todas las etapas.

En esta simulación se considera que todas las tareas han pasado la comprobación excepto una para visualizar cuales son los diferentes caminos que una tarea puede llevar. En el caso de la tarea que no pasa la verificación vuelve al estado de estudio donde todos los miembros se reúnen para buscar una nueva fórmula de implementar la tarea, se modifica el documento con la información de las instrucciones y la tarea vuelve a empezar el proceso de implementación.

Una vez logradas todas las tareas se eliminarán del panel y se guardaran de manera simbólica en el *graveyard* (Zona donde se guarden los post-its) y se realizara una *milestone review* (reunión) para ver cómo ha ido todo el proceso y que puede mejorarse en la próxima iteración.

7. Conclusiones

Para finalizar este trabajo se va a concluir sobre los aspectos más destacados que se han encontrado durante todo el proyecto.

La conceptualización de la metodología ha sido parte fundamental del proyecto y de dónde se han extraído más puntos a concluir.

Una primera conclusión es que existen múltiples dependencias entre cada una de las tareas y que es necesario que todos los departamentos se involucren durante el desarrollo de las f*eatures*. En la metodología y en el control de las tareas se tiene que prever cuáles van a ser esas dependencias e involucrar al equipo para intentar minimizar el impacto que pueda conllevar un error en este cálculo.

También se ha comprobado que no es óptimo para el desarrollo de un videojuego planificar a largo plazo, ya que provoca que las tareas realizadas durante el desarrollo del proyecto se difuminen y se puedan llegar a planear tareas innecesarias o bien no definir aquellas que pudieran resultar sumamente importantes. Resulta más eficaz planificar y realizar seguimientos en un corto espacio temporal para poder corregir posibles desviaciones respecto al objetivo final.

Sobre la metodología conceptualizada podemos concluir que es sumamente importante que el método se adapte al equipo, que se pueda adecuar a las necesidades de cada estudio y no provocar que el equipo modifique su forma de trabajar. La metodología ha de adaptarse al equipo y no al revés.

De los puntos anteriores podemos discernir que hay aspectos a mejorar en el desarrollo de la metodología y uno de ellos es la comunicación y

la visibilidad del trabajo entre los miembros del equipo para lograr más cohesión en el ámbito laboral. En lugar de centrarnos en definir una metodología única para toda la duración del proyecto, se propone como aspecto de mejora el realizar pequeños módulos que ayuden a la gestión del proyecto y que permitan al estudio adaptarse a sus necesidades en cada momento. En el caso de los estudios indie, este tipo de metodología por módulos consideramos que es especialmente adecuado ya que cuentan con un escaso número de empleados y unos recursos económicos también limitados.

Por ello, incluir en el proceso de desarrollo una metodología que permita gestionar la producción y maximizar los resultados es imprescindible.

Las metodologías estudiadas, pese a estar adaptadas al desarrollo de videojuegos y a su utilización por casi la totalidad de las empresas del sector, siguen arrastrando grandes

problemas dado su origen en fábricas con elevado personal y producción mecanizada. En la actualidad, a pesar de la existencia de compañías multinacionales, con recursos prácticamente ilimitados, vemos que cada vez se crean más estudios de pequeño tamaño y que una metodología adaptada a sus características les beneficia considerablemente.

Finalmente concluimos que, aún habiendo realizado encuestas a profesionales del sector, no se conocerá el potencial de la metodología híper-ágil hasta que se pueda aplicar a un desarrollo real. Es por ello que el objetivo será que en un futuro próximo podamos adaptar y mejorar este modelo de metodología para que los estudios indie puedan aplicarla a cualquier tipo de proyecto, haciendo hincapié en el fomento de la comunicación entre los miembros del equipo.

8. Bibliografía y Referencias

Agile Alliance. (2001). Obtenido de Agile Alliance: https://www.agilealliance.org

Anderson, D. J., & Carmichael, A. (2016). *Essential Kanban condensed.* Seattle: Lean-Kanban University.

