**Modelo cascada**

|  |  |
| --- | --- |
| **Pros** | **Contras** |
| Es bueno para un software de menor porte o algunas veces de mediano porte. | Es malo para un software de gran porte. |
| Se puede crear un software con pocos requisitos. | No tiene retorno después de pasar a otra etapa. |
| Eficacia y simplicidad. | Si se plantea implementar algo nuevo se debe comenzar de cero otra vez. |
| La planificación es sencilla. | Se demora mucho en pasar por todo el ciclo. |

**Modelo iterativo incremental**

|  |  |
| --- | --- |
| **Pros** | **Contras** |
| Permite la entrega de versiones parciales a medida que se va construyendo el producto final. | Requiere de mucha planeación (Tanto administrativa como técnica). |
| Te permite trabajar en paralelo con otras etapas del proyecto. | Puede ser costoso por la pérdida de tiempo al volver atrás |
| Cada versión incorpora los anteriores incrementos de las funcionalidades y requisitos que fueron analizados como necesarios. | No es recomendable para casos de sistema de tiempo real, alto nivel de seguridad, de procesamiento distribuido o de alto índice de riesgo |
| Permite una entrega más rápida que la del modelo cascada. |  |
| Resulta muy útil con baja dotación de personal |  |

Modelo espiral

|  |  |
| --- | --- |
| **Pros** | **Contras** |
| Proporciona potencial para el desarrollo rápido de versiones incrementales. | Se tiene que analizar mucho el nivel de riesgo. |
| Como el software evoluciona a medida que progresa el sistema, el desarrollador y el cliente comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos. | Debido a su complejidad no se aconseja utilizarlo en sistemas de pequeño porte. |
| Evita muchos problemas al demandar una reducción de riesgos. | Modelo costoso. |
| En la producción de grandes sistemas ha doblado la productividad. | Requiere experiencia en la identificación de riesgos. |