04/02/2026

TAREA 5.1 - Metodologías de desarrollo de software

Nombre y apellidos: Cristina Sandoval Laborde

Curso: 2ºDAM

Asignatura: Desarrollo de interfaces

Índice

1. Límites reales del modelo en cascada

Sin definir qué es el desarrollo en cascada, responde:

a) Describe **dos situaciones reales** (no teóricas) en las que el uso de este modelo pueda provocar:

•retrasos,

•sobrecostes,

•o un producto que no encaje con el cliente.

1.El desarrollo de una aplicación para gestión de criptomonedas. Los requisitos se cierran en enero, pero en marzo el gobierno aprueba una nueva ley de transparencia fiscal que obliga a cambiar cómo se registran las transacciones. Debido a la rigidez del modelo, integrar este cambio requiere detener la maquinaria y retroceder a la fase de análisis para redefinir documentos, lo que genera **retrasos masivos** y **sobrecostes** al tener que rehacer trabajo que ya estaba planificado como "terminado".

2. Una clínica encarga un software para que los médicos anoten diagnósticos. El análisis de requisitos se hace únicamente con los directores de la clínica (que no pasan consulta). Cuando el software llega a manos de los médicos meses después, estos descubren que la interfaz es demasiado lenta para una consulta real de urgencias. El resultado es un **producto que no encaja con el cliente final**, a pesar de haber cumplido estrictamente con el contrato inicial.

b) Elige una de esas situaciones y explica:

• ¿En qué fase aparece el problema?

El problema sale a la luz únicamente en la **fase de pruebas** (testing) o incluso en la de **mantenimiento**, que es cuando los usuarios reales interactúan por primera vez con el sistema funcional.

• ¿Por qué no se detecta antes?

Porque en este modelo el cliente no prueba el software de forma continua; solo valida documentos de texto en la fase de análisis. Al no existir entregas parciales o "sprints", no hay feedback intermedio que permita corregir el rumbo de la interfaz antes de que esté programada por completo.

• ¿Qué consecuencias tiene en fases posteriores del proyecto?

Detectar un error de concepto en la interfaz tan tarde dispara el **coste de reparación**. Corregir un fallo estructural en esta etapa puede ser muchísimo más caro que si se hubiera detectado en el diseño.

2. Uso real del modelo tradicional (búsqueda obligatoria)

Busca en internet una **fuente técnica o profesional** (artículo, caso real, documentación de empresa…) donde se justifique el uso de un modelo **tradicional o en cascada**.

Indica:

•Enlace a la fuente

https://instituteprojectmanagement.com/es/blog/que-es-la-metodologia-en-cascada-todo-lo-que-necesitas-saber/

•Tipo de proyecto donde se aplica

Proyectos de **Construcción** e Ingeniería de Infraestructuras

•Por qué en ese contexto los requisitos no cambian

En este contexto, la fuente justifica el uso del modelo en cascada porque el proyecto sigue un planteamiento lineal "extremadamente estable por naturaleza". Una vez que los requisitos del sistema han sido finalizados y el trabajo real comienza, **"no hay posibilidades de desviarse del plan original"**. Esto se debe a que las fases son secuenciales y dependientes; por ejemplo, no se puede avanzar sin completar la anterior y cualquier cambio posterior sería prohibitivo en términos de coste y viabilidad física.

•Relación con lo visto en clase.

Análisis de Requisitos: La fuente destaca que los requisitos deben ser "bien conocidos y estar establecidos", lo que coincide con lo que indica el temario cuando dice en la fase de análisis los requisitos **"tienen que estar muy claros desde el inicio"** ya que no se espera que cambien.

Impacto del Coste: La fuente señala que el modelo es "costoso e inflexible" si se intentan hacer cambios tarde. Esto se relaciona con la gráfica de **Coste de reparación de errores** del tema, que muestra cómo un error detectado en la fase de producción puede ser **640 veces más caro** que en la fase de codificación.

