Detección y resolución de conflictos en diseño distribuido

José D. Ceroni
Alvaro Velásquez C.
y
Guillermo Bustos R.
Escuela de Ingeniería Industrial, Universidad Católica de Valparaíso
Valparaíso, Chile

RESUMEN

Cada vez más, productos complejos tales como autos, computadores y software, están siendo diseñados utilizando procesos de colaboración distribuidos. Un factor clave en la efectividad de dichos procesos es el manejo de los conflictos que aparecen principalmente por la interacción entre los diseñadores. Se debe desarrollar nuevos enfoques para apoyar efectivamente al diseño. Las funciones de apoyo deben estar orientadas principalmente a evitar, identificar y resolver conflictos a lo largo de todo el proceso. En este estudio se presenta las funciones de apoyo debe cumplir un sistema de detección y resolución de conflicto. Se presenta también un prototipo del modelo propuesto y los resultados experimentales obtenidos.

Palabras Claves: Diseño distribuido, conflicto, colaboración, apoyo al diseño, toma de decisiones.

1. INTRODUCCIÓN

Los métodos de diseño clásicos pueden ser caracterizados como seriales e iterativos, esto ha originado tiempos de desarrollo extensos y la posibilidad de generar productos finales con incompatibilidades de diseño, baja calidad y alto costo de realización debido a la repetición de actividades de diseño[1].

Dada la alta complejidad de los productos actuales, su desarrollo se ha vuelto progresivamente una tarea paralela y cooperativa que requiere de una coordinación efectiva para enfrentar las interdependencias altamente complejas, originadas por la distribución de los diseñadores participantes [2]. Distintas tecnologías de soporte para la coordinación han sido presentadas en la literatura, pero la mayoría de ellas poseen limitaciones. En este artículo se presenta un modelo integrado de Detección y Resolución de Conflictos para asistir a diseñadores.

En diseño cooperativo, se debe detectar y resolver los conflictos que se presenten en el transcurso del proceso de diseño. Los conflictos son inherentes a esta forma de trabajo y pueden perjudicar los resultados de la utilización de recursos o la calidad del diseño en desarrollo.

2. DETECCIÓN Y SOLUCIÓN DE CONFLICTOS

Los procesos que se llevan a cabo a lo largo de una instancia de diseño se asocian en grupos, con tablas ponderadas que presentan los pro y contra relativos a la utilización de uno u otro proceso para la definición de características, descripciones o componentes [3].

Debe llevarse a cabo un proceso de búsqueda para luego resolverlo mediante un proceso de resolución de conflictos automático [4].

El proceso de manejo de conflictos que se presenta en la Figura 1 consiste en las siguientes sub-tareas:

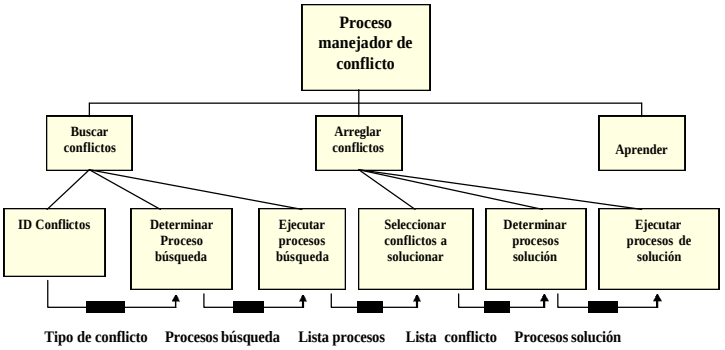

Figura 1 – Proceso de manejo de conflictos

Identificar los conflictos: Esta tarea consiste en decidir qué clases de conflicto resolverá el proceso de solución de conflictos.

Determinar procesos de búsqueda: Esta etapa determina que método de búsqueda (anticipación o detección) de conflictos se utilizará.