Assassin's Creed (2007). Ubisoft.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifesto for Agile Software Development*. Obtenido de Manifesto for Agile Software Development: http://agilemanifesto.org

Bethke, E. (2003). *Game Development and Production.* Plano, Texas: Wordware Publishing, Inc.

Braid (2008). Number None, Inc.

Call of Duty (2003). Activision.

Castle Crashers (2008). The Behemoth

Castrillón, E. P. (2011). Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje -MESOVA-. *Revista Virtual Universidad Católica del Norte*. Obtenido de http://revistavirtual.ucn.edu.co/

Cohn, M. (2006). *Agile estimating and planning.* Upper Saddle River, NJ: Pearson Education, Inc.

Cohn, M. (2006). Planning Poker. *Chapter, 6*, 56 - 59.

Cusumano, M. A., & Smith, S. (1995). *Beyond the Waterfall: Software Development at Microsoft.*

Donkey Kong (Arcade, 1981). Nintendo.

Galaga (Arcade, 1981). Namco.

Grand Theft Auto (1997). Rockstar.

Gómez, C. L., García, A. Á., & Dedo, R. d. (2017). *Métodos Ágiles. Scrum, Kanban, Lean.* ANAYA MULTIMEDIA.

Keith, C. (2010). *Agile Game Development with SCRUM.* Addison Wesley.

Kent, S. L. (2016). *La gran historia de los videojuegos*. B DE BOOKS.

Latorre, Ó. P. (2016). Indie or Mainstream? Tensions and Nuances between the Alternative and The Mainstream in Indie Games. *Digital game*, 15-30. doi:http://dx.doi.org/10.7238/a.v0i54.2818

Mantilla, M. C., Ariza, L. L., & Delgado, B. M. (2013). Metodología para el desarrollo de aplicaciones móviles. *Tecnura: Tecnología y Cultura Afirmando el Conocimiento, 18*(40), 20-35.

Pacman (Arcade, 1980). Namco.

Pong (Arcade, 1972). Atari.

Rahimian, V., & Ramsin, R. (2008). Designing an agile methodology for mobile software development: A hybrid method engineering approach. *2008 Second International Conference on Research Challenges in Information Science* (págs. 351-356). Marrakech: IEEE.

Rufí, J. P. (2016). El Imperio Indie del Videojuego Español: Debilidades, Amenazas, Fortalezas y Oportunidades de la Industria Española Del Software de Ocio. *Razón y Palabra*, 839-852. Obtenido de https://archivos.revistarazonypalabra.org/index.php/ryp/article/view/48

World of Goo (2008). 2D Boy.

Centres universitaris adscrits a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Disseny i Producció de Videojocs

Anexos

Adrián Cañete Paterna
TUTOR: Dr. Adso Fernández Baena
2018 - 2019

TecnoCampus Mataró-Maresme

Índice

1. Transcripción entrevista Javier Flores

- ¿Cuántos empleados sois?

49

- ¿Estáis desarrollando algún proyecto actualmente?

En estos momentos estamos haciendo la transformación de nuestra plataforma web a un entorno móvil, que nos permita migrar toda nuestra experiencia y catálogo de juegos a un entorno móvil. Y de paso aprovechar para mejorar y adaptarnos a las nuevas tecnologías en la página web para poder compartir los recursos.

- ¿En qué fase de desarrollo esta?

Tenemos una primera versión funcional y abierta al público, en un estado quizás muy primitivo pero que a día de hoy es una base relativamente solida sobre la que ir construyendo todos los demás estado siempre en función de la necesidad del usuario y siempre enfocados a un crecimiento de negocio.

Metodologías

- ¿Utilizáis algún tipo de metodología ágil?

Si

- ¿Cuál es esa metodología?

Scrum, siempre intentando implementar las funcionalidades que creemos necesarias en cada momento del desarrollo.

- ¿Habéis hecho alguna modificación de esta para adaptarla a vuestro equipo?

