FDT 인터페이스 사양

1. 서론

본 장에서는 프로젝트의 수행 동기, 목적 및 FDT(Field Device Tool) 개념에 대한 솔루션을 설명합니다.

1.1 과제

공정 자동화에서 제어 시스템은 종종 10,000개 이상의 바이너리 및 아날로그 입력/출력 신호로 이루어진다. 필드버스를 사용하는 경우 이러한 신호는 버스를 통해 전송되는데, 이를 위해 필드 장치는 버스에 직접 연결하거나 원격 I/O를 통해 측정할 수 있다. 다양한 장치 제조 업체의 100가지 이상의 필드 장치 유형이 자주 사용됩니다.

장치는 각 작업에 대해 구성하고 파라미터화 한다. 필드버스 커플러(Coupler) 및 버스 통신을 구성하는 경우 장치별 특성 및 설정을 고려해야 하며 제어 시스템에 장치를 인식시켜야 합니다. 장치가 제공하는 입력 및 출력 신호뿐만 아니라 함수 블록을 생성하고 제어 시스템의 함수 계획에 통합시켜야 합니다.

1.2 최신 설비

지금은 하나의 제어 시스템 프로젝트에 다수의 상이한 장치 유형과 공급 업체를 사용하게 됨으로써 작업구성이 어렵고 시간도 많이 소비하게 됩니다. 서로 다른 도구의 사용법을 완전히 익혀야 하며 이러한 도구 간에 데이터

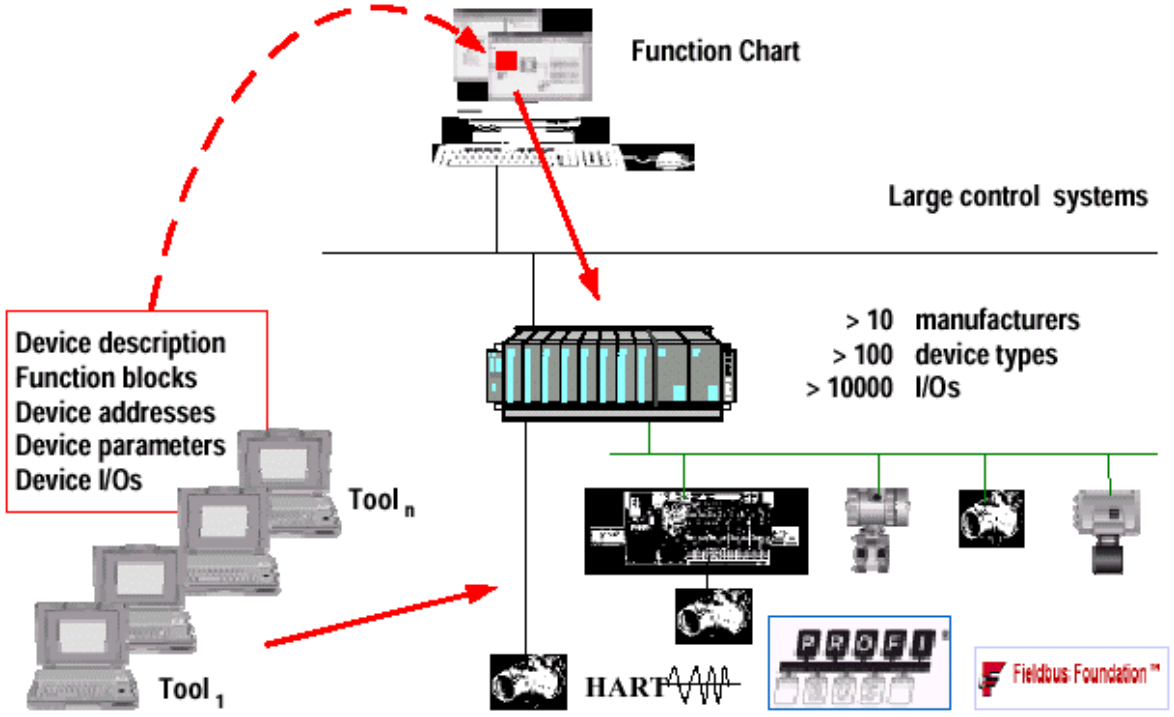

그림 1 : Different tools and multiple data input have determined field device integration to data

를 교환해야 하는데 데이터 교환은 표준화되어 있지 않습니다. 따라서 종종 데이터를 변환해야 하기 때문에 세부적인 전문가적 지식이 필요합니다. 결국 데이터, 문서 및 구성의 일관성은 강도 높은 시스템 테스트를 통해서만 보장할 수 있습니다.

제어 시스템의 서비스 및 진단 작업을 위한 중앙 작업 공간은 필드버스 장치의 기능을 완전히 처리할 수 없을 뿐만 아니라 서로 다른 장치별 도구를 시스템의 소프트웨어 도구로 통합할 수도 없습니다. 보통, 장치별 도구는 필드버스 라인이나 필드버스 장치에 직접 연결하는 것만 가능합니다.

1.3 목적

공정 제어 기술의 연속성과 작동 신뢰성을 유지하기 위해서는 필드버스 장치를 공정 자동화의 하부 구성요소로 완전히 통합할 필요가 있습니다. 공정 제어 시스템은 시스템 및 필드버스를 통해 중앙 엔지니어링 또는 오퍼레이터 작업 공간에서 개별 필드 장치까지의 통신 경로를 제공해야 합니다.
여기서 주요 목적은 다음과 같습니다.

- 모든 필드 장치에 대한 직접 액세스를 통해 계획, 진단 및 서비스를 수행하기 위한 중앙 작업공간
- 공정 제어 시스템, 필드버스 및 장치에 대한 일관성 있고 통합된 구성 및 문서
- 공정 제어 시스템 및 필드 장치에 대한 공용 데이터의 구성
- 중앙 데이터 관리 및 데이터 보안
- 서로 다른 유형의 장치를 공정 제어 시스템으로 빠르고 용이하게 통합

공정 제어 기술의 엔지니어링 시스템으로 필드 장치 기술을 통합하는 경우 이를 일반적 검증을 거친 얼마 안 되는 구성, 서비스 및 진단 기능에만 국한시켜서는 안됩니다. 즉, 이는 PROFIBUS 정의를 필드 장치에 대한 기본 정의로 통합한다는 것을 의미한다. 대신, 공급된 상이한 장치 유형의 특수 기능 및 개별 장치 특성, 특징 모두를 통합하는 결과를 얻어야 한다. 장치 제조 업체가 제공한 계획 및 서비스 도구를 엔지니어링 시스템에 장치별 소프트웨어 구성요소로 통합시켜야 한다. 장치 제조 업체는 자사가 만든 장치에 대한 구성, 서비스 및 진단 기능을 자체적으로 정의하며 또한 공정 제어 시스템의 엔지니어링 환경에서 사용되는 장치의 외관을 디자인한다.

그리고 나서 이러한 구성요소를 모든 제어 기술 공급 업체의 엔지니어링 시스템으로 통합할 수 있어야 한다. 이렇게 되면 장치 제조 업체는 지능형 필드 장치에 대한 모든 구성, 서비스 및 진단 기능을 갖는 하나의 표준화된 소프트웨어 구성요소를 제공하기만 하면 되기 때문에 비용을 줄일 수 있다. 하나의 장치 유형에 대해 여러 번 개발하고 유지 보수해야 하는 프로젝트별 또는 제어 시스템별 잦은 개조는 표준화된 구성요소 기술을 통해 제거해야 한다.

제어 시스템 제조 업체는 모든 필드버스 장치의 통합을 위해 정의된 인터페이스를 한 번만 구현해야 하며, 제조 업체별 및/또는 장치별 구현 및 그 유지보수 요소를 제거해야 한다.

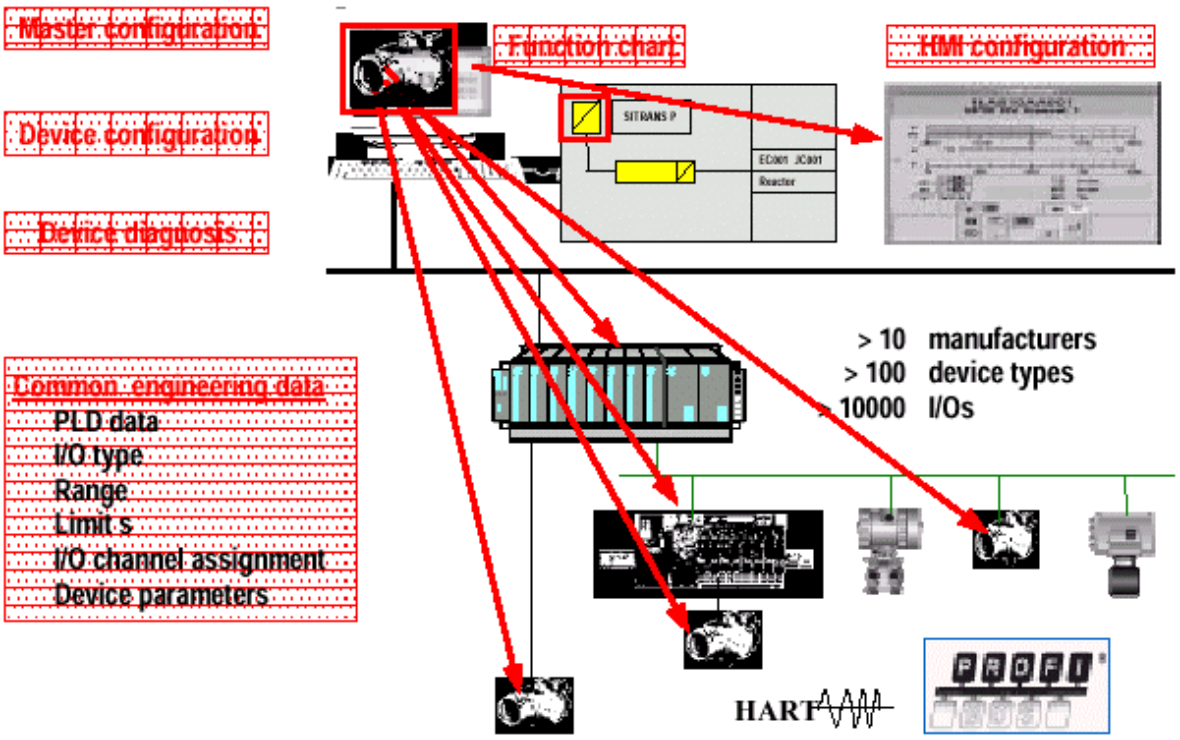

그림 2 : The potential of the field bus technology cannot be used until the field bus has been homogeneously integrated into the engineering systems

