Paradigmas Fundamentales de Programación

Limitaciones y extensiones de la programación declarativa

Juan Francisco Díaz Frias

Maestría en Ingeniería, Énfasis en Ingeniería de Sistemas y Computación
Escuela de Ingeniería de Sistemas y Computación,
home page: http://eisc.univalle.edu.co
Universidad del Valle - Cali, Colombia

Modelos y Paradigmas de Programación

Universidad del Valle

Plan

Modelos y Paradigmas de Programación

Universidad del Valle

Plan

Limitaciones y extensiones de la programación declarativa (1)

Lo que hemos ganado

- Construcción simple de sistemas: los componentes declarativos se pueden construir y depurar independientemente de los otros componentes.
- La complejidad de un sistema es la suma de las complejidades de sus componentes.
- ¿Todo puede ser programado en una forma declarativa, de manera que los programas sean a la vez naturales y eficientes?

Limitaciones y extensiones de la programación declarativa (1)

Lo que hemos ganado

- Construcción simple de sistemas: los componentes declarativos se pueden construir y depurar independientemente de los otros componentes.
- La complejidad de un sistema es la suma de las complejidades de sus componentes.
- ¿Todo puede ser programado en una forma declarativa, de manera que los programas sean a la vez naturales y eficientes?

Posibles problemas

- Eficiencia: un programa es eficiente si su desempeño difiere sólo en un factor constante del desempeño del programa en lenguaje ensamblador que resuelve el mismo problema.
- Naturalidad: un programa es natural si requiere muy poco código para resolver problemas técnicos no relacionados con el problema original. Tres asuntos asociados: modularidad, no-determinismo, e interfaces con el mundo real.

Limitaciones y extensiones de la programación declarativa (1)

Lo que hemos ganado

- Construcción simple de sistemas: los componentes declarativos se pueden construir y depurar independientemente de los otros componentes.
- La complejidad de un sistema es la suma de las complejidades de sus componentes.
- ¿Todo puede ser programado en una forma declarativa, de manera que los programas sean a la vez naturales y eficientes?

Posibles problemas

- Eficiencia: un programa es eficiente si su desempeño difiere sólo en un factor constante del desempeño del programa en lenguaje ensamblador que resuelve el mismo problema.
- Naturalidad: un programa es natural si requiere muy poco código para resolver problemas técnicos no relacionados con el problema original. Tres asuntos asociados: modularidad, no-determinismo, e interfaces con el mundo real.

Limitaciones y extensiones de la programación declarativa (1)

Lo que hemos ganado

- Construcción simple de sistemas: los componentes declarativos se pueden construir y depurar independientemente de los otros componentes.
- La complejidad de un sistema es la suma de las complejidades de sus componentes.
- ¿Todo puede ser programado en una forma declarativa, de manera que los programas sean a la vez naturales y eficientes?

Posibles problemas

- Eficiencia: un programa es eficiente si su desempeño difiere sólo en un factor constante del desempeño del programa en lenguaje ensamblador que resuelve el mismo problema.
- Naturalidad: un programa es natural si requiere muy poco código para resolver problemas técnicos no relacionados con el problema original. Tres asuntos asociados: modularidad, no-determinismo, e interfaces con el mundo real.

Limitaciones y extensiones de la programación declarativa (1)

Lo que hemos ganado

- Construcción simple de sistemas: los componentes declarativos se pueden construir y depurar independientemente de los otros componentes.
- La complejidad de un sistema es la suma de las complejidades de sus componentes.
- ¿Todo puede ser programado en una forma declarativa, de manera que los programas sean a la vez naturales y eficientes?

Posibles problemas

- Eficiencia: un programa es eficiente si su desempeño difiere sólo en un factor constante del desempeño del programa en lenguaje ensamblador que resuelve el mismo problema.
- Naturalidad: un programa es natural si requiere muy poco código para resolver problemas técnicos no relacionados con el problema original. Tres asuntos asociados: modularidad, no-determinismo, e interfaces con el mundo real.