Idoneidad del Modelo: Finalmente, el artículo profesional concluye que es el modelo adecuado cuando los requisitos son estáticos, reafirmando la "Idea clave" de los apuntes: **"Este tipo de metodología funciona bien cuando los requisitos están muy claros desde el principio y apenas cambian"**.

3. Valores ágiles aplicados a situaciones reales

Los valores ágiles no siempre encajan bien entre sí.

a) Elige **dos valores ágiles** que puedan entrar en conflicto en un proyecto real.

Colaboración con el cliente sobre negociación contractual: Se da importancia al contacto continuo y a que el feedback del cliente pueda provocar cambios durante el desarrollo.

Individuos e interacciones sobre procesos y herramientas: Se da más importancia a las personas del equipo y a su capacidad de comunicación que a seguir procesos rígidos.

b) Describe una situación concreta donde:

•cumplir uno dificulta cumplir el otro,

•el equipo tenga que tomar una decisión.

Un proyecto donde se desarrolla una aplicación de logística. El cliente es muy entusiasta y, para cumplir con el valor de **colaboración continua**, solicita asistir a todas las reuniones diarias del equipo (Daily Scrum de 15 minutos). Su intención es dar feedback en tiempo real sobre lo que se hizo ayer y proponer cambios inmediatamente al escuchar los avances.

El conflicto: Cumplir con esta colaboración dificulta enormemente las interacciones del equipo. Los desarrolladores, al sentirse observados por quien paga el proyecto, dejan de hablar con sinceridad sobre sus errores o problemas técnicos por miedo a dar una mala imagen. La reunión, que debería ser una interacción ágil entre individuos para coordinarse, se convierte en un proceso rígido de "rendición de cuentas" ante el cliente.

Decisión del equipo: El equipo debe decidir si permite la entrada del cliente para mantener la transparencia total o si protege su espacio interno para asegurar una comunicación sana.

c) Indica qué valor priorizarías y **por qué**, asumiendo claramente las consecuencias negativas de esa decisión.

En esta situación, yo priorizaría **Individuos e interacciones sobre procesos y herramientas**.

Según los principios ágiles, un equipo bien organizado y con interacciones sinceras suele dar mejores resultados que cualquier proceso, por muy colaborativo que sea. Si la presencia del cliente corta la comunicación del equipo, los problemas técnicos no se compartirán a tiempo y la calidad final del software caerá, por mucho feedback que dé el cliente. La excelencia técnica depende de que el equipo pueda interactuar sin barreras.

Consecuencias negativas asumidas:

- **Sensación de exclusión:** El cliente puede sentirse molesto o desconfiar del equipo al no permitírsele estar en "las mini sesiones " del proyecto, lo que podría tensar la relación.
- **Feedback retrasado:** Al no estar el cliente en la reunión diaria, sus correcciones llegarán más tarde (quizás en la reunión de revisión semanal), lo que podría obligar a rehacer trabajo.
- **Percepción de falta de transparencia:** El cliente podría pensar que el equipo oculta retrasos o errores graves al no dejarle participar en sus interacciones internas.

4. Principios ágiles y problemas de aplicación

Reflexión y ajuste regular (Principio 12)

a) Buscar en internet una **fuente profesional fiable** donde se describa:

https://www.paradigmadigital.com/techbiz/11-males-mas-comunes-retrospectivas-escalados/

•Una dificultad real al aplicar ese principio

El artículo describe la **"Falta de seguimiento de las acciones"**. La dificultad reside en que, aunque el equipo identifica problemas, en el día a día la presión por las entregas hace que las mejoras acordadas se olviden, cayendo en un ciclo de inacción.

•Una mala aplicación del mismo.

