* Start by defining a data model, then map the data-model to D DS domains, data types and Topics. 일반적인 TCP/IP 또는 시리얼통신 등을 위하여 패킷을 정의할 때와 유사한 방법으로 데이터 모델을 정의할 필요가 있으며, 이렇게 정의된 데이터 모델을 도메인, 데이터 타입 및 토픽으로 매핑을 시켜야 한다.
* 이렇게 함으로써 서브 시스템을 별도의 도메인 단위로 분리할 수 있으며, 도메인 단위로 분리된 경우에 상당한 성능상의 이점을 갖게 된다.
* 데이터 모델 형식으로 정의를 한 후 UML 클래스 다이어그램과 같은 관계성을 검증할 수 있다. 다음과 같은 체크 리스트를 적용할 수 있다.
  + UML 클래스 다이어그램과 같이 자료 구조와 그 상호 관계 측면에서 데이터 모델을 표현할 수 있다.
  + 이러한 데이터 모델을 통하여 프로그래밍이나 소프트웨어 공학에 대한 지식이 없더라도 문제 영역에 대하여 논의 및 이해가 좀더 쉽게 가능하다.
  + 데이터 모델은 프로그래밍 언어, 플랫폼 등에 독립적인 UML, XML 또는 IDL과 같은 모델링 언어로 표현할 수 있다.
  + 데이터 모델은 위치 투명성을 가져야 한다. 즉, 데이터 모델은 하나의 컴퓨터에 배치될 수도 있으며, 원격지의 어디에도 배치될 수 있다.
* 토픽 정의 시 토픽의 아이템으로 사용되는 데이터 타입은 새로 정의해서 사용하는 것이 유리하다. 가능하면 string 이나 opaque byte array 타입의 사용을 지양해야 한다. 이유는 데이터의 내부 형태가 불분명하기 때문에 미들웨어가 제공하는 기능을 충분히 활용하기가 어렵기 때문이다. 예를 들어 필터링 기능 사용시 또는 DBMS에 저장 시에 제약을 받을 수 있으며, marshalling 및 demarshalling 시 좀 더 복잡한 과정을 거칠 수 있다.
* 또한 가능한 경우에 키를 갖는 토픽을 정의하는 것이 여러 가지 경우에 유리하다. 예를 들어 토픽을 DBMS에 저장하고 조회하는 경우에 키를 가짐으로 인하여 속도를 높일 수 있으며, 어플리케이션끼리 샘플 송수신 시 필터링 기능을 사용하여 관심 있는 샘플만 수신이 가능하기 때문이다.

It is often useful to divide the data model into a control plane and a data plane. This allows separation of the

application and system management functions from the logic of the application components them-selves.

A checklist for a well-defined model:

• It can be fully expressed in terms of data structures and their relationships.

• It can be easily explained and understood to someone familiar with the problem domain, without

requiring the person to be knowledgeable in programming or software engineering.

• It is defined using a platform-independent modeling language, such as UML, XML or IDL. This way the

model is not tied to a specific platform, technology, or deployment scenario. The same model could be

deployed on top of different technology stacks (DDS, WSDL, message bus), programming languages

(Java, C++, C#), and configurations (platforms, networks, etc.).

• There are no location dependencies in the model. It can be deployed on a single computer or on a

network.

• The different scenarios and behaviors of the system can be expressed in terms of changes to the state of

elements in the data model.

* Configure QoS using XML Profiles. XML 파일을 이용하여 QoS설정하는 것이 QoS 관련 API를 이용하는 것보다 유연하며, 관리하기가 쉽다. 또한 QoS 값이 실행 파일 외부에 존재하기 때문에 개발이 완료된 후 재컴파일이 없이 어플리케이션 튜닝 등에 쉽게 적용할 수 있기 때문에 매우 유리하다.
* 일정 규모 이상의 프로젝트에서는 DDS API를 직접 사용하기 보다는 프로젝트 초기에 wrapper를 만들어서 사용하는 경우가 있다. 이렇게 함으로 인하여 종속성이 줄어드는 단점이 있지만 이 경우에도 주의할 점은 방대한 DDS의 특장점을 잘 이해하고 있어야 개발 중에 이미 방대해진 wrapper를 변경하는 경우가 없게 된다. 특히나 각 QoS 정책 중 주요 파라메타는 DDS에 있어서 매우 중요하면서도 샘플 송수신 시 미치는 영향이 매우 크기 때문에 특히나 주의해야 한다.