패트리 넷을 통한 SOFA/DCUP의 동적 컴포넌트 업데이트에 대한 상호관계 분석

김천호* ,정화영** ,송영재*
경희대학교 컴퓨터공학과*, 예원예술대학교 전자상거래학과**
kch1115@cvs2.khu.ac.kr, jmichael@hanmir.com, yjsong@khu.ac.kr

An Inter-Relation Analysis of Dynamic Component Update in SOFA/DCUP using PetriNets

ChunHo Kim*, HwaYoung Jeong**, YoungJae Song*
Dept. of Computer Engineering, Kyung Hee Univ*, Dept. of Electronic Commerce Yewon Art Univ**

요 약

기존의 SOFA(SOFtware Appliances)형의 컴포넌트에서 DCUP(Dynamic Component UPdating)통해 컴포넌트 조립이 가능하다. 이에 대해 비동기적이고 동적 시스템에 적합한 패트리 넷을 이용하여 컴포넌트의 동적 업데이 시 기존 컴포넌트와의 상호작용과 각 인터페이스에 대한 프로세스의 흐름을 정형적으로 정의하고 설계상의 모호성을 줄일 수 있도록 하였으며 보다 나은 동적 업데이트의 최적화를 위한 기반을 제공한다.

1. 서 론

최근 엔지니들과 연구원들의 프로그램과 디자인기술은 컴포넌트 기반에서 많이 이루어진다. [1] 이에 따라 동적인 컴포넌트 시스템의 중요성도 커지고 있다. SOFA/DCUP는 시스템이 실행 중에도 컴포넌트를 업데이트 할 수 있는 기반을 제공한다. 하지만 동시적 시스템의 복잡성으로 인해 그 표현이 어려웠다. 이러한 동시 병행적이고 복잡한 시스템의 표현과 다양한 자원을 행위기반을 통해 패트리 넷으로 모델링 하는데 유리하다. [3] 패트리 넷을 통해 컴포넌트의 동적 업데이트에 대한 기존 컴포넌트 간의 상호작용에 대한 에러 및 모호성에 대한 정형적 정의를 통해 SOFA/DCUP의 동적 업데이트에 대해 보다 명확하고 명세적인 표현을 하였다.

2. 관련 연구

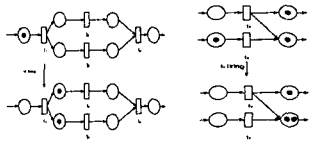
2.1 Petri Nets

패트리 넷은 비동기적이며, 동시 발생적인 이벤트 시스템처럼 상태가 동적으로 변화하는 시스템을 모델링 하는데 유리한 도구이다.

패트리 넷은 다섯 개의 tuple로 구성되어 있다.

- $PN=\{P,T,F,W,M_0\}$
- $P = \{p_1, p_2, p_3 \cdots\cdots p_n\}$: Place의 유한 집합 ($n \geq 0$)
- $T = \{t_1, t_2, t_3, \cdots\cdots t_m\}$: Transition의 유한 집합($m \geq 0$)
- $F \subseteq (P * T) \cup (T * P)$: Arc의 집합, P→T &T→P로 가는 집합
- W: F → {1, 2, 3, ...}: 가중치 함수
- M_0: P→ {0, 1, 2, 3 .....}: 초기 마킹
- $P \cap T = \varnothing$ & $P \cup T \neq \varnothing$

여기서 Place는 시스템의 상태 또는 조건을 나타내며 원으로 표시한다. Transition은 시스템의 상태의 변화 또는 이벤트를 나타낸다. M은 마킹으로 각 Place의 토큰에 대한 상태를 표시하며, 토큰은 Place의 조건의 진위 또는 시스템의 가용자원을 나타내며, 동적 시스템 및 병행적인 특성을 나타내기 사용된다. 이벤트가 발생할 때 필요한 조건을 만족할 경우 Place에 토큰을 설정한다. 트랜지션은 자신에게 입력되는 모든 Place에 토큰이 존재 할 때 점화 할 수 있다. Transition이 점화 되면 P에서 T로의 입력 Place에서 토큰 제거되고 T에서 P로의 출력 Place에 토큰을 생성한다. 토큰의 생성과 제거로 인해 하나의 Place에 2개 이상의 토큰을 가진 Place가 생성되거나, 토큰이 존재하지 않는 Place도 생성 될 수도 있다.