Si, si, por supuesto yo siempre digo en las reuniones que la metodología tiene que ser una herramienta o un facilitador de nuestras tareas de nuestro día a día y no podemos permitir que la metodología sea como un impedimento que nos retrase, sino algo que nos permita ser más ágiles y que nos permita trabajar de una forma más sencilla. Creo que la metodología Scrum no es perfecta y evidentemente no es aplicable a todas las empresas, casos, etc. Tememos una base de Scrum donde intentamos seguir los artefactos los eventos o líneas principales. Cuando hay una entrega final sí que somos más estrictos, ahora que estamos en una fase intermedia somos un poquito más flexibles intentamos aplicarla en función del estado del desarrollo.

- ¿Cuál fue el motivo de la modificación?

Al final la modificación viene sola no se hace una planificación de hacerlo de una manera u otra es el día a día lo que nos hace tomar las diferentes decisiones dependiendo de cada

situación de cada requisito. Los cambios como podrían ser añadir una persona nueva al equipo o retirar a una persona de este. Esto es lo que hace que no te puedas ceñir mucha a la metodología sin que te implique perder esta flexibilidad.

- ¿En qué consiste esa modificación?

Se flexibiliza la gestión de las tareas, es decir dependiendo del miembro del equipo permitimos que él se genere sus stories creándose los elementes en tickets y auto gestionándose o dejamos total libertad para que gestionen las tareas más a su gusto.

- ¿Si tuvieras que cambiar la metodología que utilizas que le pedirías?

Para mí la base de una metodología tiene que ser la visibilidad, uno de los primeros problemas de lo que se queja los equipos es de la falta de visibilidad de las tareas y las dependencias que pueden tener con los compañeros. Saber en cada momento que se está haciendo si se ha hecho ya la tarea. Visibilidad hacia los stakeholders, para permitirnos realizar una estimación de cuán grande es la tarea o cuando la vamos a tener hecha. Una buena metodología tiene que permitirte saber, donde estas y hacia dónde vas. La clave es que el equipo sepa dónde está, donde está el resto y hacia dónde vamos.

Información para mi metodología

"Explicación de la metodología propuesta"

- ¿Consideras útil el rol del Team Manager?

Por lo que he visto hace un poco de coaching un rol un poco de recursos humanos para mantener el buen rollo. Creo que es interesante este rol, pero creo que si estamos enfocados a estudios indie que son estudios con un numero de recursos limitados, creo que es un lujo tener una persona cien por cien dedicada a esto. Yo entiendo que se pueda enfocar que sea o bien el CEO o un Lead que además haga estas tareas motivacionales. Como un rol independiente lo veo poco realizables para un estudio pequeño el poder tener un gestor de felicidad como los llaman ahora.

- ¿Crees conveniente, programar un tiempo para el estudio de una tarea y un tiempo de revisión antes de considerarla finalizada?

Todos necesitamos deadlines, por el riesgo de parálisis por análisis. Yo estoy acostumbrado a liderar equipo de arte y una de las cosas importantes para un artista es tener una fecha limite porque pueden estar trabajando sobre lo mismo indefinidamente. Las fechas ayudan a todo el mundo a organizarse y a saber si vamos en tiempo o no, siempre manejando fechas realistas si tú tienes que leer documentación por espacio de una semana y te dan una hora para hacerlo obviamente no lo vas a conseguir. Soy partidario siempre de encontrar un punto intermedio en las fechas para poder saber si estas dentro de unos tiempos, pero siempre siendo realistas con la tarea.

- ¿Ves conveniente limitar las secciones de Estudio y Revisión en número de personas que la realizan?

En realidad, compartir conocimiento siempre es interesante, pero lo que pasara en realidad es que si estas en un estudio de cinco o seis personas, habrá un back-end, un front-end, una persona de arte. A la practica tendrás roles muy separados para cosas muy concretas. Con esta configuración difícilmente vas a poder crear estos equipos en el caso de que sea posible sí que es interesante, no es inteligente concentrar todo el conocimiento en una persona por mil razones, puede irse de la empresa, nos estar disponible… Sí que es importante compartir conocimiento, pero no siempre es posible.

