TITULO DEL PROYECTO:	JUEGO VR HTC VIVE: NOT ALONE…

[Pol Puigdomenech]

Contenido

1 Release Product Backlog

Funcionalidad	Descripción
Compatibilidad con HTC Vive	Este punto es el más importante de todos ya que quiero que el juego sea para HTC Vivie. Básicamente que el juego soporte las gafas de realidad virtual HTC Vive.
Ambientación	El ambiente tiene que dar miedo, es decir, las habitaciones tienen que estar oscuras o tener una luz muy tenue. Los muebles deberían ser viejos, la casa en general debería estar en mal estado, las paredes podrían tener manchas, incluso alguna frase escrita con sangre.
Enigmas	Enigmas para poder ir pasando las fases del juego se tienen que ir resolviendo diferentes enigmas. La intención de los enigmas no es hacer pensar mucho al usuario, ya que sino el juego pierde la dinámica. Por ejemplo, un enigma podría ser solo introducir un código que previamente te has encontrado, otro enigma podría ser abrir una puerta con una llave que has encontrado previamente.
Sustos/miedo	En el juego tiene que haber sustos, ya sea de forma auditiva (algún ruido fuerte de golpe, un “screamer”, que justo al entrar a una habitación se cierre la puerta atrás tuyo de golpe.
Interactuar con el mapa	Poder abrir puertas, activar botones, coger objetos.
Buenos gráficos	El punto principal del proyecto es la jugabilidad, aunque no estaría mal poder implementar unos gráficos bastante realistas. Texturas de realismo en HD
Movimiento lo más real posible (buena jugabilidad)	Poder usar los mandos para agarrar cosas, para apuntar hacia una dirección, mover la cabeza para mover la cámara. Intentar que sea lo más sumergible posible.
Menú para guardar partida	Menú para poder guardar y cargar partidas (no creo que vaya a ser realmente necesario del todo ya que va a ser un juego bastante corto el cual te lo puedes pasar en unos 20 minutos)
Pequeños detalles finales	Detalles que no afectan en nada al juego pero que estaría bien poder implementar. Por ejemplo, que en algunas ocasiones se oigan ruidos a las habitaciones de al lado, que cuando la linterna tenga poca batería la luz parpadee,
Easter Egg	Insertar algún Easter Egg en el juego.

2 Diseño técnico

TECNOLOGIA

Voy a utilizar Unity 3D, voy a programar en C#. Utilizare este lenguaje porque según me he estado informando es de los mejores para utilizar el Unity, aparte mucha documentación sobre funciones de las gafas VR en Unity la he encontrado en C#, además en clase hacemos C# con lo cual no lo voy a tener que aprender del todo desde 0.

Utilizare una API de las Steam VR para que Unity sea compatible con las gafas VR y las pueda usar con Steam VR. A partir de ahí empezare a configurar los controles. Utilizare esta API ya que es la oficial de Steam y es la mas recomendada.

HERRAMIENTAS

Voy a necesitar:

- Steam VR: Para poder calibrar y usar las gafas VR es necesario este programa
- Unity 3D: Es el motor para crear el juego, he escogido este porque ya tengo algo de experiencia previa y a mi parecer es uno de los mas adecuados para empezar a realizar juegos cuando eres principiante
-

ARQUITECTURA DEL PROYECTO

Todos los datos se guardan en local, donde el usuario tenga instalado el juego. No es necesario ninguna red ni ninguna base de datos.

ALMACENAMIENTO DE LA INFORMACIÓN

La información se va a almacenar en el disco duro del usuario ya que al ser un juego no se necesita compartir ningún tipo de datos ni tampoco es necesario guardar ningún dato en la nube.

SEGURIDAD DE LA INFORMACIÓN

La información va a estar guardada de forma local en el ordenar del usuario, así que la seguridad de la cual va a disponer depende del usuario. Si el usuario tiene un antivirus, tiene contraseña en el ordenador, etc… va a ser mas seguro que si no tiene nada de eso. Aunque cuando el usuario guarde la partida no se van a guardar ningún tipo de datos comprometedores ya que solo se guarda la partida con el progreso actual, nada más.

