실시간 특성을 지닌 평면 디스플레이 시스템 소프트웨어 재사용성 측정

Measuring Software Reusability for Flat Panel Display System with Real-Time Constraints

이종인, 전승훈
삼성전자 디지털 미디어 연구소 소프트웨어 플랫폼 랩

*Jong In Lee, Seung-Hun Jeon
Software Platform Lab Digital Media R&D Center Samsung Electronics
*chadlee@samsung.com

요 약

정보 가전 분야에 있어서 급속한 기술 발전으로 인해 하루가 다르게 새로운 기능이 추가됨에 따라 내장형 시스템 소프트웨어의 크기 및 복잡도 또한 함께 증가하고 있고 이를 개발하고 유지보수 하는데 있어서도 막대한 비용과 노력이 요구된다. 이를 해결하기 위한 방안으로 소프트웨어의 재사용성을 높이기 위한 노력이 이어지고 있다.

본 논문에서는 실시간 제약 특성을 지닌 평면 디스플레이 시스템 소프트웨어의 재사용성을 높이기 위하여 기존 내장형 시스템 소프트웨어에서 사용되던 순차적 구조에서 탈피하여 마이크로 커널 기반 태스크 구조를 제안하고 각각의 구조에 따른 소프트웨어의 재사용성을 측정하기 위한 기준(Metrics)과 그 측정 결과를 통하여 제안된 구조가 재사용에 적합함을 보이고자 한다.

1. 서 론

최근 정보가전 분야에 있어서 디지털 기술의 복합(Digital convergence)과 그에 따른 다양한 기능의 추가로 인해 내장형 시스템(Embedded System) 소프트웨어의 크기와 복잡성이 증가하고 있고 이에 따라 소프트웨어를 개발하고 유지보수 하는데 많은 시간과 인력이 요구되고 있다. 이에 따라 정보가전 제품의 내장형 소프트웨어에 대해서도 재사용에 대한 관심이 높아지고 실제 적용과 그 효과를 측정하고자 하는 노력이 이어지고 있다[1].

1980 년대 이래로 객체(Object) 기술은 종종 소프트웨어의 재사용 문제를 풀기 위한 핵심기술로 알려져 왔지만 객체 기술은 기업 적용 개발 프로젝트를 위해서는 너무 미미한 단위로 알려져 왔고 또 다른 대안으로 컴포넌트 기술이 소프트웨어 개발에 있어서 재사용성을 옹호해줄 새로운 기법으로 소개되어 왔다[2].

그러나 많은 제품의 소프트웨어가 이미 구현되어 있는 내장형 시스템의 경우제품의 사양이 크지 않은 폭에서 계속 변경되고 동일한 제품군에서 여러 가지 버전의 소프트웨어가 양산되는 특성과 시스템의 리소스를 고려해 볼 때 구현 소프트웨어데 대해 객체 기술이나 컴포넌트 기술을 바로 적용하는 데에는 무리가 있다. 특히 기존의 CE(Consumer Electronics) 제품군에 기존의 소프트웨어에 대해 큰 변경이나 많은 리소스의 추가가 요구되지 않는 상태에서 재사용을 위한 소프트웨어 구조 향상이 가능해져야 한다.

내장형 시스템 소프트웨어를 구현할 때, 기존의 주된 접근 방식은 순차적으로 시스템의 구동이 이루어지는 Super-Loop(혹은 Foreground/Background) 구조를 이용하는 것이다[3]. 이 구조는 초기에는 구현이 쉬우나 기능이 추가됨에 따라 그 복잡도가 현저하게 증가하게 된다.

상기의 제약 사항을 따르면서 기능의 추가 및 개발, 유지보수가 용이한 구조를 지니기 위해서는 마이크로 커널을 이용한 태스크 기반 구조가 대안으로 제시될 수 있다.

본 논문에서는 2 장에서 실시간 제약 특성이 있는 정보가전 제품인 평면 디스플레이 시스템에 대해 기존의 Super-Loop 구조의 소프트웨어에 대해 개략적으로 설명하고 마이크로 커널을 적용하여 태스크 기반으로 재구성(Restructuring)된 구조를 제안하고, 3 장에서는 각 구조로 구현된 소프트웨어에 대해 재사용성을 측정하기 위한 기준(metrics)에 대해 기존에 제시되었던 사항과 새로이 제시되는 사항을 나열하고, 4 장에서 그 측정 결과를 통하여 제안된 구조가 재사용에 적합함을 보이고자 한다.

2. Software Architecture

입력 신호를 필터링하고 디스플레이 패널(Display panel)에 전달해주기 위해 각 입력 신호를 적절한 입력 형태(input type)에 맞게 처리하는 디바이스(Devices)로 구성된 평면 디스플레이 시스템은 플래쉬 메모리(Flash memory)에 탑재되는 내장형 시스템 소프트웨어를 통해 스케일러(Scalar) 및 각종 디바이스가 구동되게 된다. 그림 1 은 평면 디스플레이 시스템 블록 다이어그램(Block diagram)의 한 예이다.

