(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0126243
(43) 공개일자 2015년11월11일

(51) 국제특허분류(Int. Cl.)
G06F 15/76 (2006.01)
(21) 출원번호 10-2014-0053708
(22) 출원일자 2014년05월02일
심사청구일자 없음

(71) 출원인
김현수
대전 유성구 엑스포로 448, 403동 902호 (전민동, 엑스포아파트)
(72) 발명자
김현수
대전 유성구 엑스포로 448, 403동 902호 (전민동, 엑스포아파트)

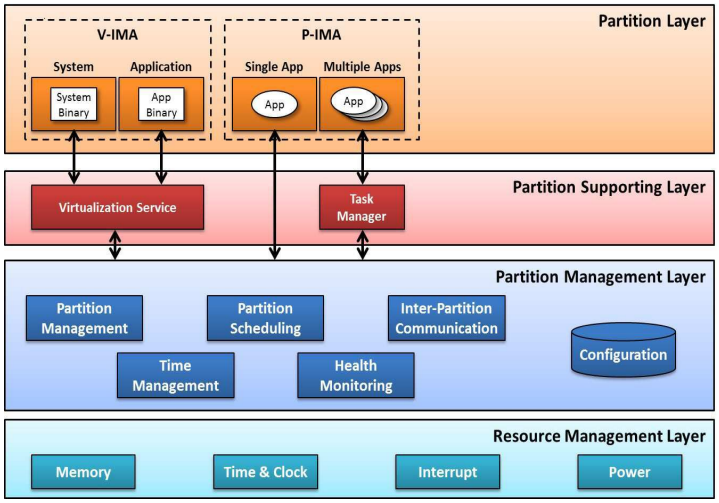
전체 청구항 수 : 총 1 항

(54) 발명의 명칭 **에이치-아이엠에이: 비행 소프트웨어의 이식성 향상을 위한 아이엠에이 기반의 혼합형 플랫폼 아키텍처**

(57) 요 약

비행 소프트웨어는 인공위성의 탑재 컴퓨터에서 사용되는 소프트웨어로, 실시간성과 고신뢰성이 요구된다. 이와 같은 요구사항으로 인해 비행 소프트웨어는 동작 환경에 대한 종속성을 갖게 된다. 이러한 문제는 새로운 시스템을 구축할 때마다 매번 다시 개발하여야 하는 상황을 초래한다. 따라서 비행 소프트웨어와 동작 환경 사이의 종속성을 제거할 필요가 있으며, 이는 비행 소프트웨어의 이식성 향상을 통해 달성할 수 있다. 본 발명에서는 이를 위해 IMA 아키텍처 기반의 플랫폼 아키텍처를 제안한다. 이 아키텍처는 이식성을 극대화하기 위해 두 가지의 IMA 아키텍처 실현 방안을 기반으로 구축된 혼합형 아키텍처이다. 본 발명에서 제안한 아키텍처를 통해서 비행 소프트웨어와 동직 환경 사이의 종속성을 제거할 수 있다.

대 표 도



명 세 서

청구범위

청구항 1

.

발명의 설명

기 술 분 야

[0001] 비행 소프트웨어에 대한 이식성을 제공하는 플랫폼을 개발하기 위한 소프트웨어 아키텍처 수립에 관한 것이다.