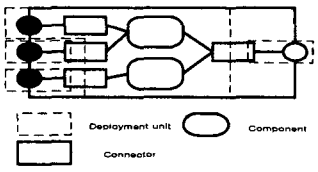


<그림 1> 패트리 넷에서 토큰 흐름

2.2 SOFA/DCUP

SOFA기반의 컴포넌트는 동적 업데이트 동안에 안전한 상태 변화를 위한 아키텍처를 제공하며, 이 구조는 일정한 프레임영역 안에서의 업데이트가 가능하도록 하였다. 사용자가 컴포넌트의 수정이 없이 자체 채널을 통해 동적 업데이트가 가능한 환경을 제공한다. [2][4]

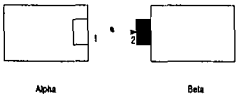
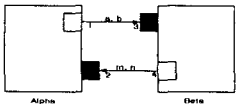
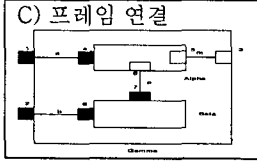
SOFA형의 컴포넌트 구조는 내부에 템플릿 저장소(TR: Template Repository)가 존재하며 외부와는 직접적인 템플릿을 설정 할 수 없다.



<그림 2> SOFA/DCUP 예

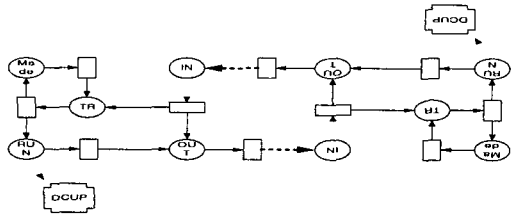
3. 패트리 넷을 통한 SOFA/DCUP 연구

SOFA/DCUP에서 컴포넌트를 업데이트 하는 경우 다음의 기본적인 3가지 경우가 존재한다.

A) 단방향 연결 	ProtAlpha = !a =!a↑ :?a↓ ProtBeta = ?a = ?a↑ :!a↓
B) 양방향 연결 	1) ProtAlpha = !a{?m + ?n} ProtBeta = ?a{!m} + ?b{!n} 2) Prot' Alpha = !a{?m}!b{?n} Prot' Beta=?a{!m+!n}+?b{!n}
C) 프레임 연결 	ProtAlpha = ?a:!p:!m ProtBeta = ?b:?p I) ProtGamma = ?b:?a:!m II) Prot' Gamma = ?a:?b:!m

<그림 3> SOFA 기본 연결형태 및 Behavior 프로토콜

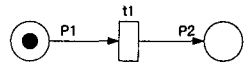
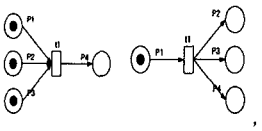
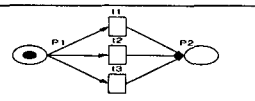
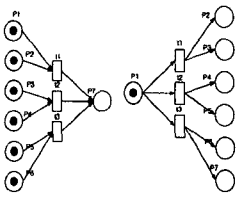
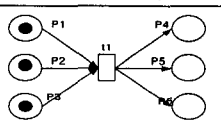
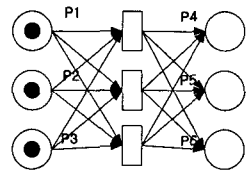
<그림 3>에서 A)형태를 보면 컴포넌트가 단방향으로 연결 되어 있으며 Alpha경우 !a=!a↑ :?a↓ 로 표현 되며 !a↑ 는 메시지가 한 방향으로 나가는 것을 표현 한다. 컴포넌트 Alpha의 기능은 인터페이스1을 통해 Beta의 인터페이스 2에게 요구하는 것을 제공한다. 유사한 형태로 B)의 경우, 컴포넌트 Alpha의 인터페이스 1은 a, b를 3을 통해 콜하고, Beta는 인터페이스4를 통해 Alpha의 인터페이스 2에게 응답을 한다.