1.4 기술 지향

프린터용 드라이버와 같이 소프트웨어 구성요소를 하드웨어 구성요소와 조합하여 사용함으로써 "플러그 앤 플레이" 원리를 필드버스 기술에도 도입하고 있다.

더욱이, Microsoft Windows가 사실상의 업계 표준으로 확립됨에 따라 운영 체제를 선택할 때 대안이 거의 없다. Microsoft가 도입한 ActiveX 기술을 사용하면 데이터뿐만 아니라 함수를 포함하는 인터페이스를 정의할 수 있다. 이러한 가능성은 OPC(OLE for Process Control) 정의와 연계하여 이미 성공적으로 사용된 바 있다.

FDT내에서는 ActiveX의 인터페이스 기술만이 필드 장치의 엔지니어링 구성요소로 지정된다. 하는 경우 ActiveX 기술은 구성요소의 자동 통합을 수행하며 장치의 소프트웨어 구성요소와 엔지니어링 시스템 간의 상호 작용을 관리한다.

기반을 이루는 객체 모델은 향후 기능성을 위해 쉽게 확장할 수 있는 클라이언트-서버 구조를 기반으로 하며, 장치 소프트웨어 구성요소와 엔지니어링 시스템 간의 데이터 교환은 XML(eXtensible Markup Language)을 통해 수행된다. XML은 또한 FDT의 향후 버전에서 교환될 정보의 내용을 쉽게 확장할 수 있도록 한다. FDT의 클라이언트 서버 구조를 구현하는 첫 단계는 Microsoft COM에 기반한다.

1.5 솔루션 개념

FDT 개념은 장치 공급 업체가 제공하는 장치별 소프트웨어 구성요소와 제어 시스템 제조 업체의 엔지니어링 도구 간의 인터페이스를 정의한다. 장치별 소프트웨어 구성요소를 DTM(Device Type Manager)이라 하며, FDT 개념은 기타 다른 응용에서 필드 장치의 처리를 위해 사용할 수 있다. 그러나, 현재 FDT 버전의 중점은 필드버스 기반 제어 시스템의 엔지니어링, 시운전, 진단 및 문서에 있다. 자산 관리의 의미에서 감사 추적을 수행하기 위해 기본 기능성이 정의된다.

장치 유형 관리자는 장치와 함께 장치 제조 업체가 제공한다. DTM의 특징은 다음과 같다.

- 일반적으로 독립형 도구가 아님
- FDT-Spec에 의해 정의된 ActiveX 인터페이스
- 장치의 모든 규칙이 인식됨
- 모든 사용자 대화상자 포함
- 사용자 인터페이스(도움말 시스템을 포함한 다중 언어)
- 파라미터 유효성 점검(다른 장치별 파라미터에도 의존)
- 종속 파라미터의 자동 생성
- 고품질 필드 장치에 대한 복잡한 보정, 일치 및 설정 절차의 처리 순서 정의
- 필드 버스로부터 파라미터의 읽기 및 쓰기
- 장치에 대해 개별 정의된 진단 기능
- 통신 구성을 위한 유형별 데이터의 제공
- 예를 들어, 함수 계획에서 사용하기 위한 장치/인스턴스별 데이터의 제공
- 장치/인스턴스별 문서
- 기타 어떤 장치에도 직접 연결하지 않음
- 엔지니어링 환경에 대한 정보 없음
- 하나 이상의 장치 지원

여기서 열거한 기능의 양(장치 제조 업체가 서비스를 위해 최적화)은 장치의 기능적 역량에 따라 다르다. 하나의 DTM은 적어도 하나의 필드 장치를 취급한다. 그러나, DTM은 예를 들어, 제조 업체의 프로파일 또는 전체 팔레트(Pallete)에 기반하여 장치 군(예: 압력 변환 장치)도 취급할 수 있다. 통신 및 데이터 관리(제어 시스템의 다양한 버스 시스템을 통해)는 엔지니어링 도구의 인터페이스를 통해 처리한다. 전체적인 시스템 계획 또는 플랜트 관리 구조 내에서 DTM은 항상 해당 엔지니어링 도구로 통합시켜야 한다. 병렬 독립형 작동은 예를 들어, 독립형 도구에서 DTM으로 이동하는 특수한 경우 구현할 수 있다. 데이터 일관성 때문에 독립형 솔루션 및 동일한 장치에 액세스하는 시스템의 엔지니어링 도구에서 실행되는 DTM에는 병렬 작동이 지원되지 않는다. 독립형 작동은 주로 플랜트의 작업장에서 테스트 목적으로 사용할 수 있다.

DTM은 장치별 작업을 통해 연관된 구성요소를 시운전하고 통신 메커니즘을 제공하며 장치 인스턴스를 관리하는 엔지니어링 도구 또는 기타 모든 어플리케이션에 대한 구성요소로 설치된다. 이후 이러한 어플리케이션을 '프레임-어플리케이션'이라고 한다.

Frame-Application에는 다음과 같은 요구 조건이 적용된다.

- 개별 장치에 대한 지식이 필요 없음
- 모든 장치 인스턴스를 관리하고 인스턴스 데이터를 저장

- 장치 통신 및 연결 생성(tool routing)
- 여러 사용자 및 서버/클라이언트 작동 가능
- 데이터 변형(Versioning) 처리

1.6 DTM으로의 Migration

현재 상황에서 비추어볼 때, 단순한 I/O 센서에서 복잡한 모듈형 원격-I/O 또는 드라이버에 이르는 매우 다양한 필드 장치가 존재한다. 복잡성에 따라 장치를 4가지 범주로 세분할 수 있다.

A : 주기적으로만 통신하는 단순 장치. 예를들러 Light barrier가 있다.
B : 고정 하드웨어 및 소프트웨어를 갖는 조절형 장치로 압력 변환 장치가 있다.
C : 모듈형 하드웨어를 갖지만 고정 소프트웨어 블록이 없는 조절형 장치로 원격 I/O가 있다.
D : 모듈형 하드웨어 및 프로그램형 소프트웨어 블록을 갖는 조절형 장치로 복합 서버 드라이브가 있다.

이러한 상이한 장치에는 장치가 제공하는 기능성에 따라 자체 구성 도구 및 장치의 능력에 대한 서로 다른 종류 설명과 함께 제공된다. FDT를 사용하면 비록 장치 제조 업체가 DTM 개발에서 일차적인 작업을 알지 못한다 하더라도 DTM을 통해 이러한 모든 장치를 일관되게 프레임-어플리케이션 내로 통합시킬 수 있다.
예를 들어, 범주 A 및 B의 단순 장치는 통신 기능에 대한 정보를 담고 있는 기존 파일 또는 장치 설명으로 충분히 설명할 수 있다. 이러한 장치 설명을 해석할 수 있는 '일반 DTM'을 개발하고 포함된 정보 및 기능을 시스템 및 그 사용자가 사용할 수 있도록 할 수 있다. 특정 장치 설명에 대한 일반 DTM을 개

