

# Building Data Warehouses That Fully Support Time Variance ... And A Business Case

## Associative Data Modeling and ISO Standards

## Kalido Architecture and Overview

Stephen Pace  
Sr. Technical Consultant  
Kalido, Inc.

# Speaker Bio



**Stephen Pace**  
**Senior Technical Consultant**  
**Kalido**

Before transferring to Kalido, Inc., Stephen spent over 10 years within Shell IT working in a number of areas including application and Web development, database design, and data warehouse consultancy. Stephen was part of the Kalido development team, and today plays the dual role of pre-sales consultant and technology evangelist. Stephen has a BS in Computer Science from Texas A&M University, and he's also an active member of The Data Warehousing Institute (TDWI).

# DAMA Discussion

## 1) Time Variance and the Data Warehouse

- a) Standard Industry Techniques (What the Gurus Say)
- b) Business Case (Sad Story of Abbots Autos)

## 2) Associative Data Modeling and ISO Standards

And Time Permitting:

## 3) Kalido Architecture and Overview

## 4) Time Travel – Setting Effective Dates In Browser

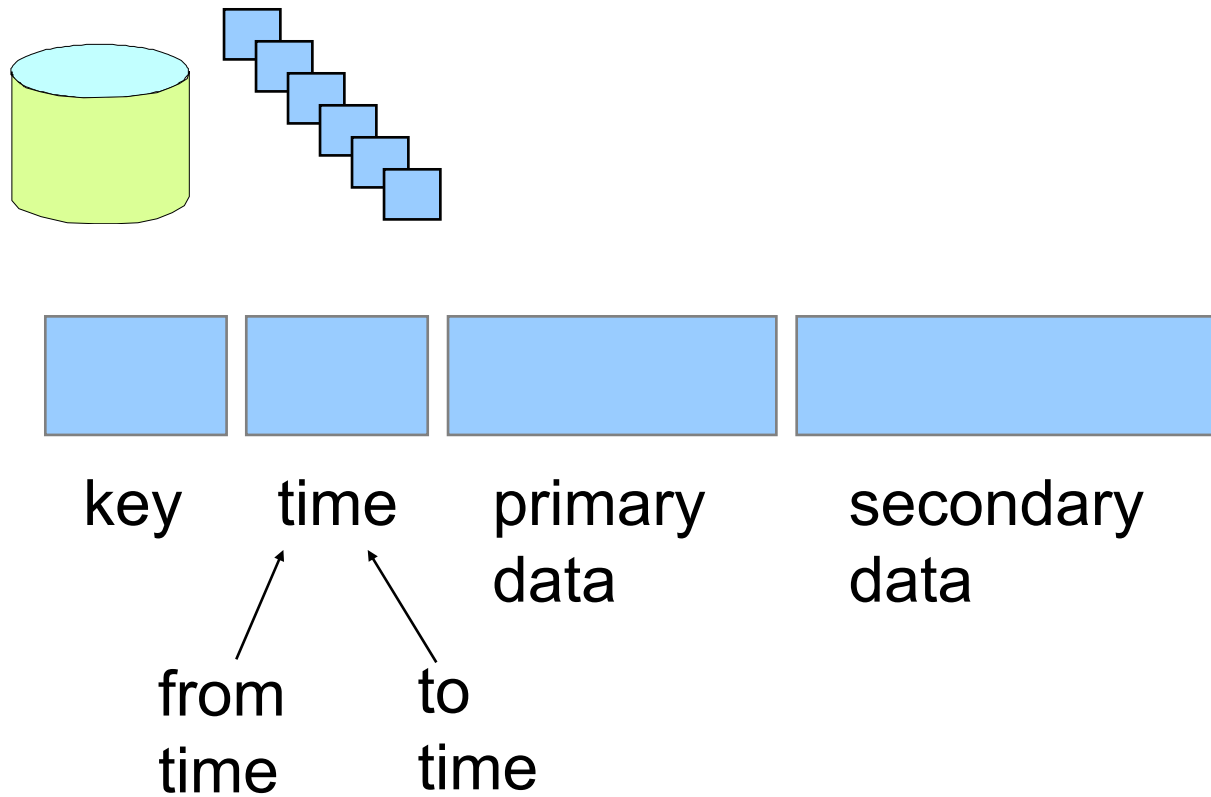
# Time Variance – Basic Overview

Bill Inmon – Continuous Time Vs. Discrete Time

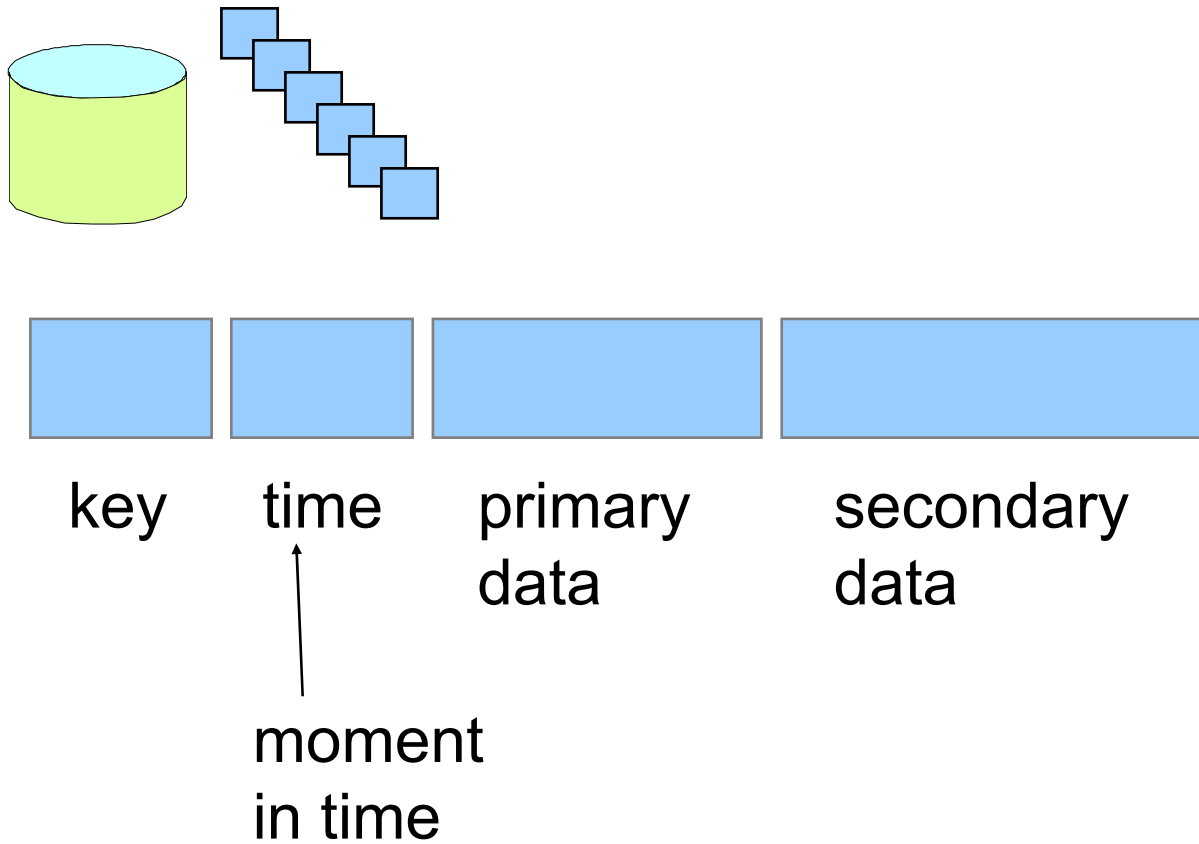
Ralph Kimball – Slowly Changing Dimensions

- > Type 1 – Overwrite the Value
  - “Rewriting History”
- > Type 2 – Add A Dimension Row
  - “Keeping Track of History”
- > Type 3 – Add A Dimension Column
  - “Keeping Track of History in Parallel”
- > Hybrids (Type “6”)
  - Predictable Changes with Multiple Version Overlays
  - Unpredictable Changes with Single-Version Overlay
  - More Rapidly Changing Dimensions

# Bill Inmon – Continuous Time Data



# Bill Inmon – Discrete Time Data



## Overwrite the Value – “Rewriting History”

- > Very Easