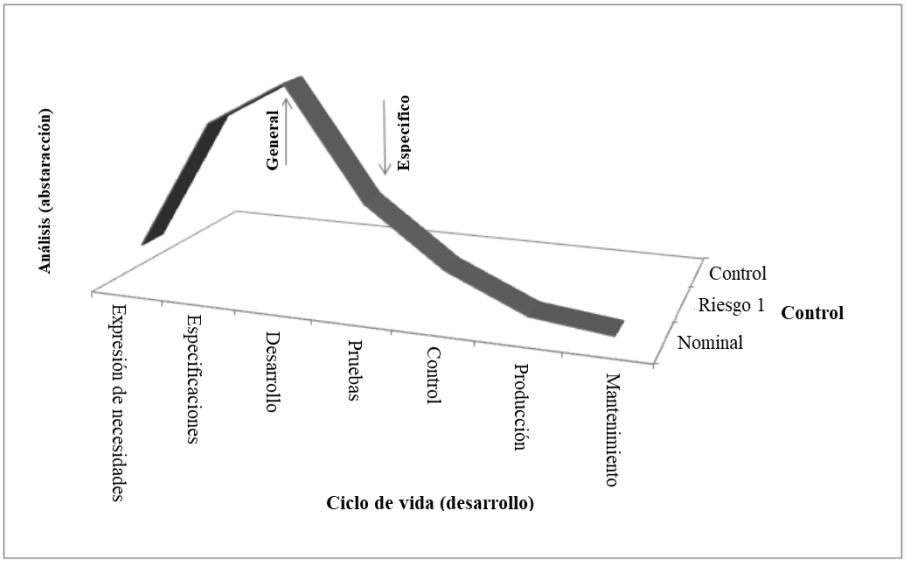
# Capítulo 3
# Evaluar el riesgo

## 1. Los tres ejes

Es normal que a veces el director de proyectos esté desconcertado debido a la cantidad de elementos que debe organizar para tener éxito. ¿En qué dirección ir? ¿Por dónde empezar y cómo continuar?

Para responder a estas preguntas, en primer lugar se debe imaginar el proceso del proyecto como se describe en el capítulo Iniciar un proyecto informático. En esta ocasión, elegimos un espacio estructurado por tres ejes, tiempo (ciclo de vida), análisis y control.

Cada dimensión de este espacio se corresponde con un método más o menos estándar, que dará lugar a un proceso que el director de proyectos tendrá que seguir.




El primer eje (abscisa horizontal) es el que describe las fases del proyecto, tal y como las podemos preconfigurar. El diagrama anterior propone un modelo de desarrollo bastante estándar, a partir de la expresión de necesidades, evolucionando hasta la entrega (producción) y el mantenimiento del proyecto. Estrictamente hablando, deberíamos introducir el final del ciclo de vida del proyecto, porque sabemos que el marco de soporte no es infinito. Como veremos más adelante, existen otros modelos de desarrollo que proponen diferentes organizaciones.

A continuación, nos encontramos con el eje de análisis (ordenadas, verticales). Este representa el nivel de abstracción del proyecto. Cuanto más alta es la ordenada, mayor es la abstracción. Por el contrario, una ordenada baja equivale a un nivel de detalle muy alto. En el ejemplo que se muestra en el diagrama, podemos ver claramente la abstracción creciendo al inicio del proyecto, antes de bajar gradualmente. Este tipo de curva es bastante característica de los métodos de análisis basados en UML (*Unified Modeling Language*) y Merise. Una vez más, existen otros esquemas que reflejan otras formas de proceder.

En el último eje, se representan los riesgos o peligros, los puntos de control y la toma de decisiones. Este es el eje de control (ciclo de decisión). En comparación con los otros dos ejes, es el que esconde más "sorpresas", pero tampoco se debe dejar completamente libre. A lo largo de los proyectos, se han forjado métodos para organizar el control y la gestión de manera metódica y eficiente.

Por lo tanto, el "secreto" durante el inicio de un proyecto, ¿se centraría en la elección de los métodos asociados a cada dimensión? Esto no es suficiente porque todavía tenemos que garantizar una mezcla armoniosa de estos tres componentes, una comunicación efectiva. En resumen, dar un sentido práctico sin el que ni siquiera las mejores recomendaciones del mundo tendrían ningún efecto. En otras palabras, la aplicación aislada y desconectada de un método en un solo eje no genera mejores resultados que un proyecto en el que nos hemos olvidado del resto de ejes.