배 경 기 술

[0002] IMA(Integrated Modular Avionics) 아키텍처는 항공 분야에서 제안된 아키텍처로, 모듈화 시스템(Modular System)의 집합으로 구성된다. IMA 아키텍처에서는 모듈화 시스템을 파티션(Partition)이라 부르며, 비행 소프트웨어는 파티션 내부에 포함된다. 파티션들은 IMA 플랫폼을 통해 관리되며, IMA 플랫폼은 파티션에게 공유 자원을 제공한다. 여기서 제공되는 공유 자원은 크게 시간적 자원(예: CPU 등)과 공간적 자원(예: 메모리, 통신채널 등)으로 나눌 수 있다. IMA 아키텍처를 통해 얻을 수 있는 이점은 다음과 같다. IMA 플랫폼을 통해서 파티션과 하드웨어를 분리할 수 있어 비행 소프트웨어와 하드웨어와의 의존성을 낮출 수 있다. 또한 파티션은 시공간적으로 완벽하게 분리되어 구성되기 때문에 파티션 혹은 비행 소프트웨어에서 발생된 오류가 시스템으로 전파되는 것을 막을 수 있다. 파티션 사이의 분리는 비행 소프트웨어의 병렬적인 개발을 가능하도록 하여 비행 소프트웨어를 개발하는데 소요되는 시간을 줄여준다. 마지막으로 공간적 분리를 통해 비행 소프트웨어의 실시간성을 보장할 수 있다. 따라서 IMA 아키텍처를 적용한다면 비행 소프트웨어가 갖는 요구사항을 모두 충족시킬 수 있다.

[0003] IMA 아키텍처를 실현하는 방법에 따라서 가상화 기반의 IMA(Virtualization based IMA, 이후 V-IMA), 파티셔닝 기반의 IMA(Partitioning based IMA, 이후 P-IMA), 혼합형 IMA(Hybrid IMA, 이후 H-IMA)로 나눌 수 있다. V-IMA는 가상화 기술을 바탕으로 개발된 IMA 시스템을 의미한다. IMA 아키텍처의 파티션을 가상화 기술의 가상머신(Virtual Machine)으로 구현한다. P-IMA는 ARINC 653 표준을 따르는 IMA 시스템을 의미한다. 가상화 기술과 달리 추상 인터페이스를 통해서 이식성을 제공한다. H-IMA는 본 발명에서 제시하는 실현 방법으로 V-IMA와 P-IMA의 혼합형 IMA 시스템을 의미한다. V-IMA와 P-IMA는 모두 이식성을 제공하나, 특정 조건에 한하여 이식성을 제공한다. H-IMA는 V-IMA와 P-IMA가 갖는 이식성을 모두 제공한다.

발명의 내용

해결하려는 과제

[0004] 비행 소프트웨어(Flight Software)는 인공위성의 탑재 컴퓨터에서 사용되는 소프트웨어로, 우주 공간이라는 특수한 환경에서 동작한다. 비행 소프트웨어는 우주 공간이 갖는 접근성의 문제로 인해 유지보수가 거의 불가능하다. 따라서 탑재된 비행 소프트웨어는 내고장성(Fault Tolerance)과 고장 확산(Fault Propagation) 방지 기능, 고신뢰성이 요구된다. 뿐만 아니라 적재적소에 주어진 임무를 수행하여야 하기 때문에 실시간성 또한 요구되며, 탑재 컴퓨터라는 한정된 자원을 기반으로 동작하여야 한다. 이와 같은 요구사항을 만족하기 위해서 지금까지 비행 소프트웨어는 고신뢰성을 보장하는 실시간 운영체제에 종속적으로 개발되었다. 비행 소프트웨어가 특정 동작 환경(예: 실시간 운영체제, 프로세서 등)에 종속성을 가짐으로 인해 비행 소프트웨어의 재사용성이 크게 떨어진다는 문제점이 발생하게 된다. 비행 소프트웨어를 개발하고 검증하기 위한 노력과 비용 등을 고려할 때 비행 소프트웨어의 재사용성은 중요한 이슈이며, 이를 위해 비행 소프트웨어는 동작 환경에 독립성을 갖도록 개발되어야 한다. 고신뢰성, 실시간성, 한정적 자원 기반의 동작과 같은 문제는 IMA 아키텍처를 통해 대부분 해결된다. 그러나 동작 환경에 대한 독립성은 IMA 아키텍처를 통해 완벽히 해결되지 않는다. IMA 아키텍처에서는 동작 환

경에 대한 독립성을 해결하기 위해서 이식성을 제공하지만, 이식성은 실현하는 방법에 따라서 그 특성이 바뀌어 상황에 따라 동작 환경에 대한 독립성을 제공하지 못한다.