MODELO DE DATOS

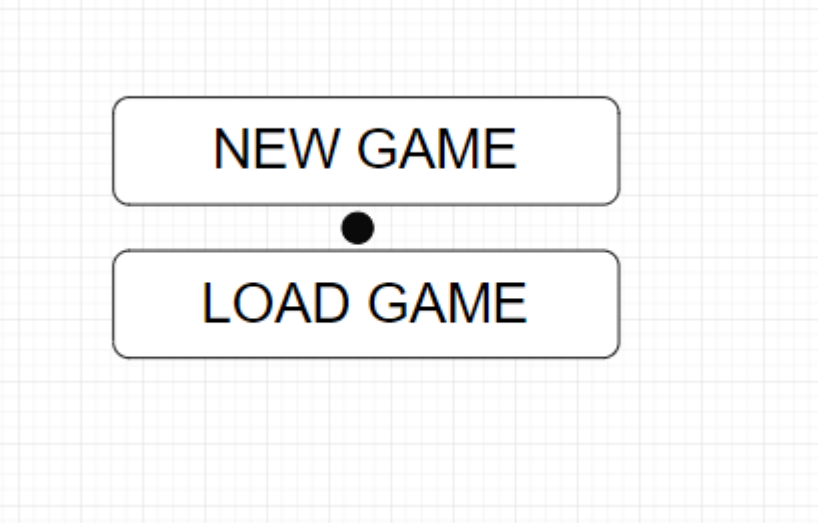
Se va a guardar donde el usuario tenga instalado el juego.

OTROS:

Cada persona tendrá sus partidas guardadas en local, no se van a compartir datos entre usuarios, ya que no es necesario. El sistema operativo para el juego va a ser Windows.

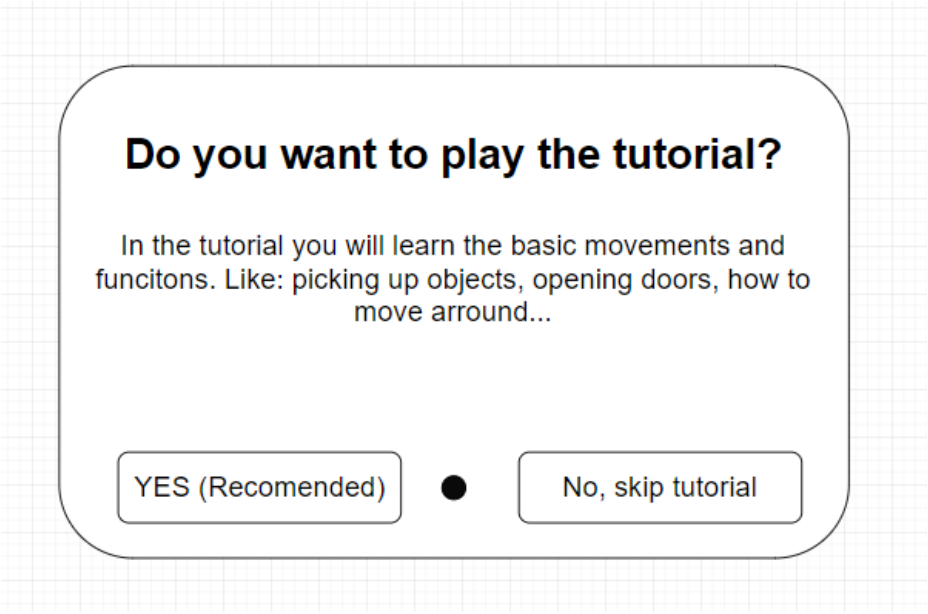
3 Diseño interfaz gráfica

Nota: Al ser un juego para las gafas VR el menú va a ser en 360 grados, por lo que no puedo poner el menú como tal.