- ¿Se ha subdividido una milestone en tres stages diferentes, consideras conveniente una división tan concreta o es mejor adaptarse a la longitud de la milestone con más o menos stages?

Dependerá del objetivo que quieras cumplir en esa milestones, es decir si estamos realizando un proyecto muy sencillo si es será muy fácil ver el camino a seguir y esto puede salir bien, pero si estamos creando un juego mucho más grande 3d con inteligencia artificial entonces es muy probable que tengamos que ir haciendo pasos previos de manera más constante. Yo no soy partidario de hacer cosas cerradas sino de ir viéndolas en función de las necesidades del proyecto y nunca hay dos proyectos iguales ni dos desarrollos iguales. Me gusta la normativa que te de cierta flexibilidad, la norma es una serie de directrices, no de leyes, vamos a ver como la podemos aplicar para traerla a tu terreno.

- ¿Consideras más óptimo hacer reuniones cada dos días en vez de una al día para conocer el progreso?

Yo estoy de acuerdo con que se celebre un daily, es el momento en que todo el mundo está obligado a tomarse el tiempo para saber qué hace el resto y el poder explicarlo para generar esa cohesión. Es decir, voy a tomarme cinco, diez minutos del día para dedicarlos a mis compañeros, no que me centro en lo mío voy avanzando y luego ya me lo encontrare. Considero el daily, si está bien gestionado, como una reserva de tiempo para dedicarlo a mis compañeros y va a evitar que me meta en mi trabajo en modo burbuja y no salga de ahí. Pero también estoy de acuerdo que una reunión mal gestionada es una total pérdida de tiempo. La reunión se tiene que preparar y saber de qué se va a hablar cada uno y que se hable de lo que toca saliendo de ella con las cosas claras.

- ¿Utilizarías esta metodología con tu equipo? ¿Por qué?

He visto cositas que me parecían bien otras que no las veo tan claras, yo en definitiva no creo en las metodologías que son como instrucciones sino de plantear una serie de ideas de pautas de posibilidades que se puedan adaptar al día a día del equipo y más aún para equipo indie. No se puede aplicar algo muy estricto porque no vale la pena, pero tampoco se puede dejar que cada uno vaya por libre. Se pueden extraer cosas interesantes, pero no todas las situaciones pueden ser adaptadas a todas las empresas o a todos los equipos.

2.Transcripción entrevista María Fernanda Díez

- ¿Cuántos empleados sois?

Nosotros nos somos un estudio indie per sé, aunque una vez lo fue y he trabajado la mayor parte en ellos como Producer. Lo normal ronda entre 4 y 10 personas. En mi equipo actualmente rondamos unas 12 personas, de las cuales 3 son Leads y 1 Producer (yo misma).

- ¿Estáis desarrollando algún proyecto actualmente?

Sí, pero es absolutamente confidencial.

- ¿En qué fase de desarrollo esta?

En *soft launch* / desarrollo.

Metodologías

- ¿Utilizáis algún tipo de metodología ágil?

Sí

- ¿Cuál es esa metodología?

Las metodologías ágiles son un *framework*, no un set de reglas estricto que se tienen que seguir al pie de la letra, a menos de que se sea muy purista. Nosotros hacemos una mezcla de metodologías ágiles de acuerdo con las necesidades del equipo y el proyecto. Mezclamos Scrum, Kanban y algunos elementos de otras de manera espontánea. Rituales como el *Daily* en la mañana, el uso ágil de Jira, retrospectivas, *user stori*es y *one-to-ones* es algo bastante habitual en nuestro equipo.

- ¿Habéis hecho alguna modificación de esta para adaptarla a vuestro equipo?

Hay cosas que se mantienen siempre, pero hay muchas otras que vamos ajustando con cada sprint. Realmente en el estudio nos definimos como ‘agile’, pero no como una metodología en concreto.

- ¿Cuál fue el motivo de la modificación?

