Aspect-Oriented Logical Architecture Design

A Layered Perspective Applied to Data Warehousing
Aspect-Oriented Architecture

- Aspectual Composition Layer composed of Composition Rules

![Diagram showing various views and their interactions in the aspectual composition layer.](image-url)
Aspects and Data Warehouses

- Data warehouse system’s design is strongly layered.
- Data Warehouse system’s have many crosscutting concerns: METADATA.
Open Questions

- How to derive an architecture from an AO specification?
- How to trace aspects throughout the software lifecycle?
Aspects and Data Warehouses

Why:

- Data warehouse system’s design is strongly layered: Operational View; DW view; Data Marts View; Individual View;
- Data Warehouse system’s have many crosscutting concerns: METADATA;
- A true Data Information System (loosely called Data Warehouse System) should be completely metadata driven and can benefit from a aspect-oriented specification and design.
Architectural Aspects

- **Architectural join points**: are well-defined points in the software architecture.
- **Architectural pointcuts**: refer to collections of architectural join points.
- **Architectural advices**: define the transformations to be performed at join points.
- **Architectural aspects**: are views formed by architectural pointcuts and advices.
- **Aspectual Views**: are groups of architectural aspects with a common purpose or features to the system’s architecture.
- **Aspectual Composition Layer**: is an architectural view consisting of composition rules where architectural pointcuts and architectural advices for the pairs of interaction between architectural aspects and components are defined. Therefore, special composition rules for the interaction between architectural aspects that are specified in the aspectual view can also be specified within this layer.
- **Composition Rule**: specifies the pairs of interactions between components and aspects. The rule includes the definition of how, when and where each aspect must be linked to each component to which it is bounded. Specifying separately each pair of interaction increases the architecture modularity and easiness of change: it allows the swapping of aspects interacting with components, without changing anything else but that composition rule.