La idea del menú seria que se vieran estas dos opciones. Al ser un juego VR el usuario puede mover la cabeza hacia los lados. Lo que he pensado es poner de fondo la sala principal (en la que comienza la historia) difuminada y luego el menú delante de la persona. Para escoger la opción tienes que mirar donde quieres escoger, es decir, si quieres New Game tienes que mirar a New Game y pulsar un botón. El circulo negro sería el “puntero”, es decir, en el menú verías un puntito negro todo el rato en el medio y seria con el que escogerías la opción.

Si le das a New Game se crea una nueva partida y te aparece un mensaje preguntándote si quieres hacer el tutorial. El método de selección de opción seria el mismo. Si le das que si comenzaría el tutorial y si le das que no empezarías la partida directamente.



En cambio, si le das a Load Game se cargaría una lista con las partidas guardadas y desde allí podrías escoger la partida que quieres cargar con el mismo método que en los menús anteriores.

4 Release plan / RoadMap

SPRINT	SUBTAREAS	FECHA INICIO	FECHA FIN
SPRINT 1 INVESTIGACIÓN DE HERRAMIENTAS	Investigación Unity 3D Investigación Github Investigación HTC Vive en Unity 3D Investigación de creación de mapas en Unity 3D	9-11	16-11
SPRINT 2 PREPARACIÓN DE LA INFRAESTRUCTURA DE LA APLICACIÓN	Instalación y configuración de Unity 3D Instalación y configuración de las gafas VR Instalación de drivers, plugins etc… necesarios para usar las VR en Unity Configuración Github	16-11	30-11
SPRINT 3 COMPATIBILIDAD DE LAS GAFAS VR CON UNITY	Sincronización de las gafas con Unity 3D Configuración del movimiento - Movimiento del personaje - Movimiento de la cámara Ejecución de pruebas	30-11	14-12

SPRINT 4 CREACION DEL MAPA	Definir el mapa Crear las habitaciones Crear las luces Insertar objetos en el mapa de decoración Configuración de la interactuación con el mapa (abrir puertas, coger objetos...) Asignar las texturas	14-12	11-1
SPRINT 5 CREACIÓN DE ENIGMAS	Pensar los enigmas Implementar los enigmas Enlazar los enigmas con el mapa Enlazar los enigmas con la trama de la historia	11-1	1-2
SPRINT 6 SUSTOS Y AMBIENTACIÓN	Implementar los elementos de la ambientación - Luces - Muebles - Ruidos Implementar sustos - Ruidos - Screamers - Movimiento de objetos	1-2	22-2
SPRINT 7 CREACION DEL TUTORIAL	Crear un tutorial Implementar un escenario para el tutorial Implementar los mensajes para el tutorial Implementar los objetos para el tutorial	22-2	15-3
SPRINT 8 MENU Y ULTIMOS DETALLES	Creación de un menú Implementar la carga y el guardado de datos Implementar mejoras en los gráficos (texturas) Implementar mejoras en la ambientación y la decoración Implementar un pequeño Easter Egg	15-3	5-4
SPRINT 9 PUESTA A PUNTO Y RESOLUCION DE BUGS	Resolver los últimos bugs Exportar el proyecto a un juego funcional	5-4	Entrega

5 Diseño de pruebas

SPRINT 1:

- Pruebas de compatibilidad: Conseguir informarme sobre todos los programas necesarios para realizar el proyecto

SPRINT 2:

- Pruebas de usabilidad: Comprobar que todos los programas están instalados y funcionan de forma correcta
- Pruebas de compatibilidad: Comprobar la compatibilidad de los dispositivos con el software
- Pruebas en dispositivos: Configurar todos los dispositivos
- Pruebas de instalación: Instalar todos los programas, api, plugins, drivers…

SPRINT 3:

- Pruebas de compatibilidad, pruebas de dispositivos, pruebas de interfaz: Comprobar que las gafas funcionan en Unity 3D para eso se van a tener que probar las gafas
- Pruebas de jugabilidad: Configurar las gafas para que se puedan realizar todas las acciones con las gafas VR
- Pruebas de usabilidad: Comprobar que se pueden usar las gafas en el juego

SPRINT 4:

- Pruebas visuales y gráficas: Crear el mapa, comprobar que el mapa
- Pruebas usabilidad: Poder interactuar con los objetos del mapa. Concretamente: poder abrir puertas, coger objetos, apretar botones…