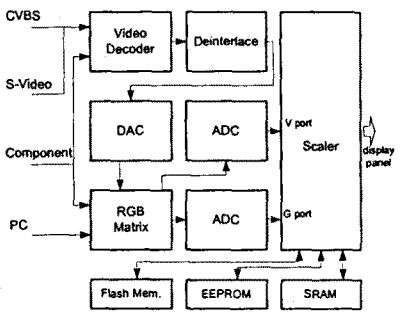

그림 1 **Example of Flat Panel System Block Diagram**

2.1. Super-Loop Structure

평면 디스플레이 시스템이 실제 구동 되는 과정은 디바이스의 초기화에 이은 상태 확인(Status check), 입력 신호 검출(Input signal detect), 디바이스 제어 등이 순차적으로 진행되고 사용자 입력 등이 인터럽트의 처리된다. 이를 하나의 무한 루프(Infinite loop)로 구현한 Super-Loop 구조는 그림 2 와 같이 표현되며 중요 operation 은 ISR(Interrupt service Routine)으로 처리되는 구조이다. 본래 ISR 은 interrupt device 로부터 data 와 status 를 획득하고 그에 대한 처리는 별도의 task 로 넘겨야 하는데 여기서는 ISR 내에서 operation 이 수행됨에 따라 더 많은 시간이 걸리게 된다.

이러한 구조에 어떠한 기능을 추가하게 되면 loop 의 execution time 이 증가하고 예측이 어려워지게 되어 소프트웨어의 유지보수 및 새로운 기능의 모듈 추가 등의 작업에 의해 효율적인 개발이 이루어지기 어렵다.

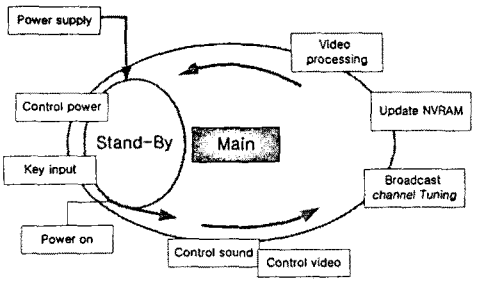

그림 2 **Super-Loop Structure**

2.2. Task-based Structure

마이크로 커널을 적용하기 위해서는 기능을 분리하여 task 와 match 하는 과정이 필요하다. 이를 위해서 실시간 시스템 디자인에 적합한 DARTS(Design Approach for Real-Time and Concurrent Systems) 방법을 적용하여 시스템 구조를 성립하고자 한다[4].

DARTS 는 각 단계를 거쳐 최종 태스크 구조도를 얻기 위한 것으로 산출물인 각 단계별 다이어그램이 서로 일관성이 유지되는지 확인함으로써 디자인에 오류가 있는지 검증할 수 있다. 여기서는 1)환경도(Context diagram), 2)상태천이도(State transition diagram), 3)데이터/컨트롤 흐름도(Data/Control flow diagram), 4)태스크 구조도(Task architecture), 5)세부 구조도(Structure chart)를 통해 설계 과정과 최종 태스크 기반의 구조에 대해 알아보도록 한다.

1) 환경도(Context Diagram)

그림 3 은 시스템의 환경도로서 시스템에 대해 내부 구성은 고려하지 않고 외부 인터페이스만을 고려한 것으로 모든 가능한 입력을 다 고려함으로써 디자인 과정에서 초기부터 일관성을 유지해 나갈 수 있다.

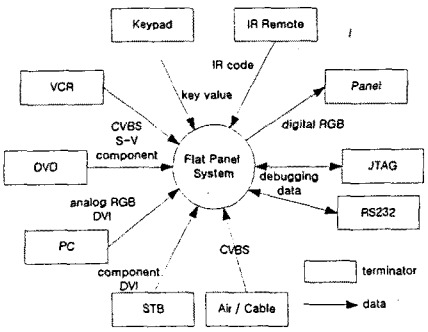

그림 3 **Context Diagram**

2) 상태천이도 (State Transition Diagram)

시스템의 상태를 정의하고 그 사이의 이동을 정의하면 시스템에 있어서 입출력에 대한 처리를 모듈화 하기 용이하게 된다. 그림 4 는 시스템의 상태천이도를 나타낸 것이다.