Que la naturaleza del proyecto y las necesidades del equipo van variando, y hay que ser flexible e iterar rápido. El éxito de un sprint y el alcanzar todos los *milestones* mucho depende de la comunicación y las sinergias entre equipos. Es aquí es donde verdaderamente entra el rol de Producer: mirar qué necesita el equipo en ese momento, cómo ‘agilizar’ y fomentar la comunicación constante.

- ¿En qué consiste esa modificación?

Por poner un ejemplo, antes mirábamos las milestones como una totalidad, es decir, como toda una *feature* del juego. Nos funcionaba, hasta que dejó de hacerlo. La naturaleza del proyecto nos requirió más sincronía entre equipos durante todo el sprint (aprox 3 semanas), con lo que incluimos más *deliverables* y más 'micro-milestones' dentro del planning de cada *feature*. Resultó muy beneficioso porque los equipos se marcaban objetivos para evitar afectar/retrasar a distintas dependencias con otros equipos. Al final, se tenía cada *deliverable* a tiempo para pasar al siguiente, con lo que empezamos a 'prototipar' antes y llegar al final del sprint cómodamente cumpliendo todo.

Información para mi metodología

- ¿Consideras útil el rol del Team Manager?

La labor más 'managerial' sí. El rol en sí, depende. El Producer - en mi experiencia- también se encarga de mantener la misión, visión y motivación del equipo, ya que tiene una noción más periférica de hacia dónde va el proyecto. Si se es un Project Manager puro y duro puede que las labores 'managerial' no sean tan clave, por ejemplo: hacer 'appraisals' y 'one-to-ones' con cada miembro del equipo. Sin embargo, yo no apuesto por una visión del Project Manager tan cuadriculada, y opto más por un balance entre Project Management puro y duro con una persona que sepa medir las necesidades y el estado de ánimo del equipo.

Cuento que el equipo propuesto es pequeño, con lo que cada departamento no tendrá un manager individual. Si esto cambiase, optaría por tener un mánager o 'lead' por cada departamento para canalizar la comunicación: Game Design, Art, Dev...

Por otra parte, yo veo más al Team Manager como un Studio Manager, que se encarga de la parte más 'business' con el Publisher, sin descartar al Producer. El Studio Manager negocia y mira más por temas financieros, recursos, etc, y el Producer puede ayudar con la toma de decisiones teniendo en cuenta el 'scope' y ritmo del proyecto.

La estructura que propones es válida y podría funcionar, pero por experiencia siempre hay un fundador/manager/CEO del estudio quien tomará responsabilidades más orientadas al trato con 'stakeholders' y negocio.

- ¿Crees conveniente, programar un tiempo para el estudio de una tarea y un tiempo de revisión antes de considerarla finalizada?

Sobre el tiempo de estudio, sugiero que esto se realice antes siquiera de priorizar las tareas. En la metodología propuesta, se menciona el obtener 10 tasks más difíciles (Bronze stage). Para hacer esto, se tiene que estudiar entre todos qué es lo que se quiere realizar, qué implicaciones puede tener y sobre todo, las dependencias. Lo que se va a desarrollar o la "feature" puede ser una tarea titánica en diseño, mientras que la implementación por parte de los programadores quizás no sea trivial, pero tampoco tenga la misma dificultad. Otro ejemplo: puede que se trabaje en una mecánica, que a nivel diseño pueda ser relativamente fácil de crear, y a nivel de programación también tenga una implementación fácil; pero para arte, puede que sea una tarea que representa gran dificultad (UI/UX complicados, mucho research de estilo, etc).

Propongo un enfoque por "Feature" y a partir de ahí mirar las dependencias entre equipos, y hacer un planning hasta que se termine dicha Feature.

Sobre el tiempo de revisión, es imperativo. Hay que revisar, verificar y hacer un post-mortem o retrospectiva si es necesario. En mi experiencia, no sobran las verificaciones, ni antes ni durante ni después.