Modelos y Paradigmas de Programación

Universidad del Valle

Plan

Limitaciones y extensiones de la programación declarativa (2)

Eficiencia

- El computador está optimizado para modificar datos en el sitio, mientras que el modelo declarativo nunca modifica los datos sino que siempre crea datos nuevos.
- No es un problema tan serio: el tamaño de su memoria activa permanece pequeño.
- Posible solución: implementar el modelo declarativo con asignación en el sitio.
- ¿Puede un compilador traducir efectivamente programas declarativos en un computador estándar?: si, si a uno se le permite reescribir el programa a uno menos natural.

Naturalidad: modularidad
Un programa es modular con respecto a un cambio en una parte específica si el cambio se puede realizar sin cambiar el resto del programa. Serían modulares las soluciones declarativas de:

Limitaciones y extensiones de la programación declarativa (2)

Eficiencia

- El computador está optimizado para modificar datos en el sitio, mientras que el modelo declarativo nunca modifica los datos sino que siempre crea datos nuevos.
- No es un problema tan serio: el tamaño de su memoria activa permanece pequeño.
- Posible solución: implementar el modelo declarativo con asignación en el sitio.
- ¿Puede un compilador traducir efectivamente programas declarativos en un computador estándar?: si, si a uno se le permite reescribir el programa a uno menos natural.

Naturalidad: modularidad

Un programa es modular con respecto a un cambio en una parte específica si el cambio se puede realizar sin cambiar el resto del programa. Serían modulares las soluciones declarativas de:

Limitaciones y extensiones de la programación declarativa (2)

Eficiencia

- El computador está optimizado para modificar datos en el sitio, mientras que el modelo declarativo nunca modifica los datos sino que siempre crea datos nuevos.
- No es un problema tan serio: el tamaño de su memoria activa permanece pequeño.
- Posible solución: implementar el modelo declarativo con asignación en el sitio.
- ¿Puede un compilador traducir efectivamente programas declarativos en un computador estándar?: si, si a uno se le permite reescribir el programa a uno menos natural.

Naturalidad: modularidad

Un programa es modular con respecto a un cambio en una parte específica si el cambio se puede realizar sin cambiar el resto del programa. Serían modulares las soluciones declarativas de:

Limitaciones y extensiones de la programación declarativa (2)

Eficiencia

- El computador está optimizado para modificar datos en el sitio, mientras que el modelo declarativo nunca modifica los datos sino que siempre crea datos nuevos.
- No es un problema tan serio: el tamaño de su memoria activa permanece pequeño.
- Posible solución: implementar el modelo declarativo con asignación en el sitio.
- ¿Puede un compilador traducir efectivamente programas declarativos en un computador estándar?: si, si a uno se le permite reescribir el programa a uno menos natural.

Naturalidad: modularidad

Un programa es modular con respecto a un cambio en una parte específica si el cambio se puede realizar sin cambiar el resto del programa. Serían modulares las soluciones declarativas de:

- Memorización en memoria oculta.
- Instrumentación de un programa: contar el cuántas veces se invoca alguno de sus subcomponentes.

Limitaciones y extensiones de la programación declarativa (2)

Eficiencia

- El computador está optimizado para modificar datos en el sitio, mientras que el modelo declarativo nunca modifica los datos sino que siempre crea datos nuevos.
- No es un problema tan serio: el tamaño de su memoria activa permanece pequeño.
- Posible solución: implementar el modelo declarativo con asignación en el sitio.
- ¿Puede un compilador traducir efectivamente programas declarativos en un computador estándar?: si, si a uno se le permite reescribir el programa a uno menos natural.

Naturalidad: modularidad

Un programa es modular con respecto a un cambio en una parte específica si el cambio se puede realizar sin cambiar el resto del programa. Serían modulares las soluciones declarativas de:

- Memorización en memoria oculta.
- Instrumentación de un programa: conocer cuántas veces se invoca alguno de sus subcomponentes.

Limitaciones y extensiones de la programación declarativa (2)

Eficiencia

- El computador está optimizado para modificar datos en el sitio, mientras que el modelo declarativo nunca modifica los datos sino que siempre crea datos nuevos.
- No es un problema tan serio: el tamaño de su memoria activa permanece pequeño.
- Posible solución: implementar el modelo declarativo con asignación en el sitio.
- ¿Puede un compilador traducir efectivamente programas declarativos en un computador estándar?: si, si a uno se le permite reescribir el programa a uno menos natural.