과제의 해결 수단

[0005] 본 발명에서는 상기의 문제를 해결하기 위해서 다양한 이식성을 제공하는 IMA 아키텍처를 제안한다. 이를 위해서 다양한 이식성에 대해 정의하고, 각 이식성을 담을 수 있도록 스케치와 블랜드 과정을 통해 혼합형 IMA 아키텍처를 수립한다.

발명의 효과

[0006] 본 발명에서는 IMA 아키텍처의 이식성이 실현되는 기술에 따라 서로 다른 특성을 띄는 것에 주목하고, 이를 모두 제공할 수 있는 H-IMA를 제시하였다. H-IMA에서는 각 솔루션에서 제공하는 이식성을 분석하고 이를 네 가지 유형의 파티션으로 분류하여 제공한다. 이와 같이 다양한 파티션을 제공함으로써, 다양한 상황에 따라 다양한 유형의 소프트웨어를 탑재할 수 있을 뿐만 아니라 시스템을 구축하는데 소요되는 시간을 줄일 수 있다.

도면의 간단한 설명

[0007] 도 1은 P-IMA과 V-IMA이 갖는 이식성에 대한 분류

도 2는 스케치 과정을 통해 도출된 H-IMA 아키텍처

도 3은 블랜드 과정을 통해 도출된 파티션의 종류

도 4는 수립된 H-IMA 아키텍처

발명을 실시하기 위한 구체적인 내용

[0008] 1. IMA 아키텍처의 이식성

[0009] IMA 아키텍처의 이식성을 살펴보기 위해서는 P-IMA와 V-IMA가 제공하는 이식성을 살펴보아야 한다. IMA 아키텍처는 단순히 개념에 불과하여 구체적인 요소가 존재하지 않기 때문이다. P-IMA와 V-IMA가 제공하는 이식성은 [도 1]과 같이 정리할 수 있다.

[0010] P-IMA는 파티셔닝을 바탕으로 IMA 아키텍처를 달성한다. 파티셔닝은 ARINC 653 표준에 명시된 개념으로, 유사한 임계성(Criticality)을 갖는 비행 소프트웨어를 하나로 엮어 오류의 확산을 막는 방법이다. 이때, 비행 소프트웨어를 엮는 단위를 파티션이라 한다. 파티셔닝을 달성하기 위해서는 시간(예: CPU)과 공간(예: 메모리)에 대해 분리가 이루어져야 한다. P-IMA는 별도의 인터페이스를 통해 이식성을 제공한다. 이와 같은 인터페이스를 일반적으로 추상 인터페이스라 부르며, 이는 이식성을 제공하는 방법 중 하나이다. 추상 인터페이스를 통해 개발된 소프트웨어는 특정 운영체제의 시스템 콜에 종속되지 않아 운영체제에 대해 이식성을 갖는다. 또한 특정 운영체제의 시스템 콜을 가져다 쓰지 않기 때문에, 소프트웨어의 소스코드를 어떤 운영체제에서든 그대로 재사용할 수 있다. 이는 곧 소프트웨어의 수정이 용이함을 의미하며, 이와 같은 특징은 파티션의 구성을 자유롭게 한다. 어떤 파티션에도 종속되지 않고 시스템에 탑재될 수 있기 때문에 P-IMA 기반의 시스템에서는 소프트웨어 자체가 이식성을 갖는다. 따라서 P-IMA는 [도 1]의 내용처럼 운영체제로부터의 이식성, 소스코드 기반의 이식성, 비행 소프트웨어 단위의 이식성을 제공한다.