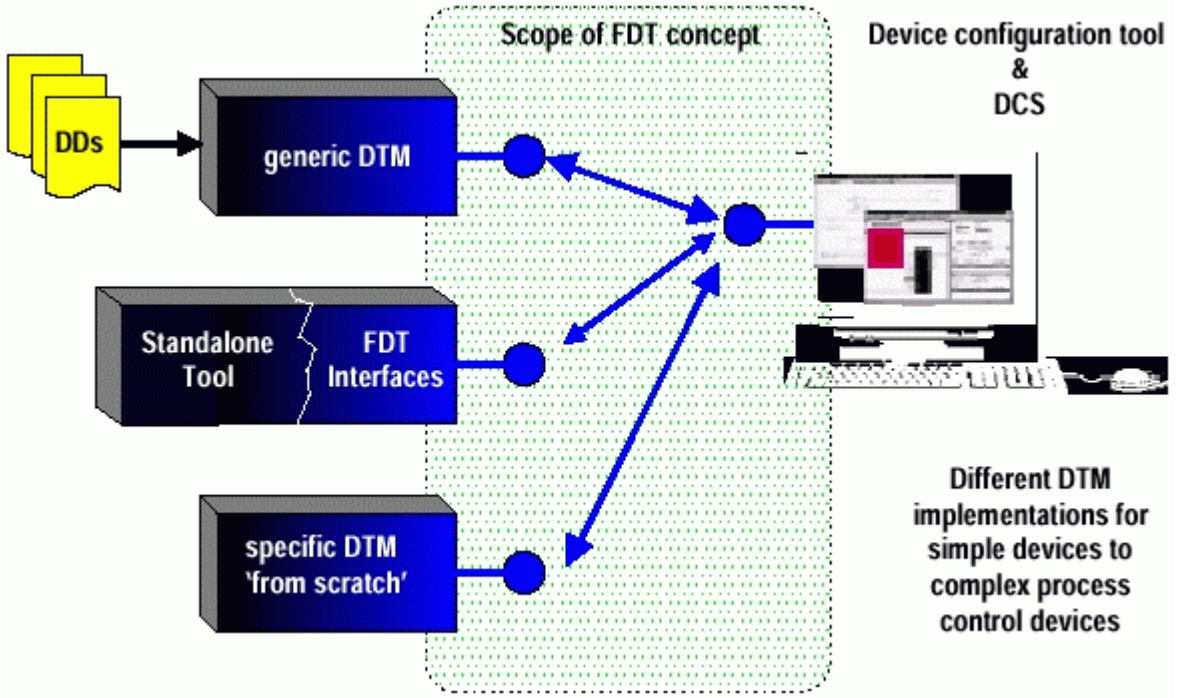

그림 3 : DTP - Implementations

발하면 이 설명을 지원하는 모든 장치를 동일한 DTM을 사용하여 통합할 수 있다.
한편, 범주 C 및 D에 속하는 장치의 경우 이미 기존 독립형 도구가 존재할 수 있다. FDT는 개방성을 제공하여 이러한 도구에 FDT 인터페이스를 구비 시키고 기존 독립형 장치로부터 DTM을 구축한다. 이런 식으로, 장치 제조 업체의 투자 비용을 보존할 수 있다.

장기적으로 보면 각 장치 제조 업체는 기존 또는 새로운 장치에 대해 다음과 같은 옵션을 자유롭게 선택할 수 있다.

- 장치 설명에 기반한 일반 솔루션을 유지한다.
- 독립형 도구로 유지된 DTM을 병렬로 공급한다.
- 장치 취급을 위한 새로운 기능을 도입하기 위해 처음부터 새로운 DTM을 구축한다.

1.7 성능 범위

FDT 개념은 모든 엔지니어링 작업 및 연관 도구에 대한 솔루션의 제공을 요구하지 않는다. 따라서, “전기 배선 계획, 기계 계획 등”의 엔지니어링 주제 또는 “유지보수, 최적화, 보관 등”과 같은 플랜트 관리 주제는 현재 FDT의 수행 범위에 속하지 않는다. 이 중 일부가 FDT 향후 버전에 포함될 수는 있다.

본 FDT 개념 버전의 범위에는 필드 장치의 엔지니어링 및 시운전이 포함된다. 문서 및 감사 추적 지원에 대한 기능도 검토된다. 예제 및 사용 사례는 주로 액터(Actor), 센서 및 원격 I/O의 경우 제공된다. 물론, FDT 인터페이스는 드라이브, 분석기, 레코더 등과 같은 장치에서도 마찬가지로 사용할 수 있다. FDT의 본 버전은 PROFIBUS 및 HART 프로토콜에 주요 중점을 두며 대응하는 XML schema이 포함된다. FDT의 확장성 있는 구조로 인해, 새로운 schema를 추가하여 Foundation Fieldbus 또는 Modbus 등과 같은 기타 프로토콜로 범위를 확장하는 것이 용이하다.
FDT 개념을 통해 장치 제조 업체는 엔지니어링 시스템에 자사 장치에 특정한 기능성을 제공할 수 있다. 따라서, FDT의 주요 능력은 장치의 기능성과 그에 따른 어플리케이션에 따라 다르다.

1.8 대상

이 규정은 FDT를 따르는 프레임-어플리케이션 및 DTM 개발자에게 참조 자료를 제공하기 위한 것이다. 여기서는 귀하께서 Microsoft ActiveX 기술 및 필드 장치의 필요성 또는 공정 제어 산업의 필요성을 잘 알고 있다고 가정한다.

이 규정은 선택 언어로 Microsoft ActiveX를 지원하는 프레임-어플리케이션의 개발을 돕기 위한 것이다. 따라서, 각 구성요소의 개발자는 그에 해당하는 특정 구성요소에 필요한 기술에 정통해야 한다.

기억할 것은 FDT가 Microsoft COM을 사용하여 구현한 클라이언트/서버 구조의 첫 단계라는 것이다.

2. FDT의 기초

이번 장에서는 FDT 모델을 소개하고 필드 장

치를 통합하기 위해 특정하게 요구되는 조건들을 다룬다.

2.1 FDT의 개요

FDT 규정은 FDT 객체 및 프레임-어플리케이션에 의해 구현된 DTM(Device Type Manager)이라는 장치별 어플리케이션과 그 인터페이스를 설명한다. DTM은 완전히 통일된 프레임-어플리케이션에 연결하거나, 또는 하나 이상의 공급 업체가 제공한 서로 다른 구성요소로 이루어진 프레임-애플리케이션에 연결할 수 있다. 역으로, 프레임-어플리케이션은 서로 다른 공급 업체가 제공하는 장치별 DTM의 통합을 지원할 수 있다.

DTM은 장치 정보 및 기능성에 대해 서버로 작용한다. 서로 다른 공급 업체가 DTM Server를 제공할 것이다. 공급 업체가 제공하는 코드는 프레임-어플리케이션이 액세스할 수 있는 데이터 및 장치 기능성을 결정한다.

수 있다. 특정한 기능성을 포함하는 장치 자체도 존재한다. 장치의 기능성에 액세스하기 위해서는 통신 채널을 통해 장치와 환경 간에 데이터를 교환해야 한다. 이러한 채널은 원격 I/O와 또는 필드버스를 통해 통신하고 송신기로 측정되는 서로 다른 프로세스 값과 동일할 수 있다.

이 모델에 의하면 DTM 서버 객체는 장치의 기능성을 담고 있고, FDT 채널 객체는 각각 장치의 채널 또는 I/O 데이터에 대한 정보를 담고 있다. 더욱이, FDT 채널 객체는 DTM과 프레임-어플리케이션 간의 데이터 교환에 대한 메커니즘을 제공한다.

각 채널 내에서 DTM은 데이터 유형, 범위, 경보 등과 같이 채널에 관한 정보를 부여하는 하나 이상의 필드버스별 파라미터를 정의할 수 있다.

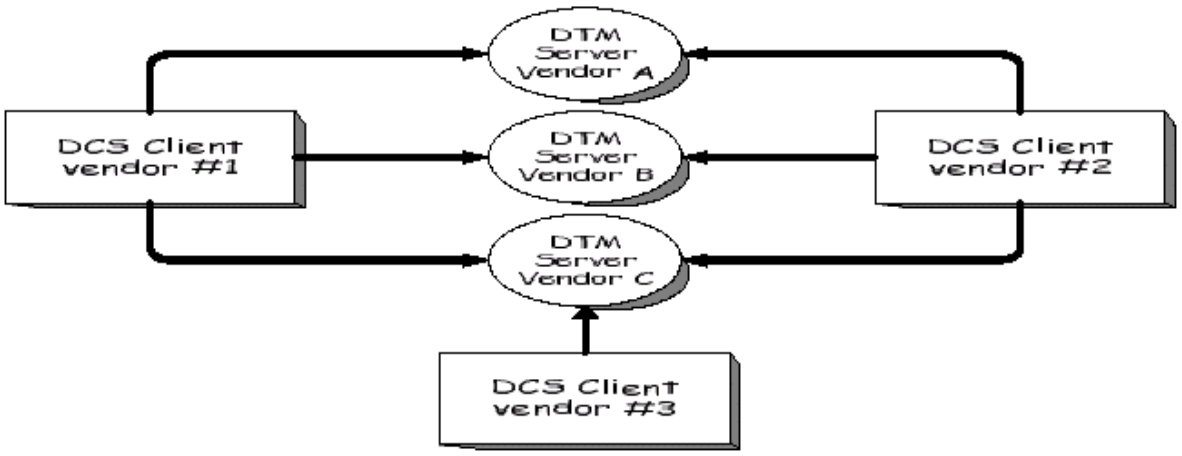

그림 2-1 : FDT Client/Server Relationship

높은 수준에서, DTM은 몇 개의 객체 즉, 서버 객체 및 채널 객체로 이루어진다. 이 모델을 통해 외부 어플리케이션은 장치를 살펴볼

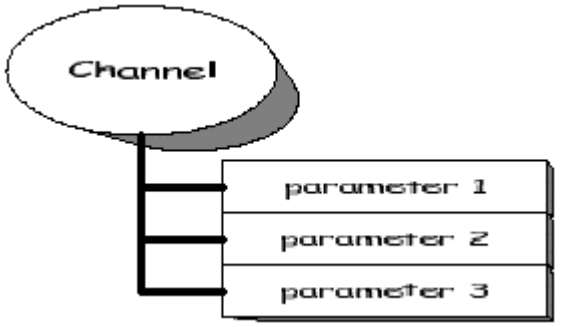

그림 2-2 : Channel/Parameter Relationship

이러한 필드버스별 파라미터에는 프레임-어플리케이션이 장치의 I/O 데이터를 통합하기 위해 필요로 하는 모든 정보가 포함된다. 일반적으로, I/O 데이터를 설명하기 위해 파라미터를 전달하는 채널은 데이터 소스가 아니며 단지 이를 표현하는 것이다. 이러한 파라미터는 주소가 참조하는 데이터의 실제 소스가 아니라 데이터의 주소를 단순히 지정하는 것으로 생각해야 한다.