Naturalidad: modularidad

Un programa es modular con respecto a un cambio en una parte específica si el cambio se puede realizar sin cambiar el resto del programa. Serían modulares las soluciones declarativas de:

- Memorización en memoria oculta.
- Instrumentación de un programa: conocer cuántas veces se invoca alguno de sus subcomponentes.

Limitaciones y extensiones de la programación declarativa (2)

Eficiencia

- El computador está optimizado para modificar datos en el sitio, mientras que el modelo declarativo nunca modifica los datos sino que siempre crea datos nuevos.
- No es un problema tan serio: el tamaño de su memoria activa permanece pequeño.
- Posible solución: implementar el modelo declarativo con asignación en el sitio.
- ¿Puede un compilador traducir efectivamente programas declarativos en un computador estándar?: si, si a uno se le permite reescribir el programa a uno menos natural.

Naturalidad: modularidad

Un programa es modular con respecto a un cambio en una parte específica si el cambio se puede realizar sin cambiar el resto del programa. Serían modulares las soluciones declarativas de:

- Memorización en memoria oculta.
- Instrumentación de un programa: conocer cuántas veces se invoca alguno de sus subcomponentes.

Limitaciones y extensiones de la programación declarativa (3)

Componente declarativo

```
fun {SC ...}
 proc {P1 ...} ... end
 proc {P2 ...}
    ... {P1 ...} {P2 ...}
 end
 proc {P3 ...}
    ... {P2 ...} {P3 ...}
 end
in
 'export'(p1:P1 p2:P2 p3:P3)
end
```

- Requerimiento: contar el número de veces que se invoca al procedimiento `P1`

Solución decl.: no natural y no modular

```
fun {SC ...}
 proc {P1 ... S1 ?Sn}
    Sn=S1+1 ...
 end
 proc {P2 ... T1 ?Tn}
    ... {P1 ... T1 T2}
    {P2 ... T2 Tn}
 end
 proc {P3 ... U1 ?Un}
    ... {P2 ... U1 U2}
    {P3 ... U2 Un}
 end
in
 'export'(p1:P1 p2:P2 p3:P3)
 end
```

Limitaciones y extensiones de la programación declarativa (3)

Componente declarativo

```
fun {SC ...}
 proc {P1 ...} ... end
 proc {P2 ...}
    ... {P1 ...} {P2 ...}
 end
 proc {P3 ...}
    ... {P2 ...} {P3 ...}
 end
in
 'export'(p1:P1 p2:P2 p3:P3)
end
```

- Requerimiento: contar el número de veces que se invoca al procedimiento `P1`

Solución decl.: no natural y no modular

```
fun {SC ...}
 proc {P1 ... S1 ?Sn}
    Sn=S1+1 ...
 end
 proc {P2 ... T1 ?Tn}
    ... {P1 ... T1 T2}
    {P2 ... T2 Tn}
 end
 proc {P3 ... U1 ?Un}
    ... {P2 ... U1 U2}
    {P3 ... U2 Un}
 end
in
 'export'(p1:P1 p2:P2 p3:P3)
 end
```

Limitaciones y extensiones de la programación declarativa (3)

Componente declarativo

```
fun {SC ...}
 proc {P1 ...} ... end
 proc {P2 ...}
    ... {P1 ...} {P2 ...}
 end
 proc {P3 ...}
    ... {P2 ...} {P3 ...}
 end
in
 'export'(p1:P1 p2:P2 p3:P3)
end
```

- Requerimiento: contar el número de veces que se invoca al procedimiento `P1`

Solución decl.: no natural y no modular

```
fun {SC ...}
 proc {P1 ... S1 ?Sn}
    Sn=S1+1 ...
 end
 proc {P2 ... T1 ?Tn}
    ... {P1 ... T1 T2}
    {P2 ... T2 Tn}
 end
 proc {P3 ... U1 ?Un}
    ... {P2 ... U1 U2}
    {P3 ... U2 Un}
 end
in
 'export'(p1:P1 p2:P2 p3:P3)
 end
```

Limitaciones y extensiones de la programación declarativa (4)

Naturalidad: No-determinismo

- El modelo concurrente declarativo siempre se comporta determinísticamente.
- Pero, los componentes que son verdaderamente independientes se comportan de manera no determinística con respecto a los otros.
- Ejemplos reales: una aplicación cliente/servidor y una aplicación de despliegue de video.

