높은 정도의 재설정을 요구하는 임베디드 소프트웨어를 위한 도구의 지원

강우철° 윤희철 김가규 이형석 김선자

한국전자통신연구원(ETRI)

{wchkang°,hcyun,ggkim,hyslee,sunja}@etri.re.kr

Configuring highly reconfigurable embedded software with intelligent tool's support

Woochul Kang° Heechul Yun Gaggu Kim Hyungsuk Lee Sunja Kim

Embedded Software Center

Electronics and Telecommunication Research Institution

요 약

임베디드 소프트웨어가 점점 더 복잡해지고, 시장화시간(time-to-market)은 점점 짧아 짐으로써 임베디드 소프트웨어 디자이너들은 세부적으로 모듈화 되고 재설정 가능한 소프트웨어를 작성하도록 요구된다. 그러나 이렇게 재설정 가능한 모듈과 컴포넌트들이 급속히 늘어남에 따라 많은 모듈과 컴포넌트들을 설정하고 관리하는 것은 매우 복잡한 작업이 되고 있다. 특히, 모듈이나 컴포넌트간에 의존성이 있는 경우 이를 적절히 기술하고 체크 할 수 있는 방법이 요구된다. 본 논문은 재설정 가능한 모듈과 컴포넌트가 수작업으로 관리하기 어렵도록 높은 수준일 때 간단한 툴인 Target Builder를 이용해 이들간의 관계를 기술하고 관리하는 방법을 제시한다. 비록, 이 툴은 간단한 수준의 기능만을 제공하지만, 실제 개발 현장에 적용한 결과 높은 정도의 재설정을 요구하는 소프트웨어 시스템의 관리에 매우 적합하며, 기존의 소프트웨어 시스템에도 쉽게 적용될 수 있었다.

1. 서 론

대부분의 임베디드 시스템을 위한 소프트웨어는 매우 간단한 기능만을 요구했기 때문에 임베디드 시스템의 개발에서 모듈[1]이나 컴포넌트들에 대한 설정의 중요성은 대부분 무시되어 왔다. 그러나 임베디드 시스템이 점점 복잡해지고 다양한 기능을 요구함에 따라 임베디드 시스템에 탑재되는 소프트웨어들은 점점 많은 재설정 가능 컴포넌트를 가지게 되었다. 일반적인 소프트웨어에서 재설정 가능 컴포넌트는 <그림 1>에서 보는 것과 같이 매크로를 이용해 정의된다.

```
#ifdef COMP_A
...                          ;; 컴포넌트 A
#endif
#ifdef COMP_B
...                          ;; 컴포넌트 B
#end
```

<그림1> 매크로를 사용한 재설정 컴포넌트의 정의

설정툴은 이 매크로들이 정의한 심벌들의 값을 타겟 시스템이 요구하는 바에 부합하도록 값을 정해서 필요한 컴포넌트(기능)만을 선택하게 된다.

	컴포넌트 개수	최소 사이즈
Qplus-T(small RTOS)	> 50	50 KB
VxWorks(medium size RTOS)	> 300	200 KB
Qplus-P(embedded Linux)	> 2000	1,000 KB