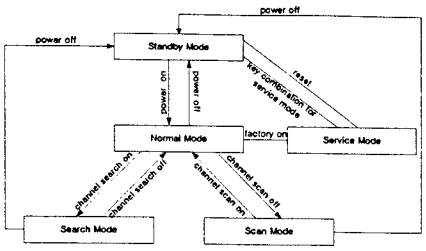

그림 4 State Transition Diagram

3) 데이터/컨트롤 흐름도(Data/Control Flow Diagram)

Task 기반의 구조를 지니기 위해 가장 중요한 단계로 각 기능을 분리/결합하는 과정이다. 방법을 외부의 입력과 내부의 데이터/컨트롤 입출력의 constancy 를 유지한 가운데 기능을 세분화 하고 다시 functional/temporal cohesion 을 통해 재결합하는 것이다. 그림 5 를 통해 세분화된 transform 과 이를 다시 결합한 task candidate 를 볼 수 있다.

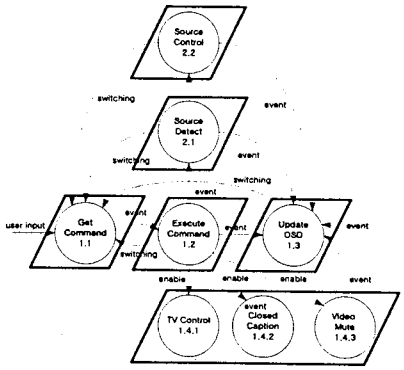

그림 5 Data/Control Flow Diagram

4) 태스크 구조도(Task Architecture)

이전 단계를 통해서 얻어진 내용을 통해 task 를 정의하고 task 간의 interface 를 정의하면 그림 6 과 같은 task architecture 를 얻을 수 있다.

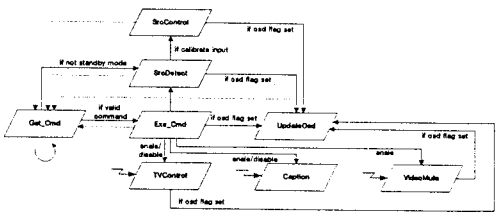

그림 6 Task Architecture

5) 구조도(Structure Chart)

태스크 구조도가 완성이 되면 각 태스크 별로 구조도를 작성하여 세부 모듈 및 모듈간 데이터/컨트롤의 흐름도를 작성하여 설계를 완성한다. 그림 7 은 생성된 task 중 하나의 구조도 예시이다.

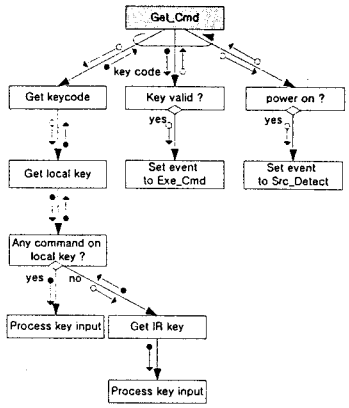

그림 7 Structure Chart Example

3. Metrics for Reusability

한 시스템의 내장형 소프트웨어가 재사용성이 높다 아니다를 단정지어 말하기는 어려울 것이다. 하지만 실제로 수 차례 재사용을 하고 그 평가를 내리기 전에 소프트웨어 자체만 가지고 그 결과를 예측하면 좋을 것이고 실제로 상당부분 예측이 가능하다[5].

3.1. Metrics for Maintainability

소프트웨어의 구조 변경 여부 이전에 소프트웨어 자체의 품질만을 가지고도 재사용성 여부를 예측할 수 있는 측정 기준들이 제시된 바 있다. 다음의 기준들은 Logiscope 를 통해 측정이 가능한 사항들로써 구조 변경에 직접 관련된 것은 아니지만 소프트웨어 자체의 analyzability, changeability 를 나타내주는 검증된 방법이기에 측정기준으로 삼았다.

metrics for Analyzability :

- **COMF**(Comments Frequency): 함수내의 프로그램 라인 수에 비례해서 설명문이 얼마나 차지하는가를 명시
- **lc_stat**(number of Statements): 함수의 시작과 끝 사이에서 수행되고 있는 모든 실행문들의 수
- **AVGS**(Average Size of Statements): 함수들 안의 수행문들 각각에 나타나는 operands 와 operations 의 평균 수
- **ct_vg**(Cyclomatic number): 최종 제어구조의 선형적으로 독립적인 경로의 수

metrics for Changeability :

- **dc_lvars**(Number of Local Variables): 함수 안에 선언되어 있는 모든 지역변수들의 수
- **ic_param**(Number of Function Parameter): 함수 내 파라미터의 수

- VOCF(Vocabulary Frequency): 각 함수 안에서 사용되어진 어휘들의 평균 사용 회수
- ct_bran(Number of Destructuring Statements): 함수 내 Goto, Break, Continue 문의 사용 회수