Se menciona la **"Retrospectiva ritualista o vacía"**. Ocurre cuando el equipo realiza la reunión simplemente porque el proceso (Scrum) lo dicta, pero sin una intención real de cambio, convirtiéndola en un trámite aburrido que no genera ningún ajuste en el comportamiento.

b) Resume con tus palabras:

•El problema descrito:

El artículo describe la **"Retrospectiva aburrida o ritualista"**. El equipo se junta solo por cumplir el calendario, pero no hay un compromiso real para cambiar. Se detectan problemas, pero nadie toma la responsabilidad de solucionarlos, lo que convierte la reflexión en una pérdida de tiempo.

•El contexto del proyecto.

Equipos que aplican Scrum pero donde las reuniones de mejora(retrospectva) se vuelven mecánicas.

c) Relaciona ese caso con lo visto en clase:

•¿Qué principio se incumple?

Principio 12 (Mejora basada en el feedback continuado).

•¿Por qué ocurre?

Ocurre cuando se priorizan los **procesos** (hacer la reunión porque toca) sobre los **individuos e interacciones** (hablar de verdad para mejorar).

•¿Qué efectos tiene en el equipo o el producto?

El equipo no evoluciona. Si el proceso no mejora, los errores de código detectados en la fase de **pruebas de aplicaciones** se repetirán en el siguiente ciclo, impidiendo reducir el **coste de reparación** (que en fases avanzadas es **640 veces más caro**).

d) Propón **una solución propia**, distinta a la de la fuente.

Implementar la **"Retrospectiva del Semáforo Técnico"**: Al final de cada reflexión, el equipo debe elegir una sola acción técnica (por ejemplo: "mejorar los tests unitarios de X módulo"). Se le asigna un color: rojo si no se ha hecho nada, amarillo si está en proceso y verde si se ha cumplido. Si el semáforo está en rojo al final de la semana, el equipo debe detenerse y dedicar las primeras 4 horas del lunes solo a esa mejora antes de seguir programando funcionalidades nuevas.

2.

a) Buscar en internet una **fuente profesional fiable** donde se describa:

https://www.bbva.es/finanzas-vistazo/agile/metodologia-agile/valores-principios-manifiesto-agil.html#principios-agiles

•Una dificultad real al aplicar ese principio

El artículo describe "**El miedo a la transparencia total**". Para muchos gestores de proyectos tradicionales, es difícil admitir que un proyecto va "al 0%" simplemente porque el software aún no se puede ejecutar, prefiriendo informar de que el "análisis está al 100%" para dar una falsa sensación de avance.

•Una mala aplicación del mismo.

Utilizar gráficos de barras o porcentajes de "horas trabajadas" para justificar el éxito de un proyecto ante el cliente, cuando en realidad el cliente no ha visto ni una sola pantalla que funcione de verdad. Esto oculta riesgos técnicos que solo aparecen cuando intentas ejecutar el código por primera vez.

b) Resume con tus palabras:

•El problema descrito:

La confusión entre "estar ocupado" y "avanzar". Se da más importancia a rellenar informes y completar fases de diseño que a tener un producto que el usuario pueda tocar y probar.

•El contexto del proyecto.

Proyectos donde se invierten meses en arquitectura y documentación sin escribir una línea de código funcional, lo que lleva a descubrir errores de diseño demasiado tarde.

c) Relaciona ese caso con lo visto en clase:

•¿Qué principio se incumple?

Principio 7: “Un equipo motivado es indicador de éxito del proyecto.”

•¿Por qué ocurre?

Ocurre porque se arrastra la mentalidad del modelo en cascada, donde el progreso se mide por el cumplimiento de fases (Análisis -> Diseño) en lugar de por incrementos de valor real para el cliente.

•¿Qué efectos tiene en el equipo o el producto?

Provoca una pérdida de confianza del cliente. Cuando llega la fecha de entrega y el software no funciona "porque falta integrarlo", el impacto organizativo es masivo y el coste de corregir esos fallos de integración de última hora es altísimo.

d) Propón **una solución propia**, distinta a la de la fuente.