- ¿Ves conveniente limitar las secciones de Estudio y Revisión en número de personas que la realizan?

No. Creo que, aprovechando que es un equipo pequeño (y en realidad, aunque no lo fuese), se debería de involucrar a todos. Un kick-off por cada milestone planning y una retrospectiva al final siempre es aconsejable. No es conveniente que puedan haber pérdidas de información o comunicación en un equipo pequeño ya que esto puede escalar si el equipo creciese. Además, muchas veces distintos miembros del equipo pueden ver conflictos potenciales o temas a tener en cuenta que a otros miembros se les escapa.

- Se ha subdividido una milestone en tres stages diferentes. ¿Consideras conveniente una división tan concreta o es mejor adaptarse a la longitud de la milestone con más o menos stages?

Veo bien esta diferenciación (con todo y que me falta información sobre las distintas dependencias) siempre y cuando funcione. Igual que cada feature, ningún milestone es igual a otro y veo bien el poder adaptarse y meter estadios intermedios si el milestone llevase más tiempo. Una planeación con "tasks" como unidad mínima puede incluso considerarse un planning a muy alto nivel, ya que cada task puede tener una serie de "sub tasks". La persona que va a realizar la tarea tiene que considerarlos antes de determinar la dificultad de la misma. Todo depende de qué tan "granular" quiera ser uno con el planning y lo que necesite el proyecto/milestone.

- ¿Consideras más óptimo hacer reuniones cada dos días en vez de una al día para conocer el progreso?

No. Por experiencia, los miembros del equipo pueden decir que no tienen nada que hablar o poner a nadie al corriente, pero si haces una reunión, terminan saliendo temas, invariablemente. Si no hay nada que comentar, perfecto. Pero nunca habrá un "update" en vano. Por experiencia, los 'dailies' son más eficientes, aunque duren 3 minutos.

- ¿Consideras óptimo limitar las reuniones a días específicos?

Creo que hay que tener un balance entre reuniones espontáneas y reuniones programadas. Reuniones espontáneas son más 'agile' y permiten iterar continuamente. Esto es algo que siempre fomento en mis equipos: hablad, hablad, hablad. Sin embargo, siempre incluyo reuniones programadas (por ejemplo, por cada 'story' o 'feature') donde se muestra el avance de todo el equipo a todo el equipo: prototipos jugables, mockups de arte, conceptos, GDDs, etc.

- ¿Utilizarías esta metodología con tu equipo? ¿Por qué?

La verdad es que no, ya que nosotros somos un estudio más grande en un estado de desarrollo más avanzado, con diversas dependencias entre equipos. Pero eso no quiere decir que no funcione. Lo bonito del agile es que todo el tiempo vas fluyendo, adaptándote y modificando acorde a necesidades. Puede que, según la naturaleza del juego y lo que se vaya a desarrollar, esta sea una buena base para arrancar. Mi consejo es que una vez empezado, no parar de mirar si es necesario modificar ciertos aspectos de la metodología inicial, hasta que se tenga una metodología propia de estudio, con algunas prácticas base.

Parece que insisto mucho en las dependencias, y efectivamente así es, puesto que en mi experiencia, infravalorarlas y mirar más por otros aspectos puede generar un atraso considerable y por ende malestar en el equipo. Una dependencia no calculada puede

generar una bola de nieve hacia el resto de los equipos, o un cuello de botella. Todos cometemos errores y a veces es muy difícil a inicio del desarrollo poder tener una estimación de todo a muy bajo nivel o de manera granular. A todos se nos pasan cosas que no las vemos hasta que nos las encontramos, o situaciones con las que no contábamos. Por eso mi opinión es que entre más tiempo y atención al detalle se tenga en la pre-producción o planning de cada milestone, mejor va a ir la producción.

3.Transcripción entrevista Ivan Magrans

- ¿Cuántos empleados sois?

25

- ¿Estáis desarrollando algún proyecto actualmente?

Dos proyectos, un juego móvil y uno para PC multiplayer

- ¿En qué fase de desarrollo esta?