3.2. Architectural Metrics

3.1 장에서 나열한 metrics 은 일반적인 소프트웨어의 maintainability 를 나타내 주지만 제안된 소프트웨어와 같이 구조적 변경으로 인한 개선 여부는 나타낼 수 없다. Component 설계 기법에 따라 제안된 architectural metric 을 참조하여 task 기반 설계 구조에 맞게 변형하여 다음과 같은 metric 을 제안하여 본다[6].

metrics :

- NF3P (Number of 3rd Party): 외부 업체에서 제공된 library 함수 및 컴포넌트가 전체에서 차지하는 비율을 나타낸다.
- NDAT (Number of Device Access in a Task): 한 task 내에서 access / control 하는 device 의 수를 나타낸다.
- NIOL (Number of Interface Over the Layer): 상위 application layer 에서 하위 device 를 control 하는 layer 간 interface 의 수를 나타낸다.
- NFID (Number of Function that provides logical operation on Input Data): Data 의 논리적인 처리만으로 구현된 함수를 나타낸다.
- NSMD (Number of Shared Memory Data): 공유 데이터의 수를 나타낸다.
- NIT (Number of Interface Type): Data 전송, enable/disable 과 같은 함수간 interface type 을 나타낸다.
- NALF (Number of Application Layer Functions): Device 를 직접 control 하지 않는 상위 레벨 함수의 수를 나타낸다.
- NDRF (Number of Device driver Control): 디바이스를 직접 제어하는 함수의 수로서 총 함수 대비 차지하는 비율은 나타낸다.
- NFIA (Number of Functions of Internally Abstracted): 추상화 및 객체화된 함수의 수로서 총 함수대비 차지하는 비율을 나타낸다.
- NRSF (Number of Return Statements in Functions): 함수 내 return 문의 사용 횟수로 많은 return 문의 사용은 소프트웨어의 가독성 및 이식성을 낮춘다.
- NSMT (Number of Shared Memory Access in a Task): 한 태스크 내에서 control 하는 공유 데이터의 수를 나타낸다.

4. 결과

표 1 은 3 장에서 선택한 metrics 에 대한 각 소프트웨어의 측정값이다. 표의 첫째 열은 metric 항목을 나타낸 것이고 두번째 열은 Reusability 에 있어서의 영향을 나타낸 것이다. +는 수치가 클수록 reuse 에 좋은 항목을, -는 수치가 적을수록 reuse 에 좋은 항목을 나타낸다.

	Reuse	SLS	TBS
COMF	+	15.36 %	17.32 %
lc_stat	-	35.29 %	8.46 %
AVGS	-	9.24 %	5.91 %
ct_vg	-	19.33 %	10.60 %
dc_lvars	-	3.19 %	6.31 %
ic_param	-	0.17 %	3.62 %
VOCF	-	21.51 %	17.32%
ct_bran	-	2.86 %	3.91 %
NF3P	+	11.5 %	31.7 %
NDAT	-	17	3.6
NIOL	-	42	15
NFID	0	57	72
NSMD	-	111	132
NIT	-	4	6
NALF	+	37	42
NDRF	-	87 %	73 %
NFIA	+	12 %	19 %
NRSF	-	155 %	123 %
NSMT	-	597	475

SLS: Super-Loop Structure **TBS**: Task-Based Structure

표 1 Architectural Measure

5. 결론 및 향후연구

총 19 개의 측정 기준에 의한 구조적 분석 / 측정(Architectural Analysis/Measure)결과 13 개 항목에서 기존의 Super-Loop Structure 보다 마이크로 커널을 기반으로 한 Task-based Structure 가 재사용성에 우수한 구조임을 보이는 결과가 나왔다.

향후 좀더 재사용성이 우수한 소프트웨어 구조 및 이를 입증할 수 있는 측정 기준과 그 측정 방법에 대한 고찰이 이루어져야 할 것이다.

참고문헌

[1] Jeffrey S. Poulin, "Measuring Software Reusability", Proceedings., Third International Conference on, 1-4 Nov. 1994

[2] Szyperski C., Component Software: Beyond Object-Oriented Programming, Addison Wesley Longman, Reading, Mass., 1998

[3] Jean J. Labrosse, "Designing with Real-Time Kernels", Embedded Systems Conference Boston , No. 424, 2001

[4] Hassan Gomaa, Software design methods for concurrent and real-time systems, Addison-Wesley, 1993

[5] Dobrica, L., Niemela, E., "A Survey on Software Architecture Analysis Methods", Software Engineering, IEEE Transactions on, Vol. 28, Jul 2002

[6] Kalyanasundaram, S.; Ponnambalam, K.; Singh, A.; Stacey, B.J.; Munikoti, R., "Metrics for Software Architecture: A Case Study in the Telecommunication Domain", Electrical and Computer Engineering, 1998. IEEE Canadian Conference on, Vol. 2, 24-28 May, 1998