<표 1> 상용 임베디드 운영체제들의 재설정 가능 컴포넌트의 개수

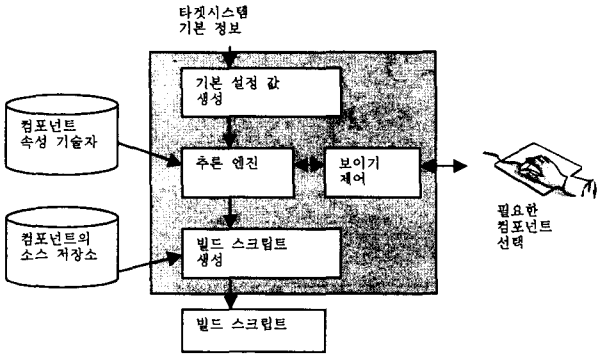
<표 1>은 상용 임베디드 운영체제들의 재설정 가능 컴포넌트의 개수를 보여준다. Qplus-T[7][8]와 같은 소형 운영체제의 경우 적은 개수의 컴포넌트만을 가지므로 설정툴과 같은 도구가 없이 수작업으로 설정이 가능하다. 그러나, Qplus-P(embedded Linux)[9]와 같은 고 기능의 임베디드 운영체제의 경우 설정툴의 도움없이는 수천개에 이르는 컴포넌트들을 타겟 시스템에 적절한 값으로 설정하기란 매우 어려운 일이다. 또한, 각 컴포넌트들간에는 의존성을 가지고 있는 경우가 있다. 예를 들어, 커널의 특정 컴포넌트는 다른 컴포넌트가 있어야만 제대로 동작하는 경우가 있다. 이렇듯 복잡화된 설정과정을 좀 더 쉽게 하기 위해서 다양한 도구들이 개발되었다[5]. 그러나 이들은 시스템에 대한 사양을 입력으로 하여 자

1) 본 논문에서 모듈과 컴포넌트는 매크로를 사용해 정의된 소프트웨어의 특정 부분(기능)을 가르키며 같은 의미로 사용된다.

동으로 설정을 생성해 내는 "black box"식의 방법을 취하고 있다. 그러나, 이러한 "black box"식의 방법은 새로운 컴포넌트들의 추가와 같은 작업이 매우 어려우며 일반적인 상용 임베디드 소프트웨어에 적용하기도 매우 어렵다[4].

본 논문에서는 일반적인 임베디드 소프트웨어에 쉽게 적용이 가능한 설정툴인 타겟빌더의 구조에 대해서 설명한다.

2. 타겟 빌더의 구조



<그림 2> 타겟 빌더를 이용한 설정과 빌드

그림 2는 타겟 빌더의 개략적인 구조를 보인다. 주요 구성요소는 기본설정 생성기, 실행엔진, 빌드 스크립트 생성기, GUI 인터페이스, 컴포넌트 속성 기술자들, 컴포넌트들의 소스이다. 사용자의 첫 입력은 자신이 원하는 타겟 시스템의 대략적인 사양을 입력하는 것이다. 즉, CPU 타입, 보드 종류 등을 입력하며 타겟 빌더는 해당 보드에 적합한 기본설정을 생성하고, 적합한 툴체인과 타겟에의 적재 방식등을 결정하게 된다. 실행엔진은 사용자가 선택한 컴포넌트간의 의존성을 체크하여 설정의 일관성을 유지시키는 역할을 한다. GUI 인터페이스는 현재설정에서 사용자에게 필요한 컴포넌트만 보이고 다른 것은 숨김으로써 사용자의 설정과정을 단순하게 만든다. 설정이 끝나면 선택된 각 컴포넌트들을 빌드하기 위한 스크립트와 헤더파일이 자동으로 생성된다. 각 컴포넌트는 소스형태로 제공되며, 컴포넌트간의 의존성을 비롯한 속성은 컴포넌트 기술자를 통해 기술된다.

3. 컴포넌트의 속성과 의존성에 대한 기술

타겟 빌더는 설정가능한 컴포넌트에 대한 기술을 위해 각 응용 패키지 별로 QPD(Qplus Package Descriptor)를 가진다. 그림 3은 TinyX라는 컴포넌트에 대한 속성을 기술한 예이다. %%requre은 컴포넌트간에 가지는 의존성을 기술한 것이다. 의존성의 기술은 컴포넌트간의 논리식으로 표현된다. 위의 예는 TinyX 라는 컴포넌트가 선택되기 위해서는 커널의 unix domain socket과 frame buffer 컴포넌트가 반드시 선택되어야 함을 보인다. %%unless ~suppre 속성은 인터페이스를 제어하기 위한 속성이다.

```
%component TINYX
%%desc the tiny x window system with minimum footprint
%%require 'UNIX_DOMAIN_SOCK' ==y and 'FB'==y
%%unless tinyx==y suppress 'GTK'
```