Ejecutar procesos de búsqueda: Esto implica poner en marcha el proceso de búsqueda identificado en la etapa anterior entregando como salida una o más instancias de conflicto.

Seleccionar conflictos a solucionar: Consiste en seleccionar los conflictos a resolver.

Determinar los procesos de solución: Se determina los procesos de solución que se usarán para manejar los conflictos seleccionados.

Ejecutar los procesos de solución: Utiliza los procesos seleccionados para completar el manejo de conflictos detectados por el sistema.

Aprendizaje: Mantiene la Base de Conocimientos en función de las decisiones de diseño tomadas.

Procesos de detección y solución de conflictos

A continuación se presenta algunas estrategias para manejar los conflictos, clasificadas en tipos de conflicto. Constituyen sólo una muestra de las alternativas posibles.

Conflicto de módulos

Detección: Se origina por la interacción entre diseñadores. La modificación de un módulo existente genera un conflicto con el original. Este conflicto se presenta solo si no existen conflictos de restricción ni compatibilidad.

Solución: La resolución del conflicto se basa en la contribución del módulo propuesto a los objetivos jerarquizados.

Los objetivos de las especificaciones se muestran de la forma:

Módulo Padre: Objetivo X debe al menos ser satisfecho Z

Módulo original y editado: El módulo satisface Y al Objetivo X

Los valores Y y Z son recuperados de la matriz de aprendizaje,

x_i: ponderación del objetivo i, i=1..n

De esta forma se plantea el siguiente algoritmo:

SI (x_i*Z1+x_{i+1}*Z2 + ... + x_n*Z_n) > (x_i *Y1 + x_{i+1}*Y2 + ... + x_n*Y_n) ENTONCES
    INCLUIR MODULO()
    REVISAR_POR_CONFLICTOS()
SI (LISTACONFLICTOS = 0) ENTONCES
    INCLUIR_MODULO()
SINO
    NO_INCLUIR_MODULO_NUEVO()
FIN-SINO
FIN-SI

Una vez resuelto el conflicto, la función de aprendizaje modificará los valores de selección de los módulos aceptados y rechazados.

Conflicto de restricción

Detección: Una o más restricciones del módulo padre no han sido respetadas. Se comparan las restricciones según el comparador de restricción.

Solución: El conflicto se resuelve impidiendo al diseñador ingresar las especificaciones que no satisface la restricción.

Conflicto de omisión

Detección: El producto no requiere de una característica en particular para lograr un mínimo funcionamiento, sin embargo la falta de ella implica no satisfacer uno o más objetivos. Este conflicto se presenta al dar finalizado el proceso diseño y el sistema, al leer los objetivos y revisar por los componentes que lo satisfacen, no se ha encontrado alguna correspondencia en los módulos.

Solución: Se informa al supervisor que según los objetivos globales del producto y/o de algún módulo específico, no están presentes los componentes o características.

Conflicto de compatibilidad

Detección: Al ingresar o editar un nuevo módulo, resulta incompatible. Las incompatibilidades pueden ser de conexión o funcionales. Se encuentra al comparar las interacciones del diseño con la matriz de compatibilidad.

Solución: En base a la incompatibilidad encontrada, se instruye al diseñador a buscar otro módulo que sea compatible.

Conflicto de integridad

Detección: El producto no cumple con los requisitos básicos para estar terminado. La ausencia de módulos o no satisfacción de especificaciones impiden la utilización del producto. Para poder cumplir este fin, se revisa una lista de todos los componentes básicos que son imprescindibles para que el producto esté completo.

Solución: Se alerta al supervisor de la falta de módulos faltantes o especificaciones no satisfechas.

3. BASES DE UN SISTEMA DE DISEÑO COOPERATIVO

Para realizar la detección y resolución de conflictos en diseño cooperativo se debe disponer de una estructura para las funciones de apoyo [5].