Limitaciones y extensiones de la programación declarativa (4)

Naturalidad: No-determinismo

- El modelo concurrente declarativo siempre se comporta determinísticamente.
- Pero, los componentes que son verdaderamente independientes se comportan de manera no determinística con respecto a los otros.
- Ejemplos reales: una aplicación cliente/servidor y una aplicación de despliegue de video.

Aplicación cliente/servidor

- El servidor tiene un flujo de entrada a partir del cual lee los mensajes (órdenes de servicio).
- Suponga que hay dos clientes independientes. ¿Qué pasa si ellos se comunican con el mismo servidor?

Limitaciones y extensiones de la programación declarativa (4)

Naturalidad: No-determinismo

- El modelo concurrente declarativo siempre se comporta determinísticamente.
- Pero, los componentes que son verdaderamente independientes se comportan de manera no determinística con respecto a los otros.
- Ejemplos reales: una aplicación cliente/servidor y una aplicación de despliegue de video.

Aplicación cliente/servidor

- El servidor tiene un flujo de entrada a partir del cual lee los mensajes (órdenes de servicio).
- Suponga que hay dos clientes independientes. ¿Qué pasa si ellos se comunican con el mismo servidor?
- El servidor puede recibir información de los dos clientes en cualquier orden y tiene que determinarlo. El comportamiento requerido es no-determinístico observable.

Limitaciones y extensiones de la programación declarativa (4)

Naturalidad: No-determinismo

- El modelo concurrente declarativo siempre se comporta determinísticamente.
- Pero, los componentes que son verdaderamente independientes se comportan de manera no determinística con respecto a los otros.
- Ejemplos reales: una aplicación cliente/servidor y una aplicación de despliegue de video.

Aplicación cliente/servidor

- El servidor tiene un flujo de entrada a partir del cual lee los mensajes (órdenes de servicio).
- Suponga que hay dos clientes independientes. ¿Qué pasa si ellos se comunican con el mismo servidor?
- El servidor puede recibir información de los dos clientes en cualquier orden y tiene que determinarlo. El comportamiento requerido es no-determinístico observable.

Limitaciones y extensiones de la programación declarativa (4)

Naturalidad: No-determinismo

- El modelo concurrente declarativo siempre se comporta determinísticamente.
- Pero, los componentes que son verdaderamente independientes se comportan de manera no determinística con respecto a los otros.
- Ejemplos reales: una aplicación cliente/servidor y una aplicación de despliegue de video.

Aplicación cliente/servidor

- El servidor tiene un flujo de entrada a partir del cual lee los mensajes (órdenes de servicio).
- Suponga que hay dos clientes independientes. ¿Qué pasa si ellos se comunican con el mismo servidor?
- El servidor puede recibir información de los dos clientes en cualquier orden y tiene que determinarlo. El comportamiento requerido es no-determinístico observable.

Limitaciones y extensiones de la programación declarativa (4)

Naturalidad: No-determinismo

- El modelo concurrente declarativo siempre se comporta determinísticamente.
- Pero, los componentes que son verdaderamente independientes se comportan de manera no determinística con respecto a los otros.
- Ejemplos reales: una aplicación cliente/servidor y una aplicación de despliegue de video.

Aplicación cliente/servidor

- El servidor tiene un flujo de entrada a partir del cual lee los mensajes (órdenes de servicio).
- Suponga que hay dos clientes independientes. ¿Qué pasa si ellos se comunican con el mismo servidor?
- El servidor puede recibir información de los dos clientes en cualquier orden y tiene que determinarlo. El comportamiento requerido es no-determinístico observable.