SPRINT 5:

- Pruebas de usabilidad y jugabilidad: Que los enigmas se puedan resolver, implementar los elementos para que los enigmas se puedan resolver
- Pruebas graficas y visuales: Implementar los elementos necesarios (códigos, botones, llaves…)

SPRINT 6:

- Pruebas visuales y gráficas: Implementar toda la decoración, todos los modelos 3D, todos los audios. Configurar los elementos para que pasen en determinadas circunstancias
- Pruebas de usabilidad y jugabilidad: Comprobar que no afecta negativamente a la jugabilidad y que no hay ningún problema de compatibilidad con las gafas VR
- Pruebas de compatibilidad y en dispositivos: Comprobar que todo funciona correctamente con las gafas VR

SPRINT 7:

- Pruebas de interfaz, graficas, y visuales: Crear todos los textos necesarios para el tutorial, implementar los objetos y los elementos necesarios para que el tutorial funcione correctamente
- Pruebas de usabilidad y jugabilidad: Comprobar la jugabilidad y la utilidad del tutorial, es decir, que el tutorial explique lo esencial para poder jugar.

SPRINT 8:

- Pruebas de interfaz, gráficas y visuales: Creación del menú
- Pruebas de carga de datos: Comprobar que los datos se pueden cargar
- Pruebas de guardado de datos: Comprobar que los datos se pueden guardar
- Pruebas en dispositivos: Comprobar que funciona con el ordenador
- Pruebas de usabilidad: Comprobar que con el menú se puede seleccionar la opción y asignarle la respectiva acción (guardar/cargar)

SPRINT 9:

- Prueba de usabilidad: Acabar de solucionar los bugs finales para que el juego este ya listo para ser lanzado
- Prueba de exportación: Exportar el proyecto como un juego
- Prueba de jugabilidad: Comprobar que no se ha perdido la jugabilidad
- Prueba de compatibilidad: Comprobar que sigue siendo compatible con las gafas y que se pueden usar para jugar

6 Herramientas a utilizar para el seguimiento y control en la ejecución del proyecto

Utilización de GitHub para guardar el proyecto y llevar un control de versiones.

7 Estimación costes del proyecto

Costes:

- Hardware:
  - PC para crear el juego: 1000€
  - Gafas HTC Viv, mandos y estaciones de movimiento: 500€
- Software:
  - SteamVR: Gratis
  - Unity 3D: Gratis
  - GitHub: Gratis
  - Git Bash: Gratis
- Marketing:
  - Anuncios (redes sociales, foros, Google ads…): 2000€
- Distribución: Gratis (Al ser un juego la gente lo descarga desde su casa)

8 Análisis de riesgos

Tabla. Análisis de riesgos

Riesgo	Categoría	Probabilidad
Los espacios no están disponibles en el momento necesario.	Ambiente/Infraestructura de Desarrollo	100%
Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»).	Elaboración de la Planificación	60%
Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.	Elaboración de la Planificación	40%
La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado.	Ambiente/Infraestructura de Desarrollo	40%
La falta de motivación y de moral reduce la productividad.	Personal	30%
No se puede construir un producto de tal envergadura en el tiempo asignado.	Elaboración de la Planificación	30%
El esfuerzo es mayor que el estimado (por líneas de código, número de puntos función, módulos, etc.).	Elaboración de la Planificación	30%
El trabajo con un entorno software desconocido causa problemas no previstos.	Producto	30%
El trabajo con un entorno hardware desconocido causa problemas imprevistos.	Producto	30%
Un diseño demasiado complejo exige tener en cuenta complicaciones innecesarias e improductivas en la implementación.	Diseño e Implementación	20%
El personal necesita un tiempo extra para aprender un lenguaje de programación nuevo.	Personal	10%

Problemas más importantes:

- Los espacios no están disponibles en el momento necesario: Al tener que desarrollar un juego con las gafas de realidad virtual HTC Vive, necesitaría poder probar las partes del código para comprobar que funciona correctamente. Cuando programe en mi casa no habrá problema, pero cundo programe en clase no podre comprobar nada del código hasta que llegue a casa. La solución que he encontrado es intentar hacer las partes de los ítems en clase, sino hacer partes de código y probarlas al llegar a casa, o sino crear un personaje que se pueda mover con las teclas “WASD” para al menos poder ver que el código compila, y que todo funciona y se ve bien.
- Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»): A la hora de planificar el proyecto puede que haya puesto algunos requisitos que después resulten demasiado difícil de implementar en el juego final. Es decir, se puede implementar de una forma sencilla pero no al 100% como me gustaría. La solución seria no quedarme atacado en un aspecto y seguir con otro.
- Un retraso en una tarea produce retrasos en cascada en las tareas dependientes: Es muy probable que, debido a deberes, exámenes, problemas con el proyecto u otras circunstancias algún Sprint no se realice en las fechas acordadas, es decir, que algún Sprint a lo mejor se acaba un poco mas tarde de lo esperado. La solución más fácil es intentar llevar las cosas al día y si algo se atrasa intentar volver a recuperar el ritmo inicial del proyecto.

- La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado: Es posible que haya complicaciones al usar Unity 3D con las gafas de realidad virtual, tanto de compatibilidad, de jugabilidad, de programación, o incluso diseño. La solución seria buscar información por internet i intentar usar alguna API.

9 Resumen ejecutivo

En este ítem voy a entregar un Product Backlog con todos los requisitos del juego. En rojo están los obligatorios para que el juego funcione, en amarillo los que estaría muy bien tener, aunque no afecten directamente al juego, es decir, sin ellos se podría jugar igual al juego, y en verde los requisitos finales, los cuales serien como los últimos detalles opcionales, es decir, si sobra tiempo y se pueden implementar perfecto, sino no pasa nada. El Backlog consiste de una tabla con todos los requisitos y una explicación del mismo.

Después he definido el diseño técnico, el cual consiste en definir diferentes campos. Define que tecnologías voy a usar y porque, que herramientas voy a necesitas y porque, la arquitectura del proyecto, el almacenaje de datos del proyecto, la seguridad de estos datos y el modelo de datos que voy a usar para el proyecto. En mi caso los datos se van a guardar en local, es decir en el disco duro del usuario, por lo cual muchos aspectos dependen del usuario como por ejemplo la seguridad de estos datos. Aunque como en mi caso los únicos datos que se van aguardar serán solo el progreso de la partida, eso significa que no hay datos de riesgo.

A continuación, se define el diseño de la interfaz gráfica, esto se compone del sketch, wireframe y el mockup.

El siguiente punto nos explica el Release Plan/Road Map. En este apartado se definen todos los Sprint que se van a realizar en el proyecto. Cada Sprint tiene un apartado donde se describe todas las acciones que se van a realizar y las fechas en las cuales se va a realizar.

A continuación, se define el diseño de pruebas, en el cual se establecen todas las pruebas que se van a realizar en cada uno de los Sprint y se define cada prueba en que consiste.

Después de esto, se establecen las herramientas para el seguimiento y control en la ejecución del proyecto, apartado en el cual he dicho que voy a utilizar GitHub para tener un control de versiones y así ir viendo el avance del proyecto.

El siguiente punto es el de estimación de costes, en este punto se hace una estimación de los costes que comporta el proyecto, por ejemplo, costes en hardware o software, en marketing, distribución, los costes de cada Sprint y del total…

Para finalizar se realiza un análisis de riesgos en el cual se rellena una tabla con los riesgos que puede tener mi proyecto y con un porcentaje de probabilidad. Para acabar se proponen unas soluciones para los riesgos mas importantes que puede sufrir mi proyecto.