[0011] V-IMA는 가상화 기술을 바탕으로 IMA 아키텍처를 달성한다. 가상화는 특정 시스템이나 프로세스에게 동형이질성(Isomorphism)을 제공하는 것을 말한다. 동형이질성이란, 본래 동작 환경과는 다른 구성의 시스템에서도 본래 동작 환경과 동일한 동작을 하는 것을 말한다. 이때, 동형이질성을 제공받는 시스템을 게스트(Guest)라 하고 동형이질성을 제공하는 시스템을 호스트(Host)라 한다. 본래 가상화는 이미 개발된 프로그램을 다른 환경에서 동작시키기 위해 등장하였으나, 현재는 그 외에도 시스템 간의 분리를 통해 효율적 서버 운영이나 보안 강화의 목적으로 사용되고 있다. 가상화가 제공하는 다양한 기능은 VMM(Virtual Machine Monitor) 혹은 하이퍼바이저(Hypervisor)라고 부르는 요소를 통해서 제공된다. VMM은 게스트를 관리하고 게스트가 호스트의 하드웨어 자원

을 사용할 수 있도록 도와준다. V-IMA는 이진코드 변환 기술을 통해 이식성을 제공한다. 이진코드 변환 기술은 게스트의 이진코드를 호스트에서 동작할 수 있는 이진코드로 변환하는 기술을 의미한다. 따라서 V-IMA는 어떤 프로세서 위에서 동작하든지 관계없이 비행 소프트웨어를 동작시킬 수 있다. 즉, 프로세서에 대한 이식성을 제공한다. 그러나 이진코드로부터 실행 가능한 이진코드를 얻어내기 때문에 수정이 용이하지 않다는 단점이 존재한다. 이러한 이유로 V-IMA에 탑재되는 시스템은 수정이 불가능하여 이전 시스템의 형태 그대로 탑재하여야 한다. 따라서 V-IMA는 [도 1]의 내용처럼 프로세서로부터의 이식성, 이진코드 기반의 이식성, 시스템 단위의 이식성을 제공한다.

[0012] 2. H-IMA 아키텍처

[0013] 본 발명에서는 V-IMA와 P-IMA의 이식성을 모두 제공하기 위해 두 실현 방안의 혼합형 아키텍처인 H-IMA를 제안한다. V-IMA와 P-IMA는 동일한 목적을 달성하고 있으나, 엄연히 다른 구성을 갖는 시스템이기 때문에 하나의 아키텍처로 혼합하는 과정은 특정한 절차와 방법에 입각하여 수행하여야 한다. H-IMA를 수립하기 위해서 다음의 절차를 따른다. (1)P-IMA와 V-IMA 사이의 공통점과 차이점을 파악한다. 다음으로 파악한 (2)공통점을 바탕으로 H-IMA의 기본 구조를 정의한 뒤, (3)기본 구조에 P-IMA와 V-IMA의 차이점을 반영한다. 이와 같은 절차를 따르는 이유는 각 시스템의 장점을 최대한 유지하기 위함이다. 일반적으로 시스템의 장점은 시스템의 잘 갖추어진 구조로부터 나오기 때문에, 시스템의 장점을 살리기 위해서는 시스템의 구조를 최대한 준수하여야 한다. 이번 절에서는 주어진 절차에 따라 V-IMA와 P-IMA의 공통점과 차이점을 파악하고, 파악된 공통점과 차이점으로부터 H-IMA의 수립 방안을 제안한다.