더욱이, 채널은 보조 통신 프로토콜을 사용하여 채널을 통해 통신하기 위한 기능을 가질 수 있다. 이러한 채널을 '게이트웨이 채널'이라 하며 'Nested Communication(연계된 통신)'에 관한 장에서 자세히 설명할 것이다.

FDT 객체 모델에 관한 상세한 정보는 3.2 장에서 찾을 수 있다.

2.2 FDT의 사용 부분

FDT의 설계가 장치의 기능성 제어와 제어 시스템 부분을 구성하기 위한 데이터 액세스에 우선적으로 맞추어 졌지만 FDT 인터페이스는 애플리케이션 내의 많은 부분에 사용할 수 있다. 최하위 수준에서, FDT 인터페이스는 장치로부터 원시 데이터를 얻은 다음 버스 마스터를 구성하기 위해 SCADA 또는 DCS로 보낼 수 있다. 최상위 수준에서는, 프레임-애플리케이션이 DTM을 통해 장치별 진단 애플리케이션을 시작할 수 있다. 구조와 설계는 확장성 있는 DTM을 구축하고 통합할 수 있도록 하며 여기서 기능성은 장치의 능력에 달려 있다.

DTM의 확장성은 이후에 설명한다.

2.3 일반적인 FDT 구조 및 구성요소

FDT는 장치별 어플리케이션과 FDT 프레임-어플리케이션 간의 상호 작용을 촉진시키기 위한 인터페이스 규정이다. 이를 그림 2-3에 나타내었다.

런타임 환경으로서 DTM은 FDT 컨테이너라는 것을 필요로 한다. 이 컨테이너는 FDT 규정에 정의된 대로 인터페이스를 제공해야 한다. 통상적으로, FDT-컨테이너는 DTM을 사용하는 클라이언트 어플리케이션, 장치/DTM 데이터의 지속적 저장을 위한 어떤 종류의 데이터베이스, 그리고 필드 장치에 대한 통신 연결로 구성된다. FDT-컨테이너는 제어 시스템을 위한 엔지니어링 도구(대개 완전한 제어 시스템) 또는 장치 구성을 위한 독립형 도구와 같은 어플리케이션으로 나타낼 수 있다. 이러한 어플리케이션을 '프레임-어플리케이션'이라 한다. 본 문서 전체에 걸쳐서 'FDT-컨테이너' 및 '프레임-어플리케이션'이라는 용어는 동의어로 사용한다.

클라이언트 어플리케이션은 구성, 관찰, 채널 할당 등과 같은 특수한 측면에 중점을 두고 서버인 DTM이 제공하는 기능성을 사용하는

단일 어플리케이션으로 본다

FDT 규정은 COM 인터페이스를 지정(인터페이스의 대상)하며 이러한 인터페이스의 구현(구현 방법)은 다루지 않는다. 이 규정은 인터페이스가 이를 사용하는 클라이언트 어플리케이션에 제공해야 하는 동작을 지정한다. FDT 규정은 DTM의 구현을 지정하지 않을 뿐만 아니라 프레임-어플리케이션의 구현도 지정하지 않는다.

2.4.1 DTM(Device Type Manager)

플랜트에 설치된 필드버스 장치에는 다양한 유형이 있다. 따라서, 이렇게 상이한 장치를 다루기 위해서는 하나 또는 몇 개의 DTM(Device Type Manager)이 필요하다. DTM은 필드버스 장치 제조 업체가 제공한다. DTM은 시스템에 설치하고 새로운 필드버스 장치에 대해 새로운 DTM을 설치하여 동적으로 이를 확장할 수 있다.

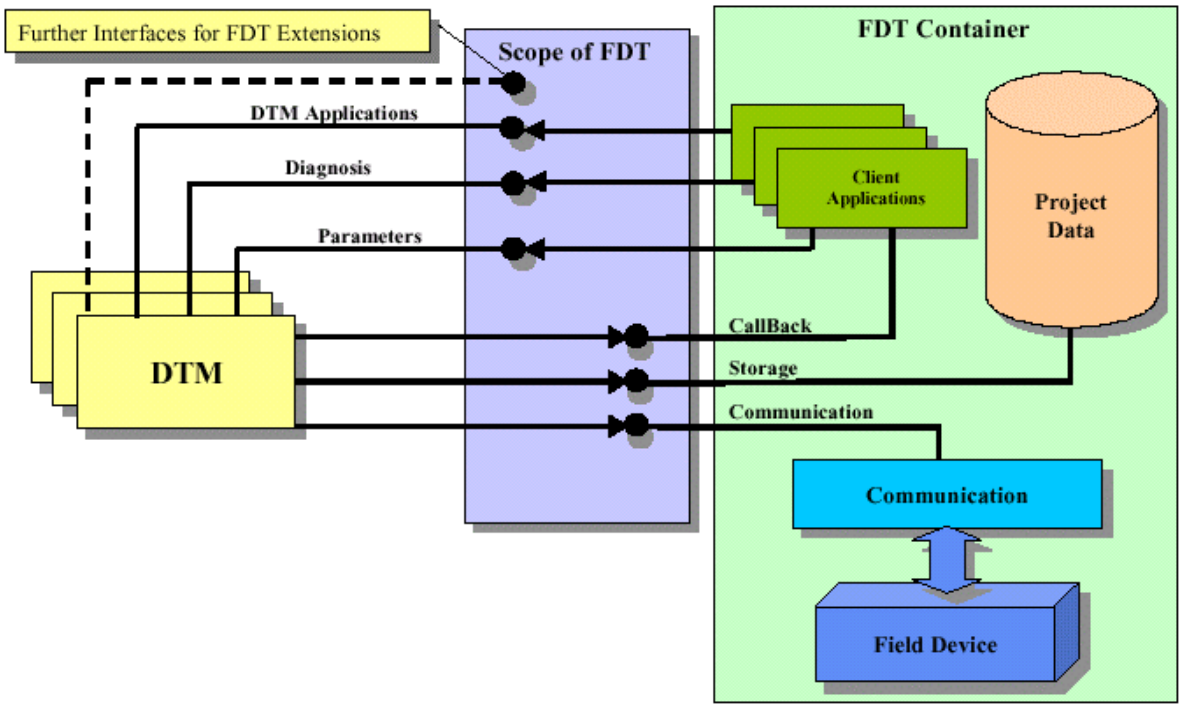

그림 2-3 : FDT Interface

구조에 대한 설명과 이러한 구조에 가장 적합한 것으로 보이는 인터페이스에 대한 설명이 포함된다. 모든 COM 구현과 마찬가지로, FDT의 구조는 클라이언트/서버 모델로서 여기서 DTM은 프레임-어플리케이션이 관리하는 서버 구성요소이다.

2.4 객체 및 인터페이스의 개요

DTM이 한 가지 유형의 장치를 처리하는 지 또는 일단의 장치 유형을 처리하는 지는 DTM의 소프트웨어 설계에 달려 있다. 심지어는 일단의 대상 장치 유형에 대해 하나의 매우 강력한 DTM을 구현할 수 있다.

보통, 하나의 DTM이 하나의 장치 유형을 처리하며 그 장치의 특정 파라미터, 동작 및 제한에 관한 모든 정보를 인식한다.

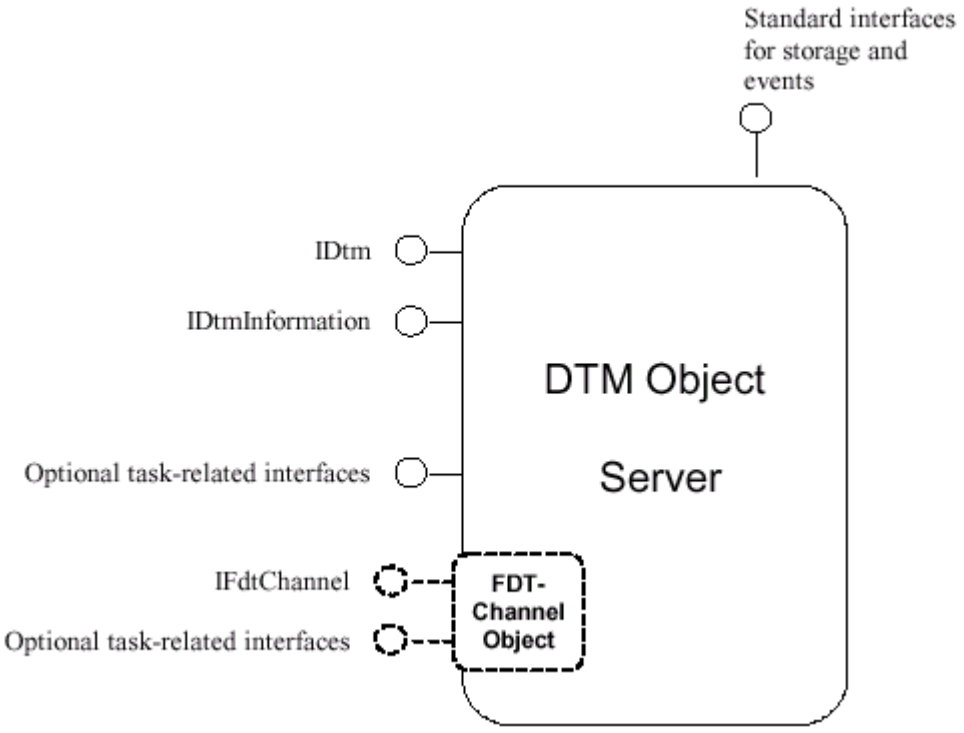

그림 2-4 : DTM Interface

인터페이스 IDtm 및 IDtm Information은 각 DTM이 구현해야 한다. 이러한 인터페이스는 DTM을 제어하기 위한 기본 기능과 정보를 제공한다. 프레임-어플리케이션의 관점에서, 장치 기능성과의 상호 작용을 위한 모든 작업 관련 인터페이스는 이러한 인터페이스를 통해 사용 가능하다. 어떤 작업 관련 인터페이스가 제공되는 지는 DTM과 그에 따른 장치의 기능에 달려 있다. 각 인터페이스는 해당 장에서 자세히 설명한다.