10 ANEXO I: Listado riesgos más habituales en el desarrollo de proyectos de software

A. Elaboración de la Planificación
A.1. Las definiciones de la planificación, de los recursos y del producto han sido impuestas por el cliente o un directivo superior, y no están equilibradas.
A.2.Planificación optimista, «mejor caso» (en lugar de realista, «caso esperado»).
A.3.La planificación no incluye tareas necesarias.
A.4.La planificación se ha basado en la utilización de personas específicas de un equipo, pero estas personas no están disponibles.
A.5.No se puede construir un producto de tal envergadura en el tiempo asignado.
A.6.El producto es más grande que el estimado (en líneas de código, en el número de puntos función, o en relación con el tamaño del proyecto anterior).
A.7.El esfuerzo es mayor que el estimado (por líneas de código, número de puntos función, módulos, etc.).
A.8.La reestimación debida a un retraso en la planificación es demasiado optimista o ignora la historia del proyecto.
A.9.La presión excesiva en la planificación reduce la productividad.
A.10.La fecha final ha cambiado sin ajustarse al ámbito del producto o a los recursos disponibles.
A.11.Un retraso en una tarea produce retrasos en cascada en las tareas dependientes.
A.12. Las áreas desconocidas del producto llevan más tiempo del esperado en el diseño y en la implementación

B. Organización y Gestión
B.1.El proyecto carece de un promotor efectivo en los superiores.
B.2.El proyecto languidece demasiado en el inicio difuso.
B.3.Los despidos y las reducciones de la plantilla reducen la capacidad del equipo.
B.4.Dirección o marketing insisten en tomar decisiones técnicas que alargan la planificación.
B.5.La estructura inadecuada de un equipo reduce la productividad.
B.6.El ciclo de revisión/decisión de la directiva es más lento de lo esperado.
B.7.El presupuesto varía el plan del proyecto.
B.8.La dirección toma decisiones que reducen la motivación del equipo de desarrollo.
B.9.Las tareas no técnicas encargadas a terceros necesitan más tiempo del esperado (aprobación del presupuesto, aprobación de la adquisición de material, revisiones legales, seguridad, etc.).
B.10.La planificación es demasiado mala para ajustarse a la velocidad de desarrollo deseada.
B.11.Los planes del proyecto se abandonan por la presión, llevando al caos y a un desarrollo ineficiente.
B.12.La dirección pone más énfasis en las heroicidades que en informarse exactamente del estado, lo que reduce su habilidad para detectar y corregir problemas.

C.Ambiente/Infraestructura de Desarrollo
C.1.Los espacios no están disponibles en el momento necesario.
C.2.Los espacios están disponibles pero no son adecuados (por ejemplo, falta de teléfonos, cableado de la red, mobiliario, material de oficina, etc.).
C.3.Los espacios están sobre utilizados, son ruidosos o distraen.
C.4.Las herramientas de desarrollo no están disponibles en el momento deseado.
C.5.Las herramientas de desarrollo no funcionan como se esperaba; el personal de desarrollo necesita tiempo para resolverlo o adaptarse a las nuevas herramientas.
C.6.Las herramientas de desarrollo no se han elegido en función de sus características técnicas, y no proporcionan las prestaciones previstas.
C.7.La curva de aprendizaje para la nueva herramienta de desarrollo es más larga de lo esperado.

D. Usuarios Finales
D.1.Los usuarios finales insisten en nuevos requisitos
D.2.En el último momento, a los usuarios finales no les gusta el producto, por lo que hay que volver a diseñarlo y a construirlo.
D.3.Los usuarios no han realizado la compra del material necesario para el proyecto y, por tanto, no tienen la infraestructura necesaria.
D.4.No se ha solicitado información al usuario, por lo que el producto al final no se ajusta a las necesidades del usuario, y hay que volver a crear el producto.