La siguiente tabla ofrece una visión general del resultado cuando los métodos se aplican de manera aislada (independientemente uno del otro) o no están presentes:

| Método/eje | Aplicación única o independiente | No aplicación (ausencia) |
|---|---|---|
| Desarrollo | En general, el método de análisis está censurado o no da resultados concluyentes. | Incumplimiento o no cumplimiento (la expresión de las necesidades se tiene en cuenta demasiado tarde).<br>Deriva del proyecto (dificultad para estabilizar la solución debido a la falta de pruebas). |
| Análisis | Visión demasiado conceptual. El proyecto se vuelve técnicamente arriesgado porque se olvidan las pruebas de integración. | Incumplimiento o no cumplimiento (centrarse en funciones menores, mientras faltan funciones principales).<br>Código no óptimo que causa dificultades para evolucionarlo (sin factorización). |

| | | |
|---|---|---|
| Control | Gestión inadecuada en relación con las características del proyecto. | Incumplimiento o no observación de los objetivos, falta de control.<br>Explosión del equipo de proyecto en caso de escalabilidad.<br>Gestión de cambios inexistente: rechazo de la aplicación entregada. |

Esta tabla resume seis construcciones (*antipatterns*) que se deben evitar a toda costa. Y, sin embargo, hay ocasiones en las que se pueden cometer errores, ya sea porque los hábitos tienen una fecha de caducidad o porque un nuevo método no siempre se puede integrar fácilmente dentro del marco de la gestión de proyectos.

**Observación**

*Hay muchos ejemplos de obstáculos. El director de proyectos siempre debe tener en cuenta que los tres componentes son esenciales y que deben cooperar para ser efectivos.*

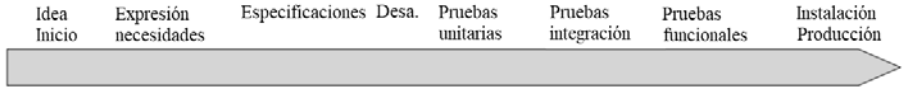
## 2. El modelo de desarrollo

El modelo de desarrollo es una de las primeras elecciones que debe tomar el director de proyectos. Da forma explícita a la organización temporal del proyecto y tiene una gran influencia en el resultado final.

No existe un modelo correcto o incorrecto, al menos entre las grandes clases enumeradas a continuación. Sin embargo, cada proyecto tiene características que hacen que la aplicación de un modelo sea adecuada o, por el contrario, ineficaz.

## 2.1 El modelo en cascada

También se llama modelo lineal o modelo nominal. Sin lugar a dudas se trata del modelo más sencillo, aunque no por ello deja de tener éxito.




En este modelo, cada etapa sigue a la anterior, sin necesidad de "esperar" su resultado. Tan pronto como se completa la expresión de las necesidades o requisitos, se escriben las especificaciones. Al final de este trabajo, se lleva a cabo el desarrollo hasta su finalización. Las pruebas unitarias se suceden entre sí, y así sucesivamente.

Antes de explicar por qué se llama en cascada, veamos las diferentes etapas previstas por este modelo.

| | |
|---|---|
| Idea - inicio | Este es el punto de partida del proyecto, el momento en que se cumplen las condiciones de hardware y materiales y el equipo comparte la visión del proyecto. |
| Expresión de las necesidades | Tras tomar la decisión de comenzar un proyecto, volvemos a recopilar información de los usuarios para obtener más detalles. Estas peticiones se registran en un documento (papel u hoja de cálculo), se identifican y, finalmente, se agrupan y se buscan sinergias entre ellas. |
| Especificaciones | Todas las peticiones se consideran en función de los diferentes enfoques dictados por el modelo de análisis: importancia, prioridad, dificultad técnica, riesgos, etc. Las especificaciones muestran un desglose técnico-funcional de la aplicación. |
| Desarrollo | Las especificaciones se traducen en código, proyectando la segmentación modular (componentes) en la plataforma elegida por el arquitecto. |

| | |
|---|---|
| Pruebas unitarias | Cada componente se prueba independientemente del resto. Su comportamiento debe cumplir con las especificaciones, dentro del rango de operativo y de funcionamiento previsto. |
| Pruebas de integración | Los componentes se ensamblan con mayor frecuencia en una plataforma técnicamente heterogénea. La lectura de registros puede causar problemas de seguridad, carga, potencia (gestión de valores NULL por ejemplo, etc.). Las pruebas de integración aseguran que el conjunto final es estable en todos los casos de uso previstos en las especificaciones. |
| Pruebas funcionales | Se examinan todos los procesos que abarca la aplicación. Estos deben respetar la metodología elegida para el usuario, responder de manera adecuada y con buen rendimiento y tiempos de respuesta aceptables. |
| Instalación | La aplicación se despliega en su plataforma final (antes de este paso, se puede realizar una fase de industrialización utilizando software de empaquetado). Algunas veces se tienen que realizar ajustes de seguridad, redefinir cadenas de conexión, etc. |

Este modelo es nominal, es decir, una base del trabajo consiste en que todos son libres de agregar otros pasos o volver a revisar los ya realizados. También es lineal porque ninguna etapa se aborda de manera paralela a otra.