Implementar el "Hito del Primer Clic": El equipo debe comprometerse a que, en la primera semana del proyecto, el cliente debe poder hacer "un clic" que ejecute una acción real (aunque sea mínima, como un login o una suma). A partir de ahí, cada semana se mide el progreso solo por la cantidad de "clics útiles" nuevos que se han añadido. Si no se puede hacer clic en algo nuevo, el progreso reportado esa semana es cero, obligando al equipo a no dejar tareas "a medias" o "pendientes de integrar"

5. Proyecto: Aplicación web de gestión de citas médicas

Condiciones del proyecto:

•El cliente quiere resultados visibles pronto.

•Los requisitos cambian con frecuencia.

•El equipo es pequeño y con poca experiencia.

•Existe presión por la calidad del código.

•El cliente no puede reunirse todas las semanas.

5.1 Evidencias del mundo profesional

Antes de elegir la metodología:

Busca en internet **una oferta de empleo real** o **una experiencia profesional documentada** donde se mencione explícitamente:

•Scrum,

•Kanban,

•o Extreme Programming.

Indica:

•Enlace

https://www.tecnoempleo.com/scrum-master-junior-teletrabajo-second-window/agile/rf-c5b118a7928693e8fd47

•Metodología mencionada

Scrum

•qué se espera del equipo

Se espera un equipo capaz de trabajar en ciclos de entrega de valor, con capacidad de autoorganización y un enfoque en la mejora continua de los procesos. El equipo debe ser colaborativo y adaptarse a un entorno de teletrabajo utilizando herramientas de gestión ágil.

•qué parte del proyecto encaja con esa metodología

Resultados visibles pronto: Scrum utiliza Sprints (normalmente de 2 semanas) al final de los cuales hay que presentar una demo funcional. Esto cumple con la exigencia del cliente de ver avances rápidos en la web de citas.

•qué parte **no encaja del todo**.

El cliente no puede reunirse todas las semanas: Este es el mayor problema. Scrum es muy estricto con la presencia del cliente (*Product Owner*) en reuniones clave como la *Sprint Review* y la *Sprint Planning*. Si el cliente no está disponible semanalmente, el equipo puede quedar bloqueado o avanzar en una dirección equivocada.

5.2 Selección razonada de metodología

a) Elige **una única metodología principal** entre:

•Cascada

•Kanban

•Scrum

•Extreme Programming

Kanban

b) Justifica por qué **descartas explícitamente las otras tres**, usando las condiciones del proyecto.

Se descarta el Modelo en Cascada: Este modelo requiere que todos los requisitos estén definidos y cerrados desde el primer día. En una aplicación de gestión de citas médicas donde los requisitos cambian con frecuencia, la Cascada sería un fracaso asegurado. Además, el cliente no vería resultados hasta el final de todas las fases, incumpliendo su condición de querer resultados visibles pronto.

Se descarta Scrum: Aunque Scrum permite resultados rápidos, depende totalmente de la disponibilidad del cliente para las ceremonias (reuniones) de *Sprint Review* y *Sprint Planning*. Si el cliente no puede reunirse todas las semanas, el flujo de Scrum se rompe, ya que no habría nadie para validar el incremento de software o priorizar el siguiente ciclo. Además, la carga de reuniones de Scrum puede abrumar a un equipo pequeño e inexperto.

Se descarta Extreme Programming (XP): XP pone un foco extremo en la calidad técnica, lo cual es positivo, pero sus prácticas (como el *Pair Programming* o el TDD) requieren una curva de aprendizaje muy alta. Un equipo con poca experiencia tardaría demasiado tiempo en dominar estas técnicas, retrasando la entrega del producto. Además, al igual que Scrum, requiere una presencia constante del cliente que aquí no existe.

Kanban es la metodología más flexible para este escenario. Permite gestionar los **requisitos cambiantes** simplemente moviendo tarjetas en el tablero. No obliga a realizar reuniones semanales estrictas (se adapta a la agenda del cliente) y es visualmente muy sencillo, lo que ayuda a que un **equipo sin experiencia** entienda el estado del proyecto de un solo vistazo sin necesidad de dominar roles complejos.