<그림 3> 설정 컴포넌트에 대한 기술

4. 인터렉티브한 의존성의 해결

타겟 빌더의 추론 엔진은 %%require에 속성된 컴포넌트간의 의존성을 자동으로 해결한다. 정확한 설정을 찾아내는 것은 전통적인 SAT(propositional satisfiability) 문제에 해당하므로 NP-complete문제에 해당한다. 따라서 GSAT[7]이나 SATO[6]같은 backtracking방식을 이용한 알고리즘들이 사용되어 왔다. 그러나 이러한 알고리즘 역시 문제 사이즈가 커질 경우 매우 많은 시간을 소모하게 된다. 따라서 타겟 빌더와 같은 인터렉티브한 설정툴에서는 사용하기 어렵다.

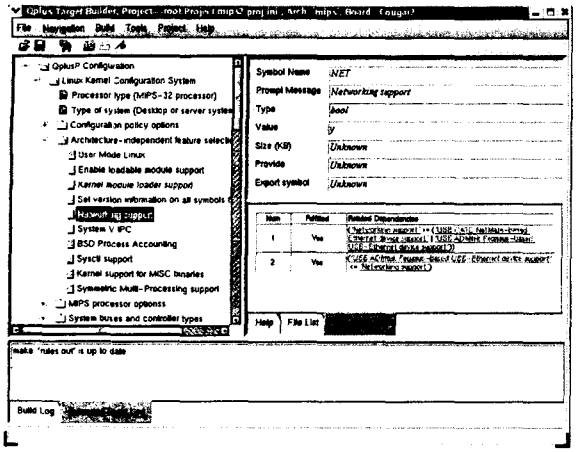
타겟 빌더는 대신 간단한 알고리즘을 사용하여 이 문제를 해결하는데, 이는 다음과 같은 특성에 기인한다. 첫째로, 우리는 SAT와 같이 조건을 만족하는지 여부에 대한 완전한 해를 원하는 것이 아니라, 많은 컴포넌트의 값이 불완전한 해(즉, 어떤 값이여도 상관없는 것)여도 상관이 없다. 둘째는, %%require문에 기술된 의존성은 대체로 X1=> X2 => ... =>Xn과 같은 체인을 형성한다는 점이다. 위 특성을 이용해서 우리가 적용한 알고리즘은 다음과 같다.

1. 컴포넌트의 값이 변하면, 추론 엔진은 해당 컴포넌트가 강제하는 다른 컴포넌트가 있는지 여부를 살핀다. 만일 다른 컴포넌트의 값을 강제한다면, 그 값을 바꾸고 'freezed' 라고 표시한다.
2. 위의 1 과정을 재귀적으로 적용한다. 만일, 'freezed'된 컴포넌트의 값을 현재의 값과 다른 값으로 바꾸려고 할 경우 설정이 잘 못 되었음을 알리고 롤백한다.
3. 만일 위의 1,2 과정이 에러없이 종료된 경우 의존성 룰들이 만족되는지 순차적으로 검사한다. 만일 의존성 규칙내의 특정 값이 정해지지 않은 경우 디폴트 값은 'not selected'이다.

타겟 빌더는 위의 알고리즘을 사용하여 인터렉티브하게 현재 설정의 값을 체크한다. 초기 실험에서 2000개의 컴포넌트와 160개의 의존성 규칙을 갖는 시스템의 경우 약 0.1초내에 모든 의존성 체크가 가능했다.

5. GUI 와 컴포넌트의 보이기 제어(visibility control)

그림 4는 타겟빌더의 GUI를 보인다. GUI는 각 컴포넌트를 그룹별로 분류하여 트리모양으로 배치하고 각 컴포넌트가 가지는 값과 의존성, 도움말 등을 표시한다.