Red de diseñadores

Para realizar el diseño concurrente lo diseñadores deben operar en una plataforma computacional conectada en red. Además de permitir el diseño en paralelo, una estructura de red facilita la interacción entre los diseñadores, disminuyendo dificultades de distancia y tiempo [6].

Interfaz para ingreso de especificaciones de diseño

Las especificaciones de diseño son ingresadas al sistema por los diseñadores. El formato de las especificaciones debe ser lógico, cómodo y simple para el diseñador, y a la vez suficientemente estructurado, preciso y significativo para que el sistema manipule los datos [6].

Describir las acciones de diseño requiere de una interfaz efectiva entre los diseñadores, lo que implica que éstas deben estar basadas en el nivel de tarea. Las interfaces a nivel de tarea permiten al usuario interactuar con el sistema en términos de entidades (recursos, componentes o características) apropiadas para la tarea en particular que se está realizando. Esto permite a los diseñadores comunicarse en un dominio natural y estructurado [7].

Los productos en diseño pueden representarse como un conjunto de módulos, cada uno con sus atributos característicos, cuyas interfaces se conectan. El módulo que posee ramas descendentes es llamado módulo padre, y los módulos conectados a dichas ramas, módulos hijos, que a la vez pueden ser padres de otros módulos. Los recursos usados por un módulo se representan mediante atributos [8].

La descripción debe comenzar con uno o más módulos representando el producto con las especificaciones que representan los valores deseados en los atributos del módulo. Esto evoluciona en descripciones más detalladas al restringir el valor de los atributos de los módulos, al conectar las interfaces de módulos (para representar las interacciones), al descomponer los módulos en sub-módulos y al especializar los módulos al refinar su clase. Los valores de las especificaciones y los atributos se representan utilizando un lenguaje estructurado [9].

Matriz de compatibilidad

Es una matriz simétrica en donde las filas y columnas representan los atributos y componentes. Al existir compatibilidad entre dos atributos y/o componentes, el valor de la celda es 1, si no existe compatibilidad el valor es 0. Un valor de 2 indica que no se aplica una relación de compatibilidad entre ambos componentes [5].

Matriz de descripción

Contiene los valores cuantitativos que describen un módulo. Los valores describen propiedades que permiten comparar cuantitativamente distintos componentes y verificar el cumplimiento de objetivos cuantitativos asociados a restricciones. [5]

Matriz de aprendizaje

Contiene valores desde 0 hasta 1 que representan el no cumplimiento hasta el cumplimiento total respecto de un objetivo. Los valores son actualizados por la función de aprendizaje.

Las tres matrices definidas conforman la Base de Conocimientos (BC) y son capaces de vincular las preferencias de los diseñadores y la información complementaria no especificada con los datos concretos que presentan los atributos de los componentes. Utilizando los datos de la BC, las decisiones de diseño pueden ser apoyadas, al beneficiarse tanto por la experiencia como por la información completa de las características de los componentes.

Objetivos del diseño

Los objetivos que dependen de preferencias de diseñadores o de requerimientos de usuarios se denominan objetivos cualitativos. Éstos objetivos pueden ser manejados con los conceptos de Lógica Difusa, en función que consideran un rango de valores de verdad. Por otro lado, los objetivos cuantitativos son específicos y obedecen a la lógica convencional.

Jerarquía de objetivos

El supervisor es el encargado de coordinar el proceso de diseño y determinar su finalización. Cuando ingresa los parámetros del diseño está registrando el deseo de que el producto cumpla algunos objetivos. Sin embargo lograr el cumplimiento total de todos los objetivos no es posible puesto que cualquier decisión de diseño conlleva la inclusión de elementos deseables y también no deseables. Esto hace necesario jerarquizar los objetivos. Al conocer cuáles objetivos son prioritarios, los procedimientos de solución de los conflictos pueden basarse en la importancia relativa que éstos tienen.