장치의 I/O 연결 또는 프레임-어플리케이션과의 데이터 교환을 위해 액세스할 수 있는 프로세스 값을 나타내기 위해, DTM은 FDT 채널 객체를 구현할 수 있다. 적어도 이 객체는 채널을 설명하는 모든 파라미터에 대한 액세스를 부여하는 IFdtChannel 인터페이스를 구현한다.

PROFIBUS 게이트웨이에 대한 HART처럼 장치가 게이트웨이 기능성을 제공하는 경우, FDT 채널 객체는 이 채널을 통한 통신을 위해 추가 인터페이스를 구현해야 한다. FDT 채널 객체의 각 인터페이스는 해당하는 각 장에서 자세히 설명한다.

2.4.2 FDT 프레임-어플리케이션

FDT 프레임-어플리케이션은 데이터를 관리하고 장치와 통신하는 완전한 기능성을 제공하고 DTM을 내장하고있다. 따라서, 프레임-어플리케이션이 엔지니어링 툴인지 독립형 도구인지 또는 웹 페이지인지는 환경과 작업에 따라 다르다. DTM은 이것이 작동하는 환경에 독립적이어야 한다. 따라서, 환경에 특정한 모든 작업은 프레임-어플리케이션이 처리해야 한다.

대부분의 경우, 엔지니어링 도구는 Data Management System(RDBMS, OODBMS)에 기반한다. 이러한 이유로 인해 DTM은 트랜잭션 전략을 구현하지 않는다. 트랜잭션이 사용될 지의 여부는 각 프레임-어플리케이션의 데이터 관리에 따라 다르다. 데이터 관리의 모든 측면은 프레임-어플리케이션 내에 캡슐화되며 이는 어떤 DTM과도 관련을 갖지 않는다.

표준 저장 인터페이스는 프레임-어플리케이션의 저장 메커니즘을 캡슐화한다. 이는 단순한 파일 시스템일 수도 있고 엔지니어링 도구의 경우, DCS 시스템 제조 업체가 제공하는 데이터베이스일 수도 있다.

이 인터페이스를 통해 DTM이 저장하는 데이터는 DTM에 특정하며 다른 어플리케이션이 사용할 수 없다. 어떤 데이터를 저장할 것인가는 DTM에 달려있지만 각 DTM은 이러한 데이터를 로드함으로써 저장된 각 장치 인스턴스의 표시를 보장할 수 있어야 한다.

복잡한 플랜트 환경에는 마찬가지로 프로세스 장치를 연결하기 위한 복잡한 통신 네트워크가 존재한다. DTM에는 시스템 네트워크의 토폴로지에 대한 어떤 정보도 필요하지 않아야 한다. 따라서 장치 액세스를 위한 라우팅 수행은 프레임-어플리케이션이 결정한다. 프레임-어플리케이션은 각 경우에 피어 투 피어 연결(물리 또는 논리)을 제공해야 한다. 따라서 여러 사용자의 장치 액세스를 처리하는 것은 프레임-어플리케이션이 결정한다.

통신 기능을 나타내기 위해 프레임-어플리케이션은 FDT 채널 객체를 구현할 수 있다. 프레임-어플리케이션의 채널은 개별 FDT에서 개별 프레임-어플리케이션 통신으로의 게이트웨이를 나타낸다. 적어도 IFdtCommunication 인터페이스를 프레임-어

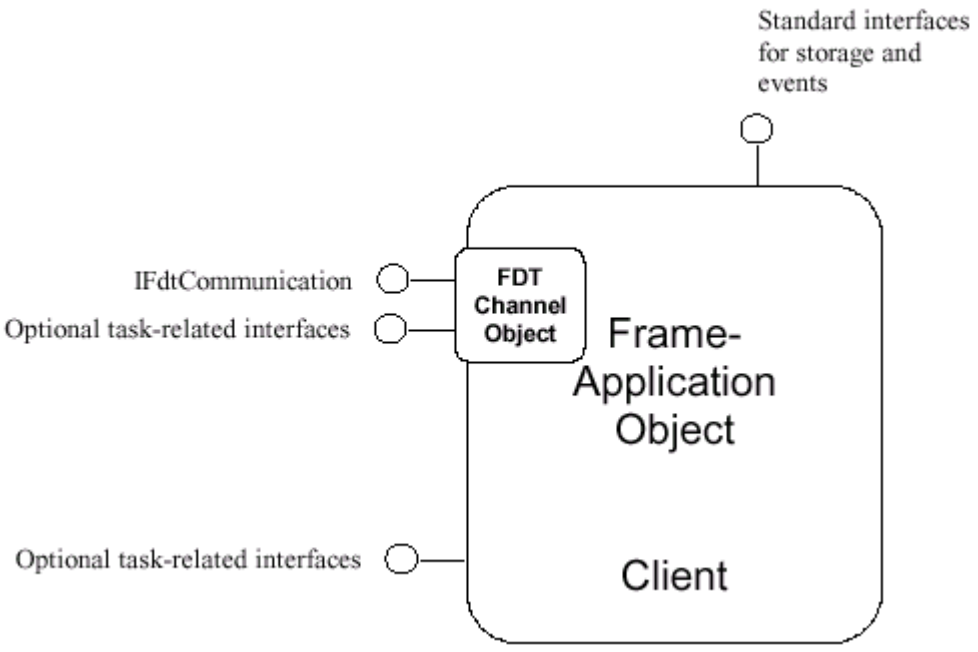

그림 2-5 : Flame Application Interfaces

플리케이션 채널에 대해 구현해야 한다. 프레임-어플리케이션 쪽에서 통신 채널은 PC I/O 보드일 수도 있고 처리 장치 및 소유 버스 시스템을 갖는 엔지니어링 Topology일 수도 있다.

IFdtCommunication은 DTM이 그 필드버스 장치에 액세스하기 위한 통신 기능성을 항상 제공한다. 물리 필드버스에 속한 모든 동작은 이 인터페이스를 사용하여 수행되어야 한다.

각 DTM은 FDT 채널이 해당 통신 프로토콜을 지원하는 경우 각 FDT-(게이트웨이) 채널과 통신할 수 있다. 구성을 통하여 DTM를 특정 FDT 채널과 연관시킨다. 구성을 위한 Topology 정보는 IFdtTopology 인터페이스를 통해 프레임-어플리케이션의 데이터베이스에 저장된다.

FDT-(게이트웨이) 채널은 동일 장치에 대한 몇 가지 연결을 처리할 수 있어야 하고 서로 다른 장치에 대한 연결을 취급할 수 있어야 한다. 구성요소는 장치에 대한 연결이 피어 투 피어 연결 형식으로 구성되도록 보장한다. 비동기 통신, 특히 호출 ID의 관리를 위해서는 독특한 피어 투 피어 연결이 필요하다. DTM은 오직 피어 투 피어 연결에 대해서만 사용 호출 ID가 모든 통신 프로세스에 대해 독특하도록 보장해야 한다(예: 구성과 관찰을 병렬로).

일반적으로, 동일한 하나의 DTM은 해당 FDT-(게이트웨이) 채널을 사용하여 서로 다른 필드버스 인터페이스에 부착된 동일 유형의 장치와 통신할 수 있다.