E.Cliente
E.1.El cliente insiste en nuevos requisitos.
E.2.Los ciclos de revisión/decisión del cliente para los planes, prototipos y especificaciones son más lentos de lo esperado.
E.3.El cliente no participa en los ciclos de revisión de los planes, prototipos y especificaciones, o es incapaz de hacerlo, resultando unos requisitos inestables y la necesidad de realizar unoscambios que consumen tiempo.
E.4.El tiempo de comunicación del cliente (por ejemplo, tiempo para responder a las preguntas para aclarar los requisitos) es más lento del esperado.
E.5.El cliente insiste en las decisiones técnicas' que alargan la planificación.
E.6.El cliente intenta controlar el proceso de desarrollo, con lo que el progreso es más lento de lo esperado.
E.7.Los componentes suministrados por el cliente no son adecuados para el producto que se está desarrollando, por lo que se tiene que hacer un trabajo extra de diseño e integración.
E.8.Los componentes suministrados por el cliente tienen poca calidad, por lo que tienen que hacerse trabajos extra de comprobación, diseño e integración.
E.9.Las herramientas de soporte y entornos impuestos por el cliente son incompatibles, tienen un bajo rendimiento o no funcionan de forma adecuada, con lo que se reduce la productividad.
E.10.El cliente no acepta el software entregado, incluso aunque cumpla todas sus especificaciones.
E.11.El cliente piensa en una velocidad de desarrollo que el personal de desarrollo no puede alcanzar.

F. Personal Contratado
F.1.El personal contratado no suministra los componentes en el período establecido.
F.2.El personal contratado proporciona material de una calidad inaceptable, por lo que hay que añadir un tiempo extra para mejorar la calidad.
F.3.Los proveedores no se integran en el proyecto, con lo que no se alcanza el nivel de rendimiento que se necesita.

G.Requisitos
G.1.Los requisitos se han adaptado, pero continúan cambiando.
G.2.Los requisitos no se han definido correctamente. y su redefinición aumenta el ámbito del proyecto.
G.3.Se añaden requisitos extra.
G.4.Las partes del proyecto que no se han especificado claramente consumen más tiempo del esperado.

H.Producto
H.1.Los módulos propensos a tener errores necesitan más trabajo de comprobación, diseño e implementación.
H.2.Una calidad no aceptable requiere de un trabajo de comprobación, diseño e implementación superior al esperado.
H.3.Utilizar lo último en informática alarga la planificación de forma impredecible.
H.4.El desarrollo de funciones software erróneas requiere volver a diseñarlas y a implementarlas.
H.5.El desarrollo de una interfaz de usuario inadecuada requiere volver a diseñarla y a implementarla.
H.6.El desarrollo de funciones software innecesarias alarga la planificación.
H.7.Alcanzar el ámbito del producto o las restricciones de velocidad requiere más tiempo del esperado, incluyendo el tiempo para volver a diseñar e implementar.
H.8.Unos requisitos rígidos de compatibilidad con el sistema existente necesitan un trabajo extra de comprobación, diseño e implementación.
H.9.Los requisitos para crear interfaces con otros sistemas, otros sistemas complejos, u otros sistemas que no están bajo el control del equipo de desarrollo suponen un diseño, implementación y prueba no previstos.
H.10.El requisito de trabajar con varios sistemas operativos necesita más tiempo del esperado.
H.11.El trabajo con un entorno software desconocido causa problemas no previstos.
H.12.El trabajo con un entorno hardware desconocido causa problemas imprevistos.
H.13.El desarrollo de un tipo de componente nuevo para la organización consume más tiempo del esperado.
H.14.Depender de una tecnología que aún está en fase de desarrollo alarga la planificación.

I.Fuerzas mayores
I.1.El producto depende de las normativas del gobierno, que pueden cambiar de forma inesperada.
I.2.El producto depende de estándares técnicos provisionales, que pueden cambiar de forma inesperada.

J.Personal
J.1.La contratación tarda más de lo esperado.
J.2.Las tareas preliminares (por ejemplo, formación, finalización de otros proyectos, adquisición de licencias) no se han completado a tiempo.
J.3.La falta de relaciones entre la dirección y el equipo de desarrollo ralentiza la toma de decisiones.
J.4.Los miembros del equipo no se implican en el proyecto, y por lo tanto no alcanzan el nivel de rendimiento deseado.
J.5.La falta de motivación y de moral reduce la productividad.
J.6.La falta de la especialización necesaria aumenta los defectos y la necesidad de repetir el trabajo.
J.7.El personal necesita un tiempo extra para acostumbrarse a trabajar con herramientas o entornos nuevos.
J.8.El personal necesita un tiempo extra para acostumbrarse a trabajar con hardware nuevo.
J.9.El personal necesita un tiempo extra para aprender un lenguaje de programación nuevo.
J.10.El personal contratado abandona el proyecto antes de su finalización.
J.11.Alguien de la plantilla abandona el proyecto antes de su finalización.