[0014] 2.1 V-IMA와 P-IMA의 공통점과 차이점

[0015] V-IMA와 P-IMA는 기본적으로 IMA 아키텍처의 실현 방안이기 때문에, IMA 아키텍처가 갖는 특징 자체가 두 실현 방안의 공통점이 된다. IMA 아키텍처의 특징은 (1)자원과 소프트웨어의 분리를 위한 계층을 제공한다는 것과 소프트웨어의 임계도(Criticality)를 바탕으로 분리하여 탑재한다는 점, 즉 (2)파티션을 제공하는 것이다. 자원과 소프트웨어를 분리하는 것은 일반적으로 운영체제에서 수행하는 것이다. 이러한 이유로 IMA 실현 방안을 운영체제 서비스를 제공하는 방법에 따라 V-IMA와 P-IMA로 나눈다. 여기까지만 보면 일반적인 운영체제와 차이가 없어 보이지만, 파티션을 통해 소프트웨어가 탑재된다는 것이 일반적인 운영체제와의 차이점이다. 따라서 IMA 아키텍처에서는 일반적인 운영체제에서 제공하는 기능 외에도 파티션을 관리하기 위한 기능을 추가적으로 제공하여야 한다. 다시 말해 IMA 아키텍처는 자원과 파티션을 관리하는 계층이 요구되며, 이것이 곧 V-IMA와 P-IMA의 공통점이 된다. V-IMA와 P-IMA의 차이점은 제공되는 이식성이다. V-IMA는 이진코드로 구성된 소프트웨어에 대해서 이식성을 제공하며, P-IMA는 추상 인터페이스를 기반으로 구현된 소프트웨어에 대해서 이식성을 제공한다. 다시 말해, V-IMA와 P-IMA는 이식성을 제공할 수 있는 소프트웨어의 유형이 다르다는 것이다. 이 차이로 인해 V-IMA와 P-IMA는 서로 다른 이식성의 달성 방법과 장점을 갖는다.

[0016] 2.1.1 V-IMA의 장점

[0017] V-IMA는 이식에 대한 전제 조건이 없다. 여기서 전제 조건이란 이식성을 제공받기 위해 소프트웨어가 충당하여야 하는 요소를 의미한다. P-IMA는 소프트웨어가 추상 인터페이스에서 제공하는 API를 기반으로 구현되지 않으면 소프트웨어에게 이식성을 제공할 수 없다. 즉, P-IMA는 추상 인터페이스를 사용하여야 한다는 것이 전제 조건이다. V-IMA는 자체에서 소프트웨어의 이식성을 위한 요소를 포함하고 있기 때문에 이식성을 위한 요소를 소프트웨어가 갖추고 있을 필요가 없다. 이와 같은 이유로 V-IMA는 실제 적용이 빠르다는 장점을 갖는다.

[0018] 2.1.2 P-IMA의 장점

[0019] P-IMA는 소프트웨어의 활용도를 높일 수 있다. V-IMA는 이진코드로 구성된 소프트웨어를 사용하는 것을 기본으로 한다. 이러한 이유로 이진코드로 구성된 소프트웨어는 그 구조를 바꾸는 것이 거의 불가능하다. 예를 들어 V-IMA에서 사용되는 소프트웨어는 보통 운영체제와 함께 탑재된다. 이렇게 탑재된 소프트웨어는 함께 탑재된 운영체제를 다른 것으로 교체하는 것이 불가능하며, 더 나아가 소프트웨어 자체를 수정하는 것 역시 불가능하다. 하지만 P-IMA는 추상 인터페이스를 사용하기 때문에 어떠한 운영체제에도 탑재될 수 있다. 또한 추상 인터페이

스를 기반으로 구현되기 때문에 소스코드 자체를 재사용할 수 있는 장점이 있다. 즉, 한 번 개발된 소프트웨어가 그 자체로만 사용되는 것이 아니라 새로운 소프트웨어를 개발할 때 또 다른 자원으로 활용될 수 있다. 따라서 P-IMA는 운영체제로부터의 독립성, 그리고 소스코드의 재사용성을 통해서 높은 활용도를 갖는 소프트웨어를 제공한다.

[0020] 2.2 H-IMA 아키텍처 수립

[0021] 앞서 파악된 공통점과 차이점을 통해 H-IMA 아키텍처를 수립한다. 본 발명에서는 공통점을 바탕으로 기본 구조를 정의하는 과정을 스케치(Sketch) 과정이라 하고, 기본 구조에 차이점을 반영하는 과정을 블랜드(Blend) 과정이라고 한다.