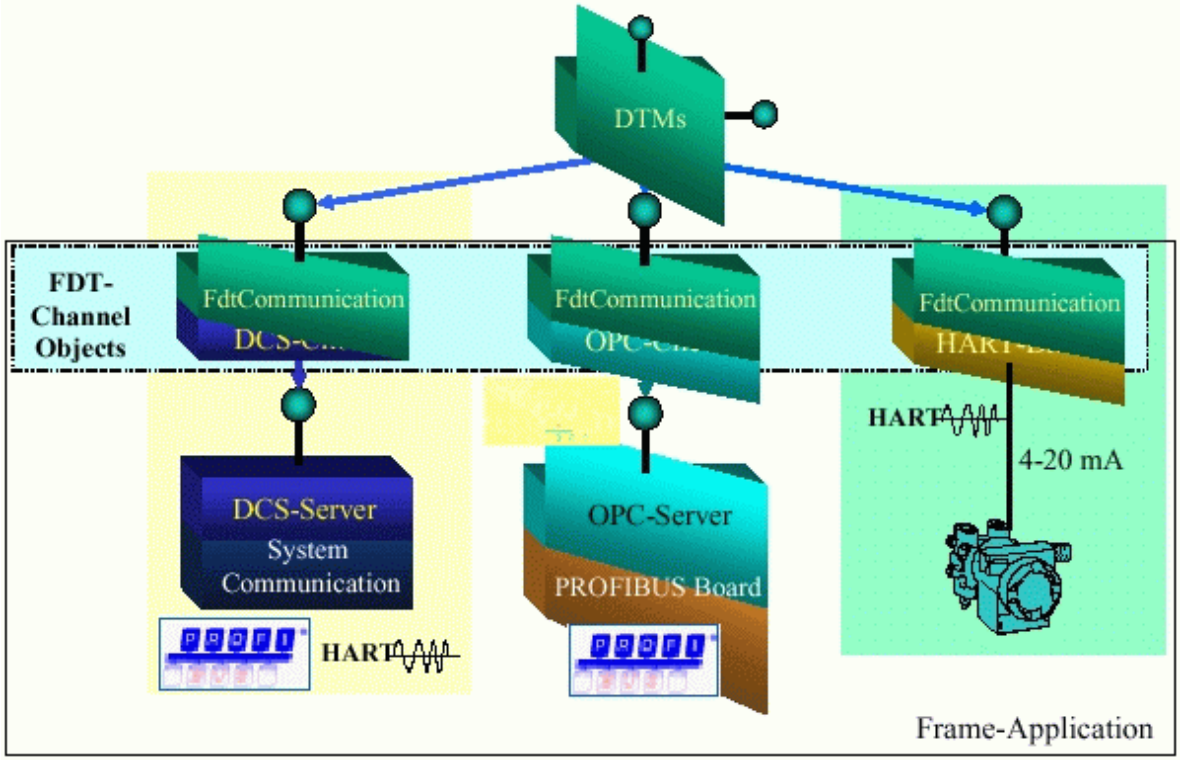

그림 2-6 : FDT Communication Layer

FDT 채널의 일반 기능성은 채널 내에 캡슐화되며 마스터 및 슬레이브 장치 또는 종속하는 소프트웨어 구성요소와의 통신을 가능하게 한다. 예를 들어, 엔지니어링 도구는 고유의 제조 업체에 특정한 방법을 사용하여 그 필드버스 마스터를 구성할 수 있다.

DTM의 관점에서, 프레임-어플리케이션과의 상호 작용을 위한 모든 작업 관련 인터페이스는 프레임-어플리케이션의 주 인터페이스를 통해 사용할 수 있다. 어떤 작업 관련 인터페이스를 제공하는 지는 프레임-어플리케이션의 능력에 따라 다르다. 각 인터페이스는 해당 장에서 자세히 설명한다.

2.5 동기화 및 직렬화 과제

한편으로 '동기화'라는 말은 클라이언트가 단일 트랜잭션으로 값과 속성을 읽고 쓰는 능력을 의미한다. 예를 들어, 대부분의 어플리케이션은 특정한 측정 채널의 값, 단위 및 제한을 동기시키려고 한다. 따라서, FDT는 XML 문서를 통한 데이터 전송을 사용한다. 문서의 모든 요소는 단일 트랜잭션 내에서 전송된다.

일반적으로, DTM은 단일 작동으로 읽고 쓰이는 데이터 항목 및 속성의 동기화를 유지해야 한다. 별도 작업에서 개별적으로 읽거나 쓰여지는 항목은 동기시킬 필요가 없다.

다른 한편, '동기화'라는 말은 '준비 완료' 및 '파라미터 변경'과 같은 상태의 신호를 보내기 위한 프레임-어플리케이션과 DTM 간의 추가적인 핸드쉐이킹(Handshaking)을 의미한다. 특히 이 핸드쉐이킹은 여러 사용자 환경에서 해당 어플리케이션의 상황에 따라 DTM을 프레임-어플리케이션 그리고 DTM 서로 간에 동기시키기 위해 필요하다. 각 DTM은 프레임-어플리케이션이 이 동기화를 수행할 수 있도록 일부 단순한 규칙을 고려해야 한다. 이러한 문제의 많은 부분은 방법에 대한 상세한 설명과 그에 따른 아래의 순차도를 통해 이해를 확실히 할 수 있다.

2.6 XML을 통한 파라미터 교환

XML을 통한 파라미터 교환의 목적은 프레임-어플리케이션과 DTM 간에 정보 교환의 수단을 제공하기 위한 것이다. 통상적으로, 공정 제어 시스템에서 관찰, 채널 할당 또는 마스터 구성과 같은 다중 클라이언트 어플리케이션은 장치의 구성에 대한 정보를 필요로 한다.

XML은 소유 형식을 대체하려는 목적이 아니고 소유 형식에 저장된 데이터에 액세스를 제공하기 위한 것이다. 데이터는 가장 합리적인 방식으로 로컬 상에 저장해야 한다. XML은 FDT 구성요소 연결을 위한 확장 가능한 표준을 제공한다. 데이터 교환은 XML 문서 또는 문서의 조각을 통해 이루어진다. XML은 데이터를 읽는 어플리케이션이 그 데이터를 해석하도록 한다. 교환 데이터에 공통된 의미를 부여하기 위해, FDT는 검증을 위해 XML schema를, 데이터 액세스를 위해 DOM(W3C의 Document Object Model)을 사용한다.

XML schema는 그 자체가 유효한 XML 구문이며 XML 데이터의 타당성을 검증하기 위해 사용한다. XML schema로 요소의 데이터 유형과 문서 구조를 검증할 수 있다.

DOM(W3C의 Document Object Model)은 문

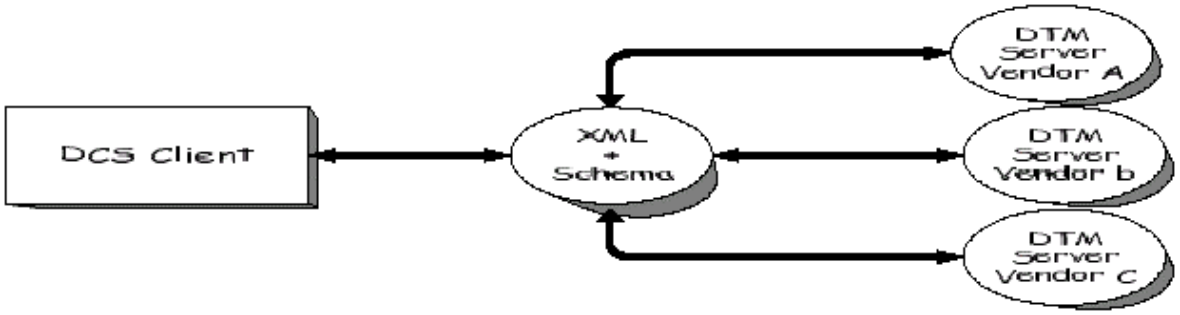

그림 2-7 : FDT Client/Server Relationship via XML

서 구조의 표준 내부 표현이며 그 목적은 프로그래머가 구성요소에 액세스하고 그 내용, 속성 및 스타일을 삭제, 추가 또는 편집하기 쉽도록 하는 것이다. 본질적으로, DOM은 프로그래머가 모든 브라우저와 서버 그리고 모든 플랫폼에서 올바로 작동하는 어플리케이션을 작성할 수 있도록 한다. 프로그래머는 서로 다른 프로그래밍 언어를 사용할 필요가 있겠지만 프로그래밍 모델을 바꿀 필요는 없다.
DOM은 보통 XML parser로 생성한다. Microsoft는 Internet Explorer 5와 함께 판매되는 XML parser를 제공한다. 따라서 DOM API를 VC++, Visual Basic 및 VBScript로 자유로이 액세스할 수 있다.
XML schema로 XML 문서를 parsing하면 잘 정의된 요소를 갖는 유효한 DOM을 얻게 된다.

FDT 개발자는 다음과 같은 이유로 항상 DOM으로 작업해야 한다.

- XML 파서는 전송된 XML 데이터로부터 DOM을 생성한다.
- 스킴은 잘 정의된 요소를 갖는 유효한 DOM을 보장한다.
- DOM은 데이터 액세스를 위한 표준 트리 및 collection-methods를 제공한다.
- DOM은 데이터 전송을 위한 XML 데이터를 생성할 수 있다.

참고:
다중 클라이언트에 걸쳐 정보가 공유되는 경우, 동일한 데이터 집합에 대한 참조를 갖는 모든 DTM에 변경된 데이터를 알림으로써 구성 정보가 다중 클라이언트에 걸쳐 일정하게 유지되도록 할 필요가 있다(예: 서로 다른 작업 스테이션상에서 동일한 장치에 대해 하나 이상의 DTM).

2.6.1 사용 예

DTM과 프레임-어플리케이션 간의 파라미터 교환은 XML 문서를 통해 이루어진다. DOM [..]을 생성하는 XML 파서 [..]를 사용하는 경우, 데이터에 대한 객체 지향적 액세스가 제공된다. 저장할(영구적으로) 인스턴스 데이터도 XML 문서로 처리하여 DTM 개발을 단순화하고 DTM 내에서 데이터를 통일되게 처리할 수 있다. 그러나 데이터를 XML로 저장하는 경우 이 데이터의 내용은 DTM만이 인식하게 된다.