El móvil muy avanzado y en soft-launch

El de PC le queda 1 año de desarrollo todavía

Metodologías

- ¿Utilizáis algún tipo de metodología ágil?

Sí

- ¿Cuál es esa metodología?

Kanban

- ¿Os habéis planteado hacer alguna para intentar mejorar la gestión del proyecto?

De momento no, Kanban es un framework muy robusto y flexible

Información para mi metodología

- ¿Consideras útil el rol del Team Manager?

Aunque en el plano teórico está bien (coge una de las funciones del departamento de recursos humanos de una empresa grande) estando fuera del equipo es muy difícil que genere la confianza con el equipo para que le compartan incidencias y problemas. Tampoco tengo claro que hace con su tiempo, en un estudio indie no creo que sea algo que te ocupe 40 horas a la semana.

- ¿Crees conveniente, programar un tiempo para el estudio de una tarea y un tiempo de revisión antes de considerarla finalizada?

Sí, imprescindible para reducir la cantidad de “rework” y bugs que aparecen posteriormente.

- ¿Ves conveniente limitar las secciones de Estudio y Revisión en número de personas que la realizan?

Depende del tamaño del equipo. A priori intentaría que todo el personal participe en ello, pero si pasamos a tener equipos de más de 10 personas es conveniente dividir el trabajo y que vaya una o dos personas de cada área de expertise (programador de cliente, backend, artista, etcétera).

- ¿Se ha subdividido una milestone en tres stages diferentes, consideras conveniente una división tan concreta o es mejor adaptarse a la longitud de la milestone con más o menos stages?

Al no tener duración fija cada "stage" parece arbitrario hacer esa división fija de 3 stages.

- ¿Consideras más óptimo hacer reuniones cada dos días en vez de una al día para conocer el progreso?

Si el equipo tiene pocas dependencias entre ellos se puede incluso reducir a dos semanales. Si hay muchas dependencias se puede aumentar a diario.

- ¿Consideras óptimo limitar las reuniones a días específicos?

Sí, es bueno que la gente tenga en su calendario todas las reuniones previstas con antelación. Si surge un imprevisto siempre se puede convocar una reunión específica. Hay que encontrar cual es el pulso/ritmo del proyecto.

- ¿Utilizarías esta metodología con tu equipo? ¿Por qué?

No lo usaría. En esta metodología se vuelve al Project Manager clásico que decide sobre el producto y sobre cómo se implementa. Mi experiencia es que hay que separar dos roles muy claros:

El *Product Manager* (Según el sector tiene distintos nombres como *Product Lead, Product Owner, Head of Product, Lead Game Designer*, etcétera): Es la persona que tiene la visión del producto y mantiene el contacto con el cliente (ya sea Publisher o usuario final)

El *Project Manager* (Según el sector o metodología se llamará: *Scrum Master*, *Agile Coach, Producer, Delivery Manager*, etcétera): Es la persona que se preocupa que se aplique la metodología de desarrollo de producto elegida y de facilitar las distintas reuniones para conseguir los objetivos y la visión del Product Manager.