J.12.La incorporación de nuevo personal de desarrollo al proyecto ya avanzado, y el aprendizaje y comunicaciones extra imprevistas reducen la eficiencia de los miembros del equipo existentes.
J.13.Los miembros del equipo no trabajan bien juntos.
J.14.Los conflictos entre los miembros del equipo conducen a problemas en la comunicación y en el diseño, errores en la interfaz y tener que repetir algunos trabajos.
J.15.Los miembros problemáticos de un equipo no son apartados, influyendo negativamente en la motivación del resto del equipo.
J.16.Las personas más apropiadas para trabajar en el proyecto no están disponibles.
J.17.Las personas más apropiadas para trabajar en el proyecto están disponibles, pero no se pueden incorporar por razones políticas o de otro tipo.
J.18.Se necesitan personas para el proyecto con habilidades muy específicas y no se encuentran.
J.19.Las personas clave sólo están disponibles una parte del tiempo.
J.20.No hay suficiente personal disponible para el proyecto.
J.21.Las tareas asignadas al personal no se ajustan a sus posibilidades.
J.22.El personal trabaja más lento de lo esperado.
J.23.El sabotaje por parte de la dirección del proyecto deriva en una planificación ineficiente e inefectiva.
J.24.El sabotaje por parte del personal técnico deriva en una pérdida de trabajo o en un trabajo de poca calidad, por lo que hay que repetir algunos trabajos.

K.Diseño e Implementación
K.1.Un diseño demasiado sencillo no cubre las cuestiones principales, con lo que hay que volver a diseñar e implementar.
K.2.Un diseño demasiado complejo exige tener en cuenta complicaciones innecesarias e improductivas en la implementación.
K.3.Un mal diseño implica volver a diseñar e implementar.
K.4.La utilización de metodologías desconocidas deriva en un periodo extra de formación y tener que volver atrás para corregir los errores iniciales cometidos en la metodología.
K.5.El producto está implementado en un lenguaje de bajo nivel (por ejemplo, ensamblador) y la productividad es menor de la esperada.
K.6.No se puede implementar la funcionalidad deseada con el lenguaje o bibliotecas utilizados: el personal de desarrollo tiene que utilizar otras bibliotecas, o crearlas él mismo para conseguir la funcionalidad deseada.
K.7.Las bibliotecas de código o clases tienen poca calidad, y generan una comprobación extra, corrección de errores y la repetición de algunos trabajos.
K.8.Se ha sobreestimado el ahorro en la planificación derivado del uso de herramientas para mejorar la productividad.
K.9.Los componentes desarrollados por separado no se pueden integrar de forma sencilla, teniendo que volver a diseñar y repetir algunos trabajos.

L.Proceso
L.1.La burocracia produce un progreso más lento del esperado.
L.2.La falta de un seguimiento exacto del progreso hace que se desconozca que el proyecto esté retrasado hasta que está muy avanzado.
L.3.Las actividades iniciales de control de calidad son recortadas, haciendo que se tenga que repetir el trabajo.
L.4.Un control de calidad inadecuado hace que los problemas de calidad que afectan a la planificación se conozcan tarde.
L.5.La falta de rigor (ignorar los fundamentos y estándares del desarrollo de software) conduce a fallos de comunicación, problemas de calidad y repetición del trabajo. Un consumo de tiempo innecesario.
L.6.El exceso de rigor (aferramiento burocrático a las políticas y estándares de software) lleva a gastar más tiempo en gestión del necesario.
L.7.La creación de informes de estado a nivel de directiva lleva más tiempo al desarrollador de lo esperado.
L.8.La falta de entusiasmo en la gestión de riesgos impide detectar los riesgos más importantes del proyecto.
L.9.La gestión de riesgos del proyecto software consume más tiempo de lo esperado