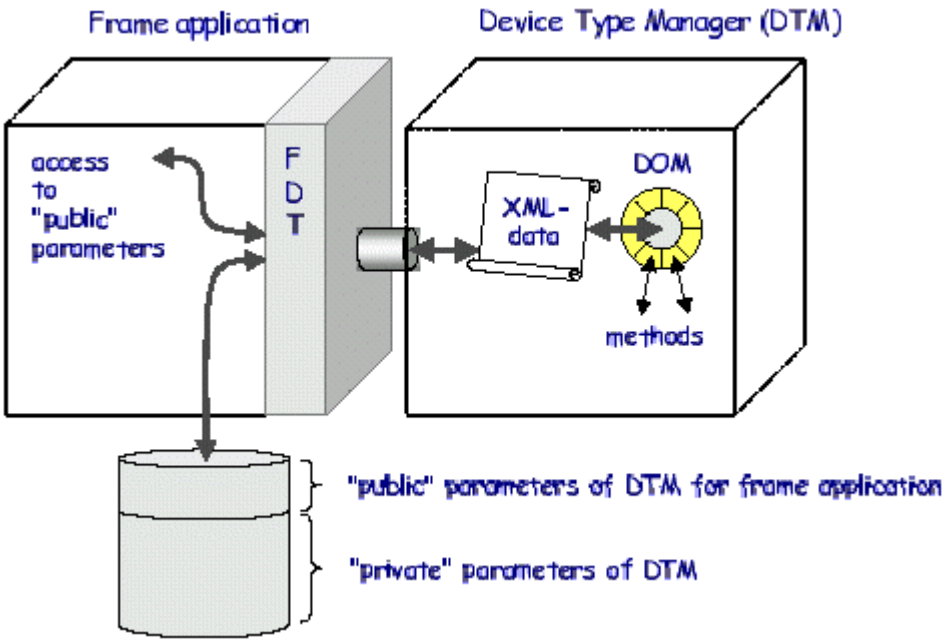

그림 2-7-1 : Data Access과 Storage

통신을 위한 XML 문서에는 장치 데이터 및 DTM과 필드 장치 간의 피어 투 피어 연결의 구성을 위한 라우팅에 필요한 정보가 포함된다.

여기서, DTM은 피어 투 피어 연결(노란색 부분)에 대한 주소 정보만을 알고 있다. 프레임 어플리케이션은 런타임 중 모든 필요한 라우팅 정보를 추가한다.

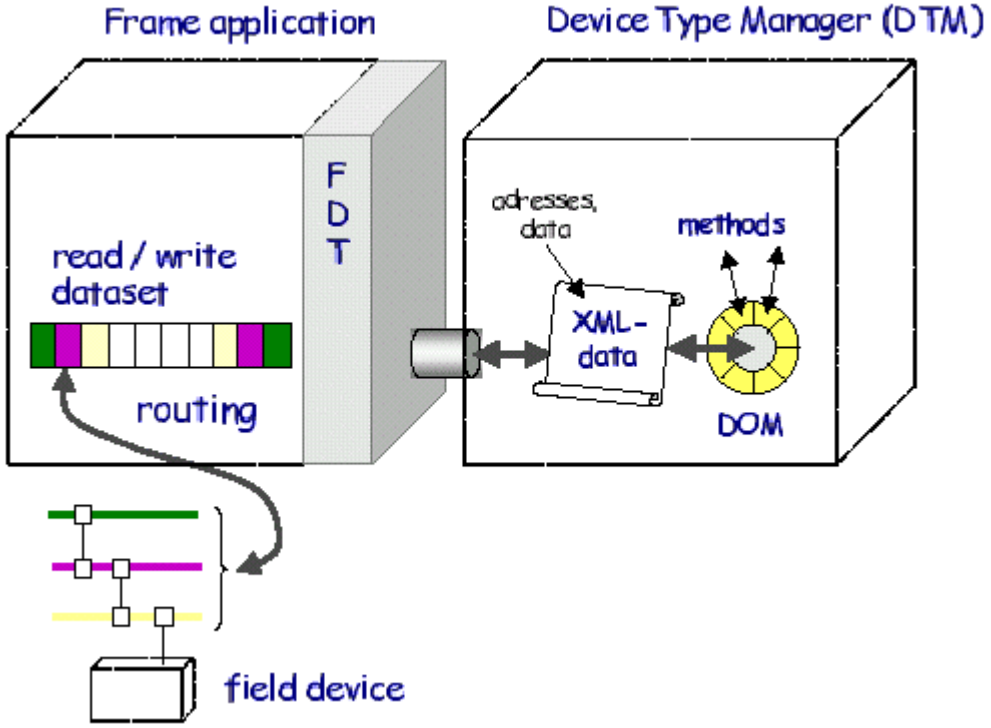

그림 2-7-2 : Communication

프로젝트 문서 내에서 필드 장치에 대한 문서를 작성하며 필드 장치별 문서의 경우에는 데이터에 대해서는 XML을, 레이아웃에 대해서는 XSL[..]이 사용된다. 기본 XSL은 프레임-어플리케이션이 지원한다.

고장 안전 필드 장치의 경우, 필드 장치로부터 업로드 후 컨트롤러(예: PLC)에 저장되는 인스턴스 데이터를 DTM이 검증해야 한다. 이 데이터의 교환 형식도 XML 문서이다.

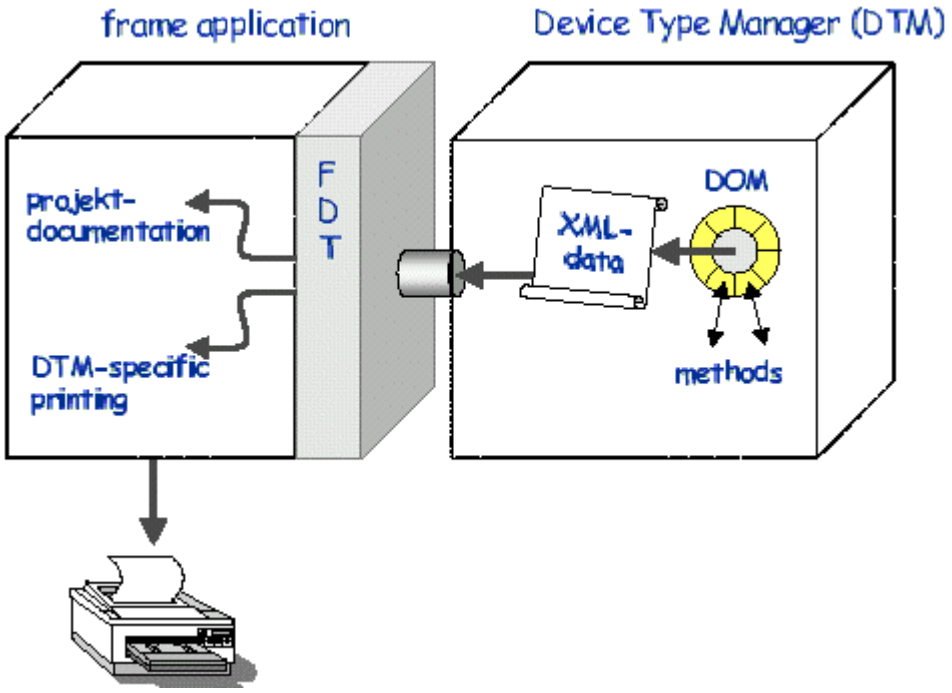

그림 2-7-3 : Documentation

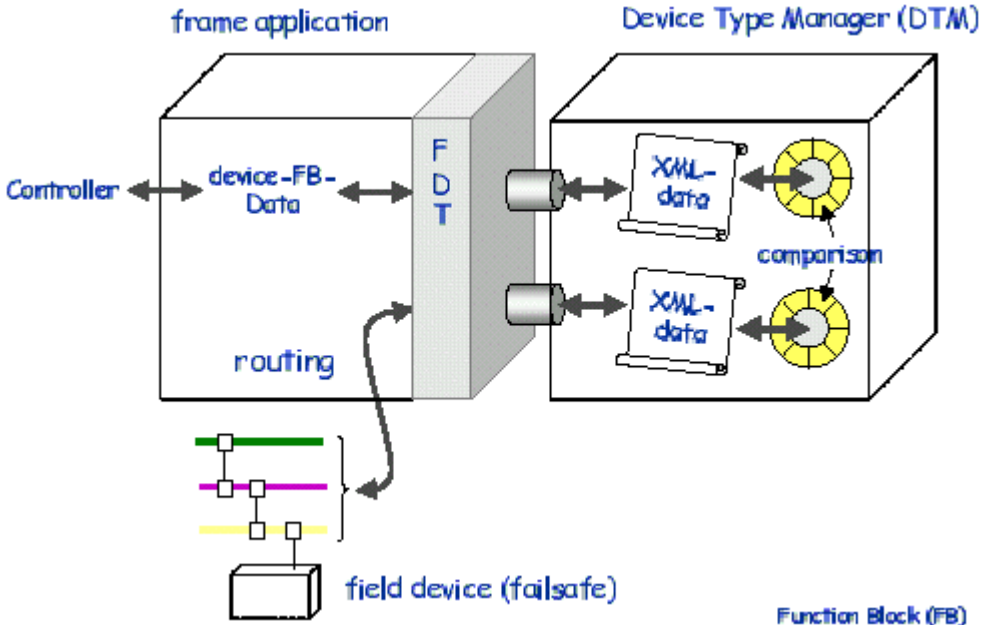

그림 2-7-4 : Parameter verification in case of failsafe device

2.7 지속적 저장에 대한 설명

2.7.1 지속성에 대한 개요

기본적으로, 두 가지 종류의 데이터 즉, 인스턴스 관련 데이터와 비인스턴스 관련 데이터를 생각할 수 있다. 인스턴스 관련은 DTM 자체에 속하고 비 인스턴스 관련은 프로젝트에 속하거나 포괄적(예: 라이브러리)이다.

