Estilos arquitectónicos  
  
**• Estilos influenciados por lenguajes de programación**

**→ Ventajas  
⇒** Representar la semántica del problema a resolver suele ser sencilla pues los niveles de abstracción de un paradigma de programación por lo general son lo suficientemente elevados como para permitir que abstraer el mundo real sea directo y sin demasiados rodeos. **→ Desventajas  
⇒** Requiere un avanzado entendimiento no sólo del estilo arquitectónico sino del paradigma de programación en cuestión, lo cual puede conducir a invertir más tiempo del esperado entender la forma de implementación en caso de no estar familiarizado previamente con el paradigma elegido

**• Estilos de niveles o capas**

→ **Ventajas**  
⇒ Al estar segmentado el programa en diversos niveles la modularización del proyecto será alta, lo cual permitirá al programador examinar el programa parte por parte para encontrar y depurar errores, sin tomar en cuenta que además una vez que las capas inferiores sean probadas, las capas superiores pueden ser implementadas sin importar lo que pase debajo  
→ **Desventajas**⇒ Pese a que la modularización suele ser una característica ampliamente valorada y recomendada puede llegar a conducir a cadenas de errores enormes si una de las capas inferiores modifica o necesita modificar su API, ocasionando que las capas superiores necesiten ajustarse potencialmente en cadena

• **Estilos cliente – servidor**

**→ Ventajas**⇒ Los datos pueden mantenerse centralizados en una fuente común que puede servir a distintos y numerosos clientes, lo cual favorece sistemas distribuidos que precisen múltiples y concurrentes acceso a una misma base de información  
**→ Desventajas**⇒ Si el servidor llega a fallar todos los clientes se verán afectados hasta que los problemas puedan ser solucionados, así que es preciso ser cauteloso y establecer medidas de precaución en caso de pérdida de comunicación o información. Además, es un estilo muy exigente en cuanto a recursos que atañen protocolos de red o comunicación

• **Estilos de flujo de datos**

→ **Ventajas**  
⇒ Principalmente se trata de un estilo muy simple de implementar, entender y ejecutar, además de que se tiene un control muy amplio sobre la ejecución misma debido a que las conexiones se hacen precisamente a mano  
→ **Desventajas**⇒ Se trata de un estilo muy antiguo y seguramente obsoleto donde no hay relación directa entre los componentes hasta que es explícitamente indicado lo que lo hace poco independiente y muy susceptible a errores humanos.

• **Estilos de memoria compartido**

→ **Ventajas**  
⇒ La estrategia necesaria para solucionar el problema provendrá de las ideas conjuntas que los agentes expresen en la pizarra común, lo cual desenvocará ideas y soluciones potencialmente más creativas, inteligentes y eficaces o inclusive podría generar soluciones para problemas complejos que, posiblemente, no podrían ser alcanzadas con ninguna otra arquitectura.   
→ **Desventajas**⇒ Pese a que la generalidad de la arquitectura pueda funcionar muy bien, las funciones particulares para cada agente no están predefinidas, motivo por lo cual calcular el costo en recursos computacionales puede llegar a ser incalculable a priori. Además de que depende completamente de la buena comunicación y aportación de los agentes para alcanzar la solución lo cual significa que no hay garantía de que una solución sea alcanzada.

**• Estilos de intérprete**

→ **Ventajas**  
⇒ Ofrece al usuario un gran control y retroalimentación de sus propios programas, incentivando a la creación de programas sobre los ya existentes, de esa manera la arquitectura misma se va especializando sin ver afectadas sus bases  
→ **Desventajas**⇒ Debido al mismo estilo arquitectónico de ejecutar línea por línea, los programas pueden ver afectada su velocidad y rendimiento, además de que el consumo de memoria es mayor comparado a si se tradujeran todas las líneas de código a la vez

**• Estilos de invocación implícita**

→ **Ventajas**  
⇒ Le otorga al usuario completa libertad de interactuar con el sistema a sabiendas de que sus acciones, decisiones e interacciones en general harán al sistema efectuar una acción a la que el usuario mismo puede guiar.  
→ **Desventajas**⇒ La ventaja principal de este estilo es también su mayor desventaja: al otorgar al usuario tanta libertad se vuelve mucho más difícil para el programador controlar que todo lo que suceda o pueda suceder en el programa funcione de la manera esperada. La, en ocasiones, inmensa cantidad de permutaciones de eventos puede dificultar en sobremedida el control total de excepciones que podrían llegar a ser ocasionadas.

**• Estilos distribuidos**

→ **Ventajas**  
⇒ Permite una amplia intercomunicación entre usuarios/sistemas, extremadamente requerida en muchos casos. Debido a que los objetos se ejecutan en distintos “hosts” el sistema general es considerablemente más firme y menos susceptible a errores graves al mantener todas las demás máquinas en funcionamiento si una llegase a fallar.  
→ **Desventajas**⇒ Si alguno de los servidores llega a ser infectado es muy probable que el software maligno sea propagado si no se cuentan con las medidas de seguridad necesarias. Además, dado que la conexión muy probablemente se realice mediante protocolos de red es posible que la latencia de transmisión entre ellas pueda verse afectada por motivos varios que podrían debilitar o perturbar la red de sistemas distribuidos.

*Carlos Andrés Reyes Evangelista*

*157068*