[0022] 2.2.1 H-IMA 아키텍처 스케치

[0023] P-IMA와 V-IMA의 공통점을 통해 하드웨어(자원)와 파티션 사이에서 파티션과 공유 자원을 관리하는 요소가 필수적임을 파악하였다. 따라서 H-IMA 아키텍처의 기본 구조를 [도 2]와 같이 표현할 수 있다. H-IMA의 기본 구조는 단순히 하드웨어와 파티션 사이에 운영체제가 존재하는 형태를 갖는다. 다만, 기존의 운영체제와 달리 파티션을 관리하는 계층, 즉 파티션 관리자(Partition Manager)가 존재하며, 파티션과 공유 자원을 관리하는 계층은 서로 분리된다. 운영체제를 두 개의 관리 계층으로 나눈 것은 관심의 분리(Separation of Concerns)를 위함이다. 파티션 관리자가 수행하여야 하는 기능은 파티션과 밀접한 관계를 가지며, 공유 자원 관리자(Shared Resource Manager)는 하드웨어와 밀접한 관계를 갖는다. 만일 파티션 관리자와 공유 자원 관리자가 수행하여야 하는 작업이 명세되고, 이를 기반으로 플랫폼을 구축한다고 가정하자. 파티션 관리자는 어떠한 시스템에 탑재되더라도 변경되지 않는다. 파티션의 구성은 파티션 관리자에게 영향을 주기 어렵고, 하드웨어의 접근은 공유 자원 관리자를 통해 수행하기 때문이다. 그러나 공유 자원 관리자는 시스템의 변경, 특히 하드웨어의 변경에 민감할 수밖에 없다. 자신이 수행하는 기능은 변하지 않더라도 그 구현은 바뀔 수밖에 없다. 만일 두 요소가 분리되지 않는다면, 하드웨어의 변경에 따라 모든 것이 변경되어야 한다. 이와 같은 이유로 운영체제를 파티션 관리자와 공유 자원 관리자로 분리한다.

[0024] 2.2.2 H-IMA 아키텍처 블랜드

[0025] P-IMA와 V-IMA의 차이점은 제공되는 이식성에 있다. IMA 아키텍처에서 이식성의 주체는 파티션이므로, 결국 파티션 자체가 두 시스템 사이의 차이점을 만들어 낸다고 볼 수 있다. 따라서 P-IMA와 V-IMA의 차이점을 구체화하기 위해서는 H-IMA 아키텍처에서 사용되는 파티션의 유형을 파악하는 것이 중요하다. H-IMA 아키텍처에서 사용되는 파티션은 간단하게 P-IMA 기반의 파티션과 V-IMA 기반의 파티션으로 나뉜다. 추가적으로 파티션에 탑재되는 소프트웨어의 개수를 통해 파티션을 분류할 수 있다. 이 기준은 단순하게 생각하면 비효율적일 수 있지만, 소프트웨어의 개수는 관리 측면에서 차이가 있다. 만일 파티션 내부에 소프트웨어가 하나만 존재한다면, 내부의 소프트웨어 자체가 하나의 파티션과 동일하기 때문에 태스크 관리에 대해 고려하지 않아도 된다. 반면, 파티션 내에 소프트웨어가 두 개 이상이면 태스크 관리는 반드시 필요하다. 즉, 태스크 관리 측면에서 단일 소프트웨어로 구성된 파티션과 다수의 소프트웨어로 구성된 파티션으로 분류할 수 있다. 이와 같은 기준으로 [도 3]와 같이 파티션을 분류할 수 있다. 제시된 기준을 통해 분류된 파티션은 총 네 가지이며, 각 파티션에 대한 설명은 다음과 같다.

[0026] - (유형1) : 유형1 파티션은 V-IMA 기반의 파티션으로, 두 개 이상의 이진 소프트웨어로 구성되는 파티션이다. 비행 소프트웨어의 집합만으로 구성되기 보다는 게스트 운영체제와 함께 구성된다. 즉, 하나의 시스템에 대한 이미지 파일이 탑재되는 경우이다. 유형1 파티션에는 게스트 운영체제가 함께 탑재되기 때문에 비행 소프트웨어에 대한 스케줄링 요소를 추가적으로 제공하지 않아도 된다.