DTM에 대해 다음과 같은 종류의 지속성 요구조건을 생각할 수 있다.

- 로컬로 데이터를 저장하지 않는 보통의 DTM은 프레임-어플리케이션의 프로젝트 데이터 내에 비밀 데이터를 저장하고 복원할 수 있어야 한다.

- 추가적인 고유 로컬 저장을 사용하는 DTM은 프레임-어플리케이션의 프로젝트 데이터 내에 그 인스턴스 관련 데이터에 대한 참조를 저장할 수 있어야 한다. 이러한 참조를 통해 DTM은 고유 비밀 저장에서 요청되는 데이터를 찾을 수 있다. 그러나 비밀 저장 메커니즘을 사용하는 DTM은 여러 사용자 및 여러 클라이언트가 데이터에 액세스하도록 데이터 일관성을 보장해야 한다. 불러오기 및 내보내기의 경우, 이러한 DTM은 프레임-어플리케이션으로의 별도 인터페이스를 통해 그 비밀 저장 데이터를 제공해야 한다.

프레임-어플리케이션은 DTM의 IdtmParameters 인터페이스를 통해 알 수 있는 DTM의 비밀 데이터 및 모든 장치 파라미터를 저장할 수 있어야 한다. 이러한 저장 구성요소(예: 엔지니어링 도구의 데이터베이스)는 DTM 및 프레임-어플리케이션에 의한 동시적 데이터 액세스와 잠금을 처리할 수 있어야 한다. 동기화의 통지는 IFdtContainer 인터페이스를 통해 수행된다.

프레임-어플리케이션의 저장 구성요소를 사용하려면 DTM이 표준 COM 인터페이스인 IPersistPropertyBag 및 IPersistStreamInit를 구현해야 한다. DTM이 저장을 수행하는 방식 또는 어떤 종류의 DTM 비밀 데이터가 저장되는 지는 결정되지 않는다.

프레임-어플리케이션은 DTM에게 비밀 데이터의 저장을 요청한다. 프레임-어플리케이션이 이를 요청하였을 때 DTM은 자신의 완전한 상태를 재 구성할 수 있어야 한다. 이는 DTM의 IPersistXXX::Load 함수를 호출하여 수행한다. IPersistXXX::Load를 몇 차례 호출하여 DTM 객체를 다시 로드하는 것은 모든 DTM 공급 업체가 지원하지 않을 수 있다. IPersistXXX:Save 요청이 있는 경우 DTM은 비밀 데이터를 프레임-어플리케이션 내에 저장해야 한다. 프레임-어플리케이션이 IPersistXXX:InitNew 매서드를 호출하는 경우 새로운 인스턴스에 대한 DTM 객체가 초기화되어야 한다.

추가적인 고유 데이터 저장을 사용하는 DTM은 IPersist 인터페이스를 통해 시운전하는데 필요한 모든 데이터를 제공해야 한다. IPersist 인터페이스를 통해 제공되지 않는 비밀 데이터는 IDtmImportExport 인터페이스를 통해 불러오기 및 내보내기를 수행하기 위해 프레임-어플리케이션에 제공되어야 한다. 이러한 불러오기 및 내보내기 데이터의 저장과 검색을 위해 IStream 객체도 사용할 수 있다.

2.7.2 지속성 인터페이스

IPersistStreamInit 및 IPersistPropertyBag에 대한 자세한 내용은 MSDN과 같은 표준 Microsoft 문서를 참고할 수 있다.

2.8 세션 모델의 기본 특징:

프레임-어플리케이션은 여러 사용자 지원, 안전한 데이터 관리 및 데이터 일관성을 위해 세션 모델을 구현해야 한다.

통상적으로, 세션은 프레임-어플리케이션의 사용자 인터페이스상에서 DTM의 시작으로 시작하며 DTM의 종결로 끝난다. 프레임-어플리케이션 내에서 사용자가 두 번째 DTM을 시작하는 경우, 별도 세션이 만들어진다.

이러한 DTM내부에서 개방된 모든 데이터 객체는 그 세션에서 등록된다. 세션의 마지막에 세션의 수정된 모든 데이터는 동기적으로 저장되어야 한다(사용자가 수정을 저장하려는 경우).

DTM이 데이터 객체를 쓰기 액세스로 개방하는 경우 데이터 객체는 잠긴다. DTM에 의해 데이터가 잠겨있는 동안 기타 DTM은 읽기로 액세스 할 수만 있으며 쓰기로 액세스 할 수는 없다.

세션은 세션 내의 잠긴 데이터에 대한 권한을 가지고 또는 권한 없이 만들 수 있다. 이 권한을 가지고 만드는 경우, DTM은 이 세션에서 데이터를 잠글 수 없다. 데이터의 수정을 허락하지 않는 사용자 역할 또는 어플리케이션 ID로 DTM을 시작하는 경우 프레임-어플리케이션은 이 권한 없이 세션을 만든다. 기타 모든 경우에는 데이터 잠금에 대한 권한을 가지고 세션을 연다.

2.9 필드버스 독립 통합

필드버스에 상관없는 프레임-어플리케이션으로의 통합은 범주 ID(CATID)로 실현된다. 이러한 범주 ID는 FDT 규정 내에 UUID로 정의된다.

한편, 범주 ID는 DTM 또는 채널이 지원하는 필드버스 유형을 지정하기 위해 Windows 레지스트리에서 사용된다. 범주 ID를 통해 등록하면 프레임-어플리케이션이 DTM의 구체적 예에 대해 미리 선택할 수 있다.

다른 한편, 동일한 범주 ID는 DTM과 프레임-어플리케이션 간의 상호 작용에 사용되는 XML 문서 내에서 사용된다. 버스 정보는 DTMParameterSchema 또는 FDTHARTChannelParameterSchema와 같은 XML 스키마 내에서 지정되며 그에 따른 버스 독립 인터페이스를 통해 사용할 수 있다. 지원되는 필드버스에 대한 정보를 사용하여 통신 중 연결 또는 버스 토폴로지의 유효성을 검증할 수 있다.

추가 필드버스의 경우, 범주 ID의 정의는 또 다른 필드버스의 특정 XML 스키마에서 확장된 보충물일 수 있다. 속성 및 각 FDT 구성요소 내의 식별자가 검증을 위해 이 정보를 사용할 수 있는 것처럼 필드버스 독립 FDT 스키마는 범주 ID를 처리한다.

이 메커니즘으로 인해, 프레임-어플리케이션은 직접 연결된 버스에 대한 범주 ID만을 알아야 한다. 낮은 하위 Topology의 경우, 프레임-어플리케이션은 검증을 위해 XML 문서 내의 범주 속성에 대한 내용을 필요로 한다. 이러한 경우가 발생하는 경우, FDT 설치 요구조건에 따라 이 범주 ID를 사용하여 DTM을 설치하고 소유 버스 시스템에 대해 새로운 범주를 정의하는 것은 DTM 개발자에게 달려있다.

이상으로 FDT의 기초까지 알아보았다.

본 내용은 3장부터는 전체적으로 다음과 같이 구성되어 있으므로 다음 내용은 PROFIBUS Guideline "FDT Interface Specification"을 참조 바랍니다.

3 FDT INTERFACE

3.1 Overview of the FDT Interfaces

3.2 FDT Obj25

3.2.1 Device related Point of View

3.2.2 Task related Point of View

3.2.3 DTM State Machine

3.3 Device Type Manager

3.3.1 Interface IDtm

3.3.2 Interface IDtmActiveXInformation

3.3.3 Interface IDtmApplication

3.3.4 Interface IDtmChannel

3.3.5 Interface IDtmDocumentation

3.3.6 Interface IDtmDiagnosis

3.3.7 Interface IDtmExtendedFunction

3.3.8 Interface IDtmImportExport

3.3.9 Interface IDtmInformation

3.3.10 Interface IDtmOnlineDiagnosis

3.3.11 Interface IDtmOnlineParameter

3.3.12 Interface IDtmParameter

3.3.13 Interface IFdtCommunicationEvents

3.3.14 Interface IFdtEvents

3.4 DTM ActiveXControl

3.4.1 Interface IDtmActiveXControl

3.5 FDT Channel

3.5.1 Interface IFdtChannel

3.5.2 Interface IFdtChannelActiveXInformation

3.5.3 Interface IFdtCommunication

3.5.4 Interface IFdtCommunicationEvents

3.5.5 Interface IFdtChannelSubTopology

3.5.6 Interface IFdtFunctionBlockData

3.6 FDT Channel ActiveXControl

3.6.1 Interface IFdtChannelActiveXControl

3 FDT Interface

3.7 FDT Container

3.7.1 Interface IDtmEvents

3.7.2 Interface IDtmAuditTrailEvents

3.7.3 Interface IFdtBulkData

3.7.4 Interface IFdtContainer

3.7.5 Interface IFdtTopology

3.8 General Information

3.8.1 Version Interoperability

3.8.2 Ownership of Memory

3.8.3 Standard Interfaces

3.8.4 Dual Interfaces

3.8.5 COM Server Registration

3.8.6 Unicode

3.8.7 Nul Strings and Null Pointers

3.8.8 Asynchronous vs. Synchronous Behavior

3.8.9 Parameter and Data Types

4 FDT SESSION MODEL AND USE CASES

----

5 FDT SEQUENCE CHARTS