Limitaciones y extensiones de la programación declarativa (4)

Aplicación: proyección de video

- Pantalla que recibe un flujo de imágenes y las proyecta.
- Las imágenes llegan a una velocidad particular (imgs/seg).
- Esta velocidad puede fluctuar: las imágenes tienen diferentes resoluciones, se puede realizar algún tipo de procesamiento sobre ellas, o el ancho de banda y la latencia de la red de transmisión es variable.

- Debido a la velocidad variable de llegada, la pantalla no puede proyectar siempre todas las imágenes.
- Algunas veces tiene que saltarse unas imágenes, por ejemplo, hasta la última imagen enviada.
- Este tipo de administración de flujos no se puede realizar en el modelo concurrente declarativo, pues no hay manera de detectar el fin de un flujo.

Limitaciones y extensiones de la programación declarativa (4)

Aplicación: proyección de video

- Pantalla que recibe un flujo de imágenes y las proyecta.
- Las imágenes llegan a una velocidad particular (imgs/seg).
- Esta velocidad puede fluctuar: las imágenes tienen diferentes resoluciones, se puede realizar algún tipo de procesamiento sobre ellas, o el ancho de banda y la latencia de la red de transmisión es variable.
- Debido a la velocidad variable de llegada, la pantalla no puede proyectar siempre todas las imágenes.
- Algunas veces tiene que saltarse unas imágenes, por ejemplo, hasta la última imagen enviada.
- Este tipo de administración de flujos no se puede realizar en el modelo concurrente declarativo, pues no hay manera de detectar el fin de un flujo.

Limitaciones y extensiones de la programación declarativa (5)

Naturalidad: el mundo real

- El mundo real no es declarativo, pues tiene tanto estado (las entidades tienen una memoria interna), como concurrencia (las entidades evolucionan independientemente).
- Pero, los programas declarativos interactúan con el mundo real: Problemas de interfaces y de especificación.
- Problemas de interfaces:A los componentes declarativos les hace falta expresividad para comunicarse con los componentes no declarativos.
- Problemas de especificación: las especificaciones frecuentemente mencionan estado y concurrencia.

¿Cuál es el modelo correcto?

Limitaciones y extensiones de la programación declarativa (5)

Naturalidad: el mundo real

- El mundo real no es declarativo, pues tiene tanto estado (las entidades tienen una memoria interna), como concurrencia (las entidades evolucionan independientemente).
- Pero, los programas declarativos interactúan con el mundo real: Problemas de interfaces y de especificación.
- Problemas de interfaces:A los componentes declarativos les hace falta expresividad para comunicarse con los componentes no declarativos.
- Problemas de especificación: las especificaciones frecuentemente mencionan estado y concurrencia.

¿Cuál es el modelo correcto?

- Los modelos de computación difieren en cuán expresivos son y en cuán difícil es razonar sobre los programas escritos en ellos.
- Los modelos más expresivos no son "mejores" que los otros, pues no siempre llevan a programas más simples y razonar sobre ellos es normalmente más difícil.

Limitaciones y extensiones de la programación declarativa (5)

Naturalidad: el mundo real

- El mundo real no es declarativo, pues tiene tanto estado (las entidades tienen una memoria interna), como concurrencia (las entidades evolucionan independientemente).
- Pero, los programas declarativos interactúan con el mundo real: Problemas de interfaces y de especificación.
- Problemas de interfaces:A los componentes declarativos les hace falta expresividad para comunicarse con los componentes no declarativos.
- Problemas de especificación: las especificaciones frecuentemente mencionan estado y concurrencia.

¿Cuál es el modelo correcto?

- Los modelos de computación difieren en cuán expresivos son y en cuán difícil es razonar sobre los programas escritos en ellos.
- Los modelos más expresivos no son "mejores" que los otros, pues no siempre llevan a programas más sencillos, y razonar sobre ellos es normalmente más difícil.
- Todos los modelos tienen su lugar y se pueden usar juntos, de manera beneficiosa, en el mismo programa.