4.Tabla analisis entrevistas

	Javier	María	Ivan
Num Empleados	49	12	25
¿Proyecto en desarrollo?	Port de la plataforma web a applicación movil.	Confidencial	Movil / Multiplayer PC
Fase de desarrollo	Beta	Soft-Launch	Softlaunch / 1 año de desarrollo aun
¿Utilización metodología?	Si	Si	Si
¿Metodología?	Scrum "Lite"	Scrum+Kanban+Elementos espontaneos	Kanban
¿Modificada?	Si pequeña modificación para dar flexibilidad al equipo para que se auto-organize dependiendo de sus costumbres.	Se adapta la matodologia dependiendo el momento concreto de desarrollo	No, kanban es robusto y flexible
¿Motivo?	La modificación viene sola y es la necesaria para adaptarse a las necesidades del proyecto.	La naturaleza del proyecto y las necesidades del equipo.	
¿En qué consiste?	Se flexibiliza la gestión de las tareas para que cada uno de los miembros.	El éxito esta en la comunicación y la sinergia entre los equipos. Se paso de considerar una milestone como feature completa. Se empezo a generar pequeños deliverables y los equipos se marcaban objetivos para no afectar a otras dependencias.	
¿Elementos metodología nueva?	Visibilidad de todo el proceso para todos los miebros implicados en la misma, para saber donde estas, donde estan los demás y hacia donde hay que ir.		
¿Team manager útil?	Si que es util y algo que se podria tener en cuenta pero no encaja dentro del target de la metodologia, es demasiado caro para un estudio indie.	La función de gestión del equipo si pero no dentro de un rol independiente. Incluirlo dentro del producer y fomentar con este los one-to-ones.	Es una función buena en teoria pero en cambio es muy difícil para implementar en un equipo pequeño.
¿Tiempo en el estudio y revisión?	Es necesaria una deadline, siempre que sea justa para la tarea pero es necesaria para estimar en todo momento el tiempo que se va a requerir para el proyecto.	Eliminar el Stage en pro "feature creation", buscar dependecias del equipo y hacer el estudio de manera previa ya en la milestone planning.	Si es necesario limitar el tiempo para realizar estas tareas.
¿Limitar Estudio y Revisión personas?	Es bueno compartir el conocimiento siempre, si es posible por la configuración del equipo si es interesante, pero en el dia a dia igual es difícil de implementar.	No limitar, involucrar a todo el equipo, antes del poker realizar el estudio de manera conjunta.	Depende del tamaño del equipo. Si este super las 10 personas si que es recomendable dividirlo sino es mejor que todo el mundo participe.
¿Sub división milestone?	Dependerá del proyecto, si es un proyecto sencillo si que se podria visualizar el camino para lograr el objetivo. En cambio si es un proyecto difícil si que es muy posible que se compliquen y no sea realista subdividir las milestones de esta manera.	No es mala idea, por otro lado se deberia de considerar las dependencias antes de la división. Usar un enfoque más agil escogiendo dividir o no dependiendo de la milestone.	Seria recomendable no limitarlo ya que no tienen una duración fija los stages.
¿BiDaily?	No, mejor una daily bien acotada, es verdad que una reunión mal estructurada es una perdida de tiempo. Es importante reservar un tiempo para el equipo y para saber en que esta trabajando.	Hacer Daily, hacer esta reunión aunque se vea inutil fomenta la comunicación entre el equipo y se acaba conociendo información de todo tipo.	Dependiendo de las dependencias que el equipo tenga entre si, si son pocas incluso se podria reducir.
¿Dias concretos para reunión?		Más que limitar las reuniones por dia mejor balancear entre espontaneas y programadas. Se tiene que fomentar la cominicación entre el equipo pero tambien programar reuniones donde especificar muy bien de que se va a hablar para mostrar el avance de el equipo.	Si que es recomendable que las personas tengan un calendario con las reuniones siempre pudiendo hacer reuniones de manera espontanea. Para esto lo mejor es encontrar el ritmo del proyecto.
¿Utilizarias esta metodología?	No. Elementos interesantes pero se tendria que probar, en un equipo indie no se pueden aplicar elementos estricto no merece la pena. Hay que adaptar siempre la metodologia al equipo y no al reves.	No. Por falta de gestión de las dependencias entre los equipo. No se puede avanzar de manera lineal sin tener en cuenta estas dependencias. No tenerlas en cuenta puede provocar un cuello de botella y mal estar en el equipo.	No. Seria recomendable separar los roles que tiene el proyect manager en dos personas diferentes para no volver al rol más clasico.

5. Puntos clave extraidos

No subdividir las milestones, variar las milestones "feature cration"
Hacer Daily
Adaptar la metodologia al equipo
Promover la comunicación
Suprimir el rol de Team manager y encargar la tarea al producer
Dependencias de las tareas
Limitar tiempos

6. Tareas reales del proyecto de tercer curso