<그림 4> 그림3의 B의 페트리 넷

C)의 경우 프레임안에 서브 컴포넌트는 인터페이스를 제공하고 부모의 컴포넌트의 인터페이스를 요구 한다. 예를 들면 6→7, 1→4,2→8,5→3을 들 수 있다. 컴포넌트 프레임의 바운더리안에서 Gamma프레임의 인터페이스 1,2,3의 형성 되는 동안 Alpha 4,5,6 그리고 Beta 7,8에 의해 프레임이 형성된다. Alpha와 Beta의 형성에 의해 Gamma의 구조가 형성되며, 프레임에 의해 Alpha, Beta와 Gamma가 연결이 된다. 이러한 컴포넌트간의 결합은 컴포넌트를 동적 업데이트 할 때 중요하다.

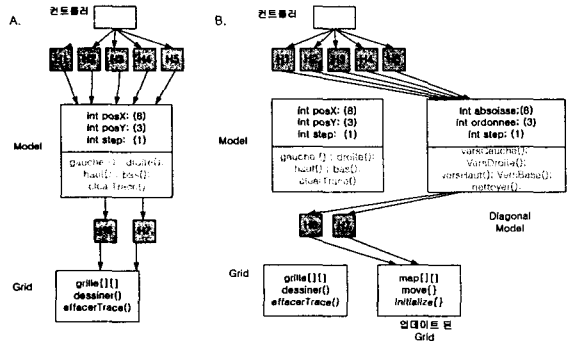
SOFA 형에서 컴포넌트의 업데이트 경우 <그림3> A)의 경우 같은 경우 2개의 컴포넌트로 간단히 표현 된다. <그

그래프	관계
 1:1:1	1) if M_0 (1,0), P1→t1 firing M_1 (0,1) 토큰 1개로 일정.
 M:1:1 1:1:M	2) if M_0(1,1,1,0) P1,P2,P3→ t1 firing M_1(0,0,0, □) 전위, 후위의 토큰 수 증가 또는 감소, $M_0 \neq M_1$
 1:M:1	3) if M_0 (1,0), P1→t1,t2, t3 firing M_1 (0,1) 결과는 1)과 같지만 토큰에 대한 누적 위험성 존재
 M:M:1 1:M:M	4) if M_0(1,1,1,1,1,1,1) P1,P2,P3,P4,P5,P6..→ t1,t2,t3firing M_1(0,0,0,0,0,0,□) 토큰 약의 급속한 감소 또는 급속한 증가로 인한 메모리 및 버퍼 관리 필요
 M:1:M	5) if M_0(1,1,1,0,0,0) P1,P2,P3→ t1 firing M_1(0,0,0,□ ,□ ,□) 동일 토큰 수를 가진 Place 증가, 입력 Place개수 = □
 M:M:M	6) if M_0(1,1,1,0,0,0) P1,P2,P3→t1,t2,t3firing M_1(0,0,0,□ ,□ ,□) -입력 Place 토큰=1 : 입력 Place 개수 = □ -입력Place의 토큰=2or3 : 입력 Place의 모든 토큰 개수 = □ -입력Place토큰이 4개 이상일 경우 : transition의 개수 보다 많으므로 제한에 걸리기 때문에 토큰이 3일 경우와 같다.

<그림 5> SOFA/DCUP 컴포넌트 업데이트 유형

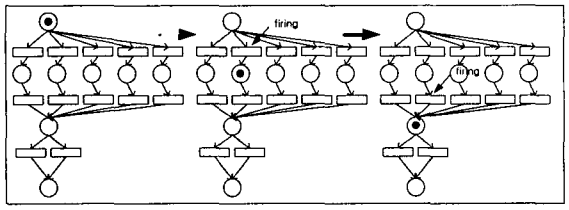
림3>을 확장하여 패트리 넷을 통해 프로세스의 흐름이나, 컴포넌트(transition)의 업데이트에 따른 관계 및 동적 업데이트에 대한 표현을 보다 다양하고 정확한 표현을 하였다. Place와 Transition으로 표현 할 경우 (Place2개, Transition1개) 또는 (Place1개, Transition2개)로 표현하게 되며, 각 단계와 그 다음 단계와의 관계는 1:1 또는 1:다수 의 경우가 존재하게 되며 즉, 3개 원소에 2가지의 경우 수가 존재하므로 총 2^3개가 존재한다.

다음은 Bean 형태의 컴포넌트를 동적 업데이트하는 예이다.