Limitaciones y extensiones de la programación declarativa (5)

Naturalidad: el mundo real

- El mundo real no es declarativo, pues tiene tanto estado (las entidades tienen una memoria interna), como concurrencia (las entidades evolucionan independientemente).
- Pero, los programas declarativos interactúan con el mundo real: Problemas de interfaces y de especificación.
- Problemas de interfaces:A los componentes declarativos les hace falta expresividad para comunicarse con los componentes no declarativos.
- Problemas de especificación: las especificaciones frecuentemente mencionan estado y concurrencia.

¿Cuál es el modelo correcto?

- Los modelos de computación difieren en cuán expresivos son y en cuán difícil es razonar sobre los programas escritos en ellos.
- Los modelos más expresivos no son "mejores" que los otros, pues no siempre llevan a programas má sencillos, y razonar sobre ellos es normalmente más difícil.
- Todos los modelos tienen su lugar y se pueden usar juntos, de manera beneficiosa, en el mismo programa.
- Principio de mínima expresividad: Cuando se programe un componente, el modelo de computación correcto es el menos expresivo que resulte en un programa natural.

Limitaciones y extensiones de la programación declarativa (5)

Naturalidad: el mundo real

- El mundo real no es declarativo, pues tiene tanto estado (las entidades tienen una memoria interna), como concurrencia (las entidades evolucionan independientemente).
- Pero, los programas declarativos interactúan con el mundo real: Problemas de interfaces y de especificación.
- Problemas de interfaces:A los componentes declarativos les hace falta expresividad para comunicarse con los componentes no declarativos.
- Problemas de especificación: las especificaciones frecuentemente mencionan estado y concurrencia.

¿Cuál es el modelo correcto?

- Los modelos de computación difieren en cuán expresivos son y en cuán difícil es razonar sobre los programas escritos en ellos.
- Los modelos más expresivos no son “mejores” que los otros, pues no siempre llevan a programas má sencillos, y razonar sobre ellos es normalmente más difícil.
- Todos los modelos tienen su lugar y se pueden usar juntos, de manera beneficiosa, en el mismo programa.
- Principio de mínima expresividad: Cuando se programe un componente, el modelo de computación correcto es el menos expresivo que resulte en un programa natural.

Limitaciones y extensiones de la programación declarativa (5)

Naturalidad: el mundo real

- El mundo real no es declarativo, pues tiene tanto estado (las entidades tienen una memoria interna), como concurrencia (las entidades evolucionan independientemente).
- Pero, los programas declarativos interactúan con el mundo real: Problemas de interfaces y de especificación.
- Problemas de interfaces:A los componentes declarativos les hace falta expresividad para comunicarse con los componentes no declarativos.
- Problemas de especificación: las especificaciones frecuentemente mencionan estado y concurrencia.

¿Cuál es el modelo correcto?

- Los modelos de computación difieren en cuán expresivos son y en cuán difícil es razonar sobre los programas escritos en ellos.
- Los modelos más expresivos no son “mejores” que los otros, pues no siempre llevan a programas má sencillos, y razonar sobre ellos es normalmente más difícil.
- Todos los modelos tienen su lugar y se pueden usar juntos, de manera beneficiosa, en el mismo programa.
- Principio de mínima expresividad: Cuando se programe un componente, el modelo de computación correcto es el menos expresivo que resulte en un programa natural.

Limitaciones y extensiones de la programación declarativa (5)

Naturalidad: el mundo real

- El mundo real no es declarativo, pues tiene tanto estado (las entidades tienen una memoria interna), como concurrencia (las entidades evolucionan independientemente).
- Pero, los programas declarativos interactúan con el mundo real: Problemas de interfaces y de especificación.
- Problemas de interfaces:A los componentes declarativos les hace falta expresividad para comunicarse con los componentes no declarativos.
- Problemas de especificación: las especificaciones frecuentemente mencionan estado y concurrencia.

¿Cuál es el modelo correcto?

- Los modelos de computación difieren en cuán expresivos son y en cuán difícil es razonar sobre los programas escritos en ellos.
- Los modelos más expresivos no son “mejores” que los otros, pues no siempre llevan a programas má sencillos, y razonar sobre ellos es normalmente más difícil.
- Todos los modelos tienen su lugar y se pueden usar juntos, de manera beneficiosa, en el mismo programa.
- Principio de mínima expresividad: Cuando se programe un componente, el modelo de computación correcto es el menos expresivo que resulte en un programa natural.