<그림 4> 타겟 빌더의 GUI

타겟 빌더 GUI의 특징은 컴포넌트가 보이는 방식에 대한 제어를 한다는 것이다. 컴포넌트의 개수가 매우 많을 경우 모든 컴포넌트를 다 보이기 보다는 현재의 설정과 관계없는 컴포넌트의 경우 아예 표시를 하지 않은 것이 바람직하다. 이를 위해 타겟 빌더는 컴포넌트에 대한 기술자인 QPD내에 "%%unless X suppress Y"속성을 통해 X라는 컴포넌트가 선택되지 않은 경우 Y라는 컴포넌트를 표시하지 않는다. 이를 통해 관련된 컴포넌트들이 점증적으로 보이게 하는 것이 가능하다.

6. 기존 임베디드 소프트웨어에의 적용과 분석

	Qplus-T	Qplus-P
Within Core OS services		114
Application <-> core OS services	4	14
Application <-> application	3	24
Etc		8
Total	7	160

<표 2> Qplus-P, Qplus-T의 의존성 분석

표2는 임베디드 운영체제인 Qlus-T와 Qplus-P를 타겟 빌더를 통해 컴포넌트 관리와 설정이 가능하도록 적용하는 과정중 찾은 의존성의 개수를 보인다. Qplus-P와 같은 중형 임베디드 운영체제의 경우 160개라는 많은 의존성을 발견할 수 있었다. ETRI 임베디드 소프트웨어 센터내의 약 20여명의 개발자들을 대상으로 한 실험에서 Qplus-T와 같은 경량 운영체제의 경우 비교적 적은 컴포넌트와 의존성을 가지므로 별다른 툴의 도움이 없이도 쉽게 설정이 가능했으나, Qplus-P 의 경우 툴의 도움없이 설정할 경우 올바른 설정을 하기위해서는 평균 30회 이상의 재설정 과정을 거쳤다. 타겟 빌더를 사용한 경우 그 회수는 약 2~3회로 줄어듬을 확인할 수 있었다.

7. 결론

임베디드 소프트웨어가 점점 모듈화되고 복잡해지면서 내부의 많은 컴포넌트들을 일일이 수작업으로 관리하고 설정하는 것은 매우 힘든 작업이 되었다. 타겟 빌더는 컴포넌트들을 체계적으로 관리하기 위한 방법을 제공하며, 각 컴포넌트간의 의존성을 기술할 수 있게 한다. 또한 의존성을 인터렉티브하게 체크함으로써 설정의 정확성을 보장하게 된다. 실제 개발과정에 적용한 결과 설정시에 겪게되는 오류의 회수를 크게 줄일 수 있음을 확인했다.

참조 문헌

[1] Lovic Gauthier, S. Yoo, A.A Jerraya, "Automatic Generation and Targeting of Application Specific Operating Systems and Embedded Software", 5th Intl. Workshop on Software and Compilers for Embedded systems, 2001.

[2] Calton Pu, T. et al. , "Optimistic Incremental Specialization: Streamlining a Commercial Operating System", 15th ACM Symposium on Operating Systems Principals, Dec. 1995.

[3] Kwangyong Lee, C. Lim, K. Kong, H. Kim, "A Design and Implementation of a Remote Debugging Environment for Embedded Internet Software", Proc. of Languages, Compilers, and Tools for Embedded Systems, June 2000.

[4] Tomas Axling, S.Hadridi, "A Tool for Developing Interactive Configuration Applications", Journal of Logic Programming, 1995.

[5] John McDermott, "R1: A rule-based configurer of computer systems", Artificial Intelligence, P39-88, 1982.

[6] Zhang, H, "SATO: An Efficient Propositional Prover", Proc. of International Conference on Automated Deduction (CADE-97), 1997.

[7] Bart Selman, H. Levesque, and D. Mitchell, "A new method for solving hard satisfiability problems", In Proceedings of the AAAI National Conference on Artificial Intelligence, pages 440--446. MIT press, 1992.

[8] Qplus-T, Thread Based RTOS Project, available at http://qplus.etri.re.kr/qplus-t

[9] Qplus-P/Target Builder Project, available at http://qplus.etri.re.kr/qplus-p

[10] Wind River Systems, Inc. VxWorks 5.4, available at http://www.wrs.co