5.3 Adaptación realista

a) Indica **dos problemas reales** que aparecerían usando la metodología elegida.

Falta de validación técnica: Al ser un equipo novato y no tener reuniones de revisión obligatorias, podrían estar programando mal la lógica de las citas sin que nadie les corrija a tiempo.

Cuellos de botella: Si el cliente tarda semanas en aparecer para validar una tarea, las tarjetas se acumularán en la columna de "Pendiente de validar", bloqueando el tablero.

b) Propón **ajustes concretos** dentro de esa metodología para reducir esos problemas (sin mezclar metodologías).

Sistema de Validación Asíncrona: Para no detener el flujo cuando el cliente no está, el equipo grabará un vídeo corto (demo de 2 minutos) por cada funcionalidad terminada y lo adjuntará a la tarjeta de Kanban. El cliente podrá validar las tareas desde su móvil o en su tiempo libre sin necesidad de una reunión presencial, permitiendo que las tarjetas sigan moviéndose en el tablero.

Inclusión de una columna de "Revisión Cruzada" (Peer Review): Para asegurar la calidad sin depender de un experto externo, se añade un paso obligatorio en el tablero: ninguna tarea puede pasar a "Hecho" sin que otro compañero del equipo haya revisado el código y el funcionamiento. Esto fomenta que el equipo aprenda entre sí y se detecten errores de calidad de forma temprana.

c) Explica qué riesgos siguen existiendo, aunque se hagan esos ajustes.

Riesgo de Deuda Técnica: Aunque los compañeros se revisen el código entre sí, al ser todos inexpertos, existe el riesgo de que pasen por alto errores graves de seguridad o de base de datos que solo un perfil senior detectaría.

Riesgo de falta de alineación: La validación por vídeo (asíncrona) es útil, pero se pierde el matiz de la conversación. Puede que el cliente valide algo que cree entender, pero que al final del proyecto no encaje exactamente con lo que necesitaba, obligando a rehacer trabajo.

6. Decisión crítica

Responde razonadamente:

Si el proyecto se alarga dos años y el cliente cambia completamente el enfoque inicial,

¿en qué momento exacto se notaría más el problema:

•con una metodología tradicional

•o con una metodología ágil?

El problema se notaría mucho más **con una metodología tradicional (modelo en cascada)**. Mientras que en las metodologías ágiles los cambios se asimilan de forma progresiva, en el modelo tradicional el cambio de enfoque al cabo de dos años supone el colapso del proyecto.

Indica:

•Fase o momento concreto

Se notaría en la **fase de Pruebas finales o Entrega**. En el modelo tradicional, el cliente solo ve el producto terminado al final del ciclo de desarrollo. Si tras dos años de trabajo el enfoque ha cambiado, el equipo presentará una aplicación que cumple con unos requisitos de hace 24 meses que ya no tienen ninguna utilidad para el negocio actual.

•Impacto técnico

El impacto es devastador. Como se construyó todo basándose en un diseño cerrado hace dos años, gran parte del código y la estructura de la base de datos no sirven para el nuevo enfoque. Habría que desechar casi todo el trabajo realizado ("tirar código a la basura").

•Impacto organizativo.

Provoca una crisis de confianza total entre el cliente y el equipo. El cliente siente que ha pagado por algo que ya no le sirve, y el equipo se siente frustrado por haber trabajado dos años en un producto que será descartado. El coste de pivotar en este punto es, a menudo, tan alto como empezar el proyecto de cero.

7.Webgrafia

https://experts-deny-b9a.craft.me/y3eKOUPLrtEM6g

https://instituteprojectmanagement.com/es/blog/que-es-la-metodologia-en-cascada-todo-lo-que-necesitas-saber/

https://www.paradigmadigital.com/techbiz/11-males-mas-comunes-retrospectivas-escalados/