Los componentes del sistema desde la especificación de los diseños hasta la jerarquización de los objetivos tratan sobre la base estructural que le permiten recibir las propiedades, sus valores, interacciones entre propiedades, definir tipos de objetivos y restricciones. Sobre ésta base se construyen los procedimientos de detección y solución de conflictos.

El aprendizaje del sistema

Las especificaciones detalladas de componentes y características son necesarias pero no suficientes para apoyar en la decisión de utilizar un componente o una propiedad de éste en el diseño. Las especificaciones permiten conocer que un módulo es mejor que otro en propiedades particulares, pero éstas son independientes del contexto en el que el módulo estará inmerso. Así, no es posible modelar todas las variables que se involucran en la decisión de aceptación o rechazo de un módulo. Debido esto, el enfoque se basa en las preferencias de los diseñadores por ciertas propiedades y módulos. Un diseñador experto en un tema tiene claro qué propiedades permiten o han permitido históricamente que un componente sea mejor que otro.

El método de aprendizaje

Las ponderaciones en la matriz de aprendizaje reflejan la preferencia por parte de los diseñadores de usar o rechazar un componente determinado al diseñar un producto, para un objetivo particular. Cada componente posee una ponderación relacionada con su cumplimiento para cada objetivo que se ha ingresado al sistema, de acuerdo a la preferencia del diseñador que lo ha agregado. La ponderación variará, mediante la función de aprendizaje, de acuerdo a las decisiones que tomen los diseñadores. Cuando el diseñador agrega la ponderación de un componente, cuenta con un 100% de certeza que el componente va a ser preferido con una ponderación X (razón de preferencia). Esta certeza representa la velocidad con que el sistema aprende de las decisiones tomadas por los diseñadores y debe ser ajustada tomando en cuenta el contexto en el que se trabaja, el tipo de módulo o propiedad. En caso de rechazo la función de aprendizaje resta el porcentaje correspondiente del total de oportunidades en las cuales el módulo se ha elegido.

4. EVALUACIÓN DEL SISTEMA

Para evaluar la efectividad del sistema de detección y solución de conflictos se construyó un prototipo basado en una estructura cliente/servidor y que consiste en una interfaz Web conectada a una BC. La naturaleza concurrente de una estructura basada en Web (Figuras 2 y 3) facilita la interacción de los diseñadores.

Mientras se realiza el diseño el prototipo verificará la existencia de conflictos, proponiendo las soluciones correspondientes.

Figura 2 – Interfaz del prototipo

Figura 3 – Interfaz del prototipo

Diseño del experimento

Para verificar la efectividad del sistema se realizó diseño de experimentos que considera los siguientes factores:

A. Complejidad del producto, expresada por el número de módulos y conexiones.

B. Número de diseñadores involucrados en el diseño.

Para el factor A se considera 2 niveles
a_1: Complejidad Alta, 26 módulos
a_2: Complejidad Baja, 17 módulos.

Para el factor B se considera 3 niveles:

b_1: Alto, 9
b_2: Medio, 6
b_3: Bajo, 3.

Las variables de respuesta a considerar en el experimento son:

1. Razón tiempo total a demoras por conflictos:

$$\frac{\text{Tiempo total en que se realiza el diseño}}{\text{Tiempo en Stand By por solución a conflictos}}$$

2. Razón de conflictos solucionados:

$$\frac{\text{Número de conflictos solucionados por el sistema}}{\text{Número de conflictos que han aparecido}}$$

Se realizó 18 experiencias correspondientes a las a b condiciones experimentales y 3 réplicas, requeridas para obtener una estimación del error experimental.

En el primer análisis se estudia si el factor A, B o la interacción de éstos inciden en una diferencia entre las medias de las muestras de eficiencia respecto a la razón 1.(Tabla 1). En el segundo, se verifica para la eficiencia del sistema respecto a la razón 2 (Tabla 2).