[0027] - (유형2) : 유형2 파티션은 P-IMA 기반의 파티션으로, 두 개 이상의 원본 소프트웨어로 구성되는 파티션이다. 유형1과 달리 게스트 운영체제가 포함되지 않으며, 오로지 비행 소프트웨어의 집합으로만 구성된다. 유형 2 파티션은 추상 인터페이스와 함께 다수의 비행 소프트웨어를 스케줄링 할 수 있는 기능이 별도로 요구된다.

[0028] - (유형3) : 유형3 파티션은 V-IMA 기반의 파티션으로, 하나의 이진 소프트웨어로 구성되는 파티션이다. 게스트 운영체제를 포함하지 않으며, 비행 소프트웨어 단독으로 탑재된다. 유형3 파티션은 단일 소프트웨어로 구성되어 비행 소프트웨어에 대한 스케줄링 요소를 고려하지 않아도 된다.

[0029] - (유형4) : 유형4 파티션은 P-IMA 기반의 파티션으로, 하나의 원본 소프트웨어로 구성되는 파티션이다. 게스트 운영체제를 포함하지 않으며, 비행 소프트웨어 단독으로 탑재된다. 단일 소프트웨어로 구성되어 비행 소프트웨어에 대한 스케줄링 요소를 고려하지 않아도 된다.

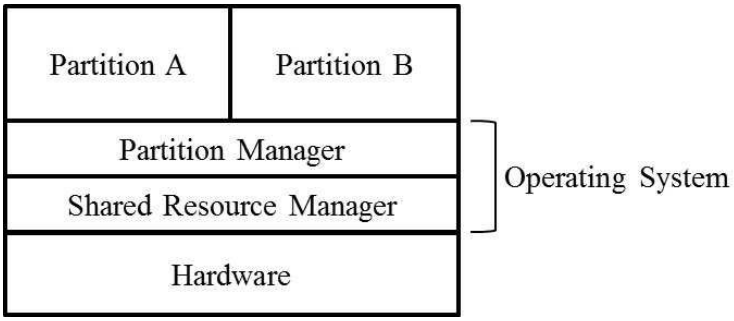
[0030] 분류된 파티션을 H-IMA 아키텍처의 기본 구조에 반영함으로써, H-IMA 아키텍처를 완성할 수 있다. 그러나 분류된 파티션이 동작할 수 있는 시스템의 구성은 각 파티션마다 다르다. 예를 들면, 유형1 파티션은 전가상화 기법을 기반으로 구성된 시스템에서 동작하며 유형2 파티션은 추상 인터페이스를 제공하는 시스템에서 동작할 수 있다. 즉, 분류된 파티션을 반영하기 위해서는 별도의 요소가 제공되어야 한다. 그러나 이러한 요소는 특정 파티션에 종속적으로 구성될 수밖에 없기 때문에 [도 2]의 운영체제에서 직접 이를 제공하면 H-IMA 아키텍처 스케치 과정에서 의도하였던 관심의 분리를 위배할 뿐만 아니라 이식성을 해친다. 따라서 파티션 지원 계층(Partition Supporting Layer)을 통해 파티션의 동작 환경을 제공한다. 즉 V-IMA 기반의 파티션인 유형1과 유형3을 위해서는 이진코드 변환 요소가 반드시 지원되어야 하고, P-IMA 기반의 파티션인 유형2와 유형4를 위해서는 추상 인터페이스가 제공되어야 한다. 이런 내용을 H-IMA 아키텍처의 기본 구조에 반영하면 [도 4]와 같은 H-IMA 아키텍처를 완성할 수 있다.

도면

도면1

P-IMA	V-IMA
Portability from Operating System	Portability from Processor
Portability based on Source Code	Portability based on Binary Code
Portability for Flight Software	Portability for System

도면2



도면3

		Software Type	
		Binary	Source
# of FSW	Multiple	Type1	Type2
	Single	Type3	Type4

도면4