<그림 6> SOFA형 Bean의 동적 업데이트

기존 빈은 업데이트가 가능하게 명세 되어 있어야 하며 기존 빈과 업데이트 되는 빈과 명세가 제대로 성립되지 않았다면 사용자가 명세를 해주어야 한다. 빈1은 소스(H1,H2,H3,H4,H5) 안되도록 처리하고 빈1 상태는 빈2의 맵핑 된 스크리트에 따라서 전송된다. 새로운 빈의 레퍼스 업데이트는 기존의 타겟 빈을 레퍼런스의 소스로부터 접속되어 전송된다. 기존 타겟 메소드 레퍼런스는 빈과 어뎁터 사이에 내부커넥터에 따라서 새로운 타겟 메소드들과 대응하여 업데이트 된다. 어떻게 내부 연결이 적용의 수행을 위해 동적인 수정을 할 수 있는지를 보여준다. 빈2와 빈1의 소스와 접속으로 모든 타겟 빈2는 H6,H7과 접속 가능 하게 된다.

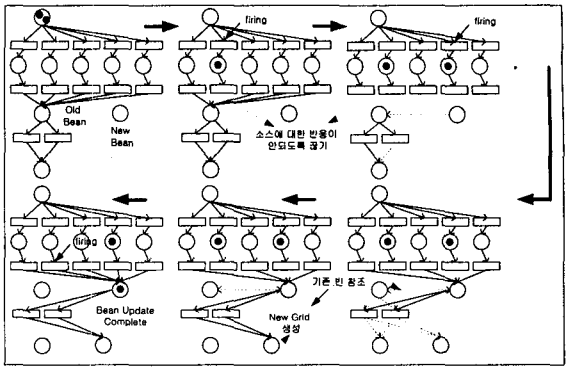


<그림 7> 그림 6.A의 프로세스 흐름

<그림 6> A.의 경우 각 트랜지션의 firing에 따라 토큰이 움직이는 것을 볼 수 있다. 새로운 빈을 동적 업데이트를 할 경우 <그림7>에서 확인 할 수 있다.

<그림 8>는 동시적인 프로세스 발생에 따른 변화를 보여주고 있으며, Old Bean이 소스에 대한 반응을 제거를 통해 New Bean을 업데이트가 가능하게 된다.

또한 기존 빈에 대한 레퍼런스를 통해 새로운 그리드를 자동으로 생성하게 된다.



<그림 8> 그림6.B에 대한 동적 빈 업데이트 흐름

4. 결론

지금까지 본 논문에서는 SOFA/DCUP에 대한 컴포넌트의 동적 업데이트를 패트리 넷을 통해 분석하였다. 동적 업데이트를 할 때의 컴포넌트 간의 연결관계와 업데이트 시 프로세스의 흐름에 대해 알아 보았고 기존SOFA/DCUP의 정형적인 정의를 통해 설계상의 모호성을 줄일 수 있도록 하였으며, 보다 나은 형태의 최적화 된 조합 될 수 있는 기반을 제공한다. 또한 컴포넌트를 동적 업데이트 하는 것은 어려움이 많으므로 본 정의를 통해 다양한 요구 사항을 수용 할 수 있도록 하여 컴포넌트의 업데이트에 대한 일반성을 더욱 향상 시킬 수 있다.

향후 연구 과제는 컴포넌트 업데이트 시 인터페이스의 변경을 확대 할 수 있도록 컴포넌트 내부의 메시지 흐름과 변경 할 수 있는 메시지 흐름 에 대한 다양한 정형적인 흐름과 정의를 통해 컴포넌트 간에 동적 업데이트 시 메시지 에러 및 검증에 대한 기법을 연구 할 것 이다.

참고문헌

[1] Wolfgang Emmerich, Nima Kaveh, " Component technologies: Java beans, COM, CORBA, RMI, EJB and the CORBA component model" ACM Press New York, NY, USA Pages: 691 - 692, 2002

[2] N. D. Hoa. Dynamic aspects in SOFA/DCUP. Technical Report Jun. 07 .2002

[3] J.-E.Hong and D.-H.Bae, " High-level Petri net for incremental analysis of object-oriented system requirements" IEE Proc-Softw., Vol 148. No. 1. P 11-18, February 2001

[4] Adamek, J., Plasil, F.: Behavior Protocols Capturing Errors and Updates, Proceedings of the 2na International Workshop on Unanticipated Software Evolution, ETAPS, Warsaw, 2003