Luego de efectuado el experimento los resultados fueron los siguientes:

Para el tiempo total a demoras por conflictos:

Factor Complejidad	Factor n° diseñadores	Réplica 1	Réplica 2	Réplica 3
A1 (26)	B1 (9)	5.2444	5.2445	5.2398
A1 (26)	B2 (6)	4.3256	4.2976	4.3254
A1 (26)	B3 (3)	3.4474	3.4369	3.4241
A2 (17)	B1 (9)	7.5385	7.5112	7.4518
A2 (17)	B2 (6)	6.8261	6.8385	6.8104
A2 (17)	B3 (3)	5.1111	5.0949	5.1055

Tabla 1 – Resumen de resultados para razón 1

Para los conflictos solucionados:

Factor Complejidad	Factor n° diseñadores	Réplica 1	Réplica 2	Réplica 3
A1 (26)	B1 (9)	0.9189	0.9254	0.9332
A1 (26)	B2 (6)	0.9167	0.9198	0.9223
A1 (26)	B3 (3)	0.9259	0.9313	0.9351
A2 (17)	B1 (9)	0.9545	0.9559	0.9552
A2 (17)	B2 (6)	0.9423	0.9434	0.9630
A2 (17)	B3 (3)	0.9556	0.9535	0.9778

Tabla 2 – Resumen de resultados para razón 2

Realizadas las pruebas, y de acuerdo a los datos recopilados y a los análisis estadísticos sobre las muestras, se concluye que, si bien por una parte el sistema disminuye su rendimiento al aumentar la complejidad del producto, apoya eficientemente a una mayor cantidad de diseñadores. Así, se concluye que se puede brindar apoyo efectivo al diseño distribuido al manejar los conflictos que aparecen a lo largo del proceso de diseño.

5. CONCLUSIONES

La colaboración de diseñadores se ve impactada negativamente por la aparición de conflictos. Los conflictos afectan la calidad y tiempo de diseño, y malgastan recursos. Se ha propuesto un sistema para detectar y resolver conflictos en diseño distribuido, el sistema es capaz de resolver seis tipos de conflictos que se asume más recurrentes en la actividad de diseño. La verificación del sistema consistió en programar un software prototipo que implementase las reglas y acciones de la detección y solución de conflictos. La experimentación realizada con el modelo permite concluir que los conflictos se resuelven automáticamente en un 90% de las ocasiones.

6. REFERENCIAS

[1] Bonnardel, Natalie; Sumner, Tamara. Supporting Evaluation in Design. Publicado en Acta Psychologica 91 (1996) 221-244.

[2] Klein, Mark. Conflict Resolution in Cooperative Design. Department of Computer Science, University of Illinois 1990

[3] Lara, M. A. Conflict Resolution of Collaborative Facility Design. In: PRISM 98, 1998 p.41-42.

[4] Nakakoji, Kumiyo; Sumner, Tamara. Perspective Based Critiquing: Helping Designers Cope With Conflicts Among design Intentions. Departament of Computer Science and Institute of Cognitive Science, University of Colorado.

[5] Díaz, Cristián y Velásquez, Álvaro. Desarrollo de un sistema de apoyo para la resolución de conflictos en diseño distribuido. Escuela de Ingeniería Industrial, Universidad Católica de Valparaíso, Valparaíso 2001.

[6] Klein, Mark. Supporting Conflict Management in Cooperative Design Teams. Boeing Computer Services, Seattle, 1993.

[7] Klein, Mark. Supporting Conflict Resolution in Cooperative Design Systems. Advanced Research Lab, Hitachi Ltd. Hatoyama, Japan 1996.

[8] Klein, Mark. Towards a Systematic Repository of Knowledge About Managing Collaborative Design Conflicts. Center for Coordination Science, Massachusetts Institute of Technology, Boston 2000.

[9] Edwards, Keith. Flexible Conflict Detection and Management In Collaborative Applications. Xerox Palo Alto Research Center. Palo Alto, California, octubre 1997.