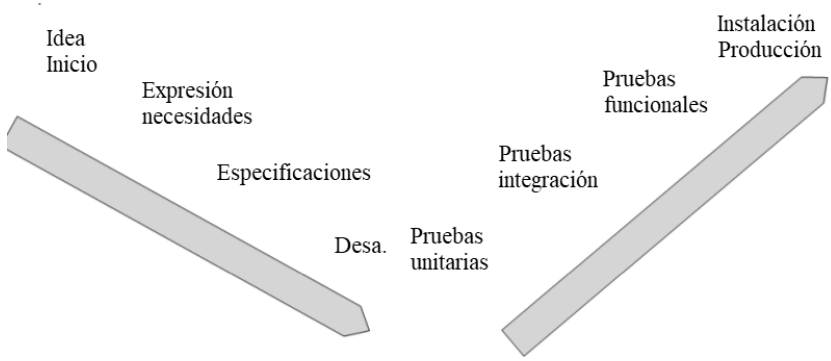
¿De dónde viene el nombre en cascada? Ir avanzando es difícil, como para los peces que intentan superar las corrientes de un río. Las esclusas, presas o cascadas son difíciles de cruzar. En el modelo en cascada, un coste unitario "en descenso" se multiplica por diez si es necesario volver atrás en algún punto. En otras palabras, un error de diseño se paga de manera exponencial. Si se olvidó especificar algo en su momento, su desarrollo costará 10 veces más de lo previsto. Si el descuido se arregla en las fases de pruebas unitarias, el coste es 100 veces más.

## 2.2 El modelo en forma de V

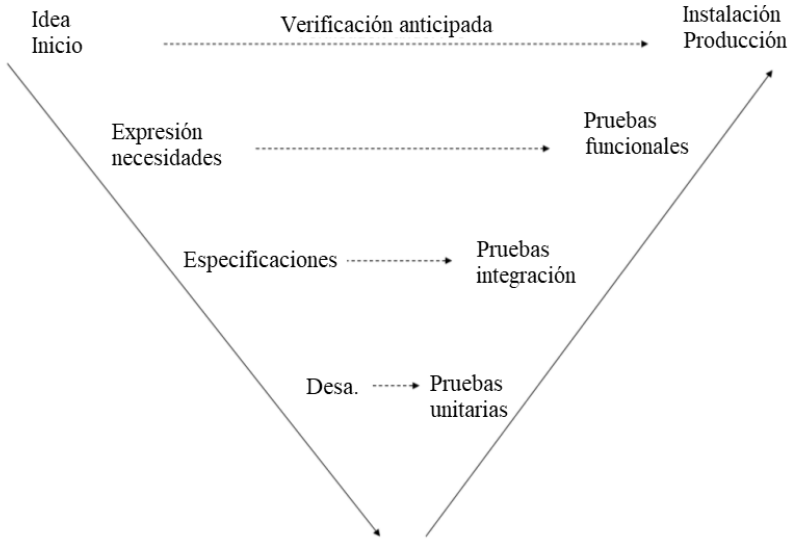
Si se observa más de cerca el modelo en cascada, se observa una simetría entre las diferentes etapas que encontramos a lo largo del proceso. Esta simetría es precisamente uno de los puntos débiles del modelo, ya que las fases de validación están en última instancia demasiado lejos de las fases de especificación. Por lo tanto, cada persona involucrada en la realización del proyecto queda atrapada en un túnel, obligada a esperar para comprobar la conformidad del producto.

Por lo tanto, el usuario espera que el producto final cumpla con sus expectativas. El diseñador no puede verificar sus hipótesis antes de realizar pruebas funcionales. El arquitecto no puede integrar componentes que aún no existen. El desarrollador no está en mejor situación ya que, dada la falta de retrospectiva y perspectiva general, hay una probabilidad alta de que sus desarrollos se tengan que retocar o se abandonen.

En lugar de posponer las pruebas funcionales tanto como sea posible, el ciclo en V las anticipa ejecutando el proceso de tal manera que estas pruebas se externalicen, se realicen en paralelo y finalmente se apliquen lo antes posible.



Es necesario externalizar las pruebas, de lo contrario no serían objetivas. Realizarlas de manera paralela tiene como objetivo verificar "sobre el papel" que la composición coincide con las necesidades expresadas. De esta manera, se puede anticipar un error importante en la cobertura funcional y ahorramos trabajo innecesario o erróneo a los desarrolladores o incluso a los analistas funcionales.




¿Cuáles son los impactos de este modelo en la duración del proyecto? El software no está disponible el doble de rápido, sino que la duración se prolonga entre un 30 y un 40%. De hecho, las pruebas se realizan dos veces. El primer lugar, de manera ficticia, con fines de control. Y la segunda, en un escenario funcional real basado en el funcionamiento operativo del software.

Por lo tanto, el modelo en forma de V es una mejora con respecto al modelo en cascada en términos de pruebas funcionales, a costa de aumentar la duración del proyecto.