Modelos y Paradigmas de Programación

Universidad del Valle

Plan

Modelos extendidos

Modelo secuencial declarativo

Abarca la programación funcional estricta y la programación lógica determinística. El comportamiento de un componente es independiente del momento en que se ejecuta o de lo que pasa en el resto de la computación.

Modelo concurrente declarativo

Modelo declarativo extendido con hilos explícitos y computación by-need. Concurrencia dirigida por los datos, y concurrencia dirigida por la demanda. Los componentes interactúan a través del uso y la ligadura de variables de flujo de datos compartidas.

Modelo concurrente por paso de mensajes

Modelo declarativo extendido con comunicación por canales (puertos). Permite restringir el no-determinismo observable a pequeñas partes del programa.

Modelo con estado

Modelo declarativo extendido con estado explícito. Permite expresar la programación secuencial orientada a objetos. Permite al componente llevar una "historia," la cual lo deja interactuar con su ambiente, adaptándose y aprendiendo de su pasado

Modelo concurrente con estado compartido

Modelo declarativo extendido con estado explícito y con hilos. Contiene la programación concurrente orientada a objetos. Razonar con este modelo es muy complejo pues pueden existir múltiples historias interactuando en formas impredecibles.

Modelo relacional

Modelo declarativo extendido con búsqueda. Permite la programación con relaciones, abarcando la programación lógica no-determinística. Precursor de la programación con restricciones.

Modelos extendidos

Modelo secuencial declarativo

Abarca la programación funcional estricta y la programación lógica determinística. El comportamiento de un componente es independiente del momento en que se ejecuta o de lo que pasa en el resto de la computación.

Modelo concurrente declarativo

Modelo declarativo extendido con hilos explícitos y computación by-need. Concurrencia dirigida por los datos, y concurrencia dirigida por la demanda. Los componentes interactúan a través del uso y la ligadura de variables de flujo de datos compartidas.

Modelo concurrente por paso de mensajes

Modelo declarativo extendido con comunicación por canales (puertos). Permite restringir el no-determinismo observable a pequeñas partes del programa.

Modelo con estado

Modelo declarativo extendido con estado explícito. Permite expresar la programación secuencial orientada a objetos. Permite al componente llevar una "historia," la cual lo deja interactuar con su ambiente, adaptándose y aprendiendo de su pasado

Modelo concurrente con estado compartido

Modelo declarativo extendido con estado explícito y con hilos. Contiene la programación concurrente orientada a objetos. Razonar con este modelo es muy complejo pues pueden existir múltiples historias interactuando en formas impredecibles.

Modelo relacional

Modelo declarativo extendido con búsqueda. Permite la programación con relaciones, abarcando la programación lógica no-determinística. Precursor de la programación con restricciones.

Modelos extendidos

Modelo secuencial declarativo

Abarca la programación funcional estricta y la programación lógica determinística. El comportamiento de un componente es independiente del momento en que se ejecuta o de lo que pasa en el resto de la computación.

Modelo concurrente declarativo

Modelo declarativo extendido con hilos explícitos y computación by-need. Concurrencia dirigida por los datos, y concurrencia dirigida por la demanda. Los componentes interactúan a través del uso y la ligadura de variables de flujo de datos compartidas.

Modelo concurrente por paso de mensajes

Modelo declarativo extendido con comunicación por canales (puertos). Permite restringir el no-determinismo observable a pequeñas partes del programa.

Modelo con estado

Modelo declarativo extendido con estado explícito. Permite expresar la programación secuencial orientada a objetos. Permite al componente llevar una "historia," la cual lo deja interactuar con su ambiente, adaptándose y aprendiendo de su pasado

Modelo concurrente con estado compartido

Modelo declarativo extendido con estado explícito y con hilos. Contiene la programación concurrente orientada a objetos. Razonar con este modelo es muy complejo pues pueden existir múltiples historias interactuando en formas impredecibles.

Modelo relacional

Modelo declarativo extendido con búsqueda. Permite la programación con relaciones, abarcando la programación lógica no-determinística. Precursor de la programación con restricciones.

Modelos extendidos

Modelo secuencial declarativo

Abarca la programación funcional estricta y la programación lógica determinística. El comportamiento de un componente es independiente del momento en que se ejecuta o de lo que pasa en el resto de la computación.

Modelo concurrente declarativo

Modelo declarativo extendido con hilos explícitos y computación by-need. Concurrencia dirigida por los datos, y concurrencia dirigida por la demanda. Los componentes interactúan a través del uso y la ligadura de variables de flujo de datos compartidas.

Modelo concurrente por paso de mensajes

Modelo declarativo extendido con comunicación por canales (puertos). Permite restringir el no-determinismo observable a pequeñas partes del programa.

Modelo con estado

Modelo declarativo extendido con estado explícito. Permite expresar la programación secuencial orientada a objetos. Permite al componente llevar una "historia," la cual lo deja interactuar con su ambiente, adaptándose y aprendiendo de su pasado

Modelo concurrente con estado compartido

Modelo declarativo extendido con estado explícito y con hilos. Contiene la programación concurrente orientada a objetos. Razonar con este modelo es muy complejo pues pueden existir múltiples historias interactuando en formas impredecibles.

Modelo relacional

Modelo declarativo extendido con búsqueda. Permite la programación con relaciones, abarcando la programación lógica no-determinística. Precursor de la programación con restricciones.

Modelos extendidos

Modelo secuencial declarativo

Abarca la programación funcional estricta y la programación lógica determinística. El comportamiento de un componente es independiente del momento en que se ejecuta o de lo que pasa en el resto de la computación.

Modelo concurrente declarativo

Modelo declarativo extendido con hilos explícitos y computación by-need. Concurrencia dirigida por los datos, y concurrencia dirigida por la demanda. Los componentes interactúan a través del uso y la ligadura de variables de flujo de datos compartidas.

Modelo concurrente por paso de mensajes

Modelo declarativo extendido con comunicación por canales (puertos). Permite restringir el no-determinismo observable a pequeñas partes del programa.

Modelo con estado

Modelo declarativo extendido con estado explícito. Permite expresar la programación secuencial orientada a objetos. Permite al componente llevar una "historia," la cual lo deja interactuar con su ambiente, adaptándose y aprendiendo de su pasado

Modelo concurrente con estado compartido

Modelo declarativo extendido con estado explícito y con hilos. Contiene la programación concurrente orientada a objetos. Razonar con este modelo es muy complejo pues pueden existir múltiples historias interactuando en formas impredecibles.

Modelo relacional

Modelo declarativo extendido con búsqueda. Permite la programación con relaciones, abarcando la programación lógica no-determinística. Precursor de la programación con restricciones.

Modelos extendidos

Modelo secuencial declarativo

Abarca la programación funcional estricta y la programación lógica determinística. El comportamiento de un componente es independiente del momento en que se ejecuta o de lo que pasa en el resto de la computación.

Modelo concurrente declarativo

Modelo declarativo extendido con hilos explícitos y computación by-need. Concurrencia dirigida por los datos, y concurrencia dirigida por la demanda. Los componentes interactúan a través del uso y la ligadura de variables de flujo de datos compartidas.

Modelo concurrente por paso de mensajes

Modelo declarativo extendido con comunicación por canales (puertos). Permite restringir el no-determinismo observable a pequeñas partes del programa.

Modelo con estado

Modelo declarativo extendido con estado explícito. Permite expresar la programación secuencial orientada a objetos. Permite al componente llevar una "historia," la cual lo deja interactuar con su ambiente, adaptándose y aprendiendo de su pasado

Modelo concurrente con estado compartido

Modelo declarativo extendido con estado explícito y con hilos. Contiene la programación concurrente orientada a objetos. Razonar con este modelo es muy complejo pues pueden existir múltiples historias interactuando en formas impredecibles.

Modelo relacional

Modelo declarativo extendido con búsqueda. Permite la programación con relaciones, abarcando la programación lógica no-determinística. Precursor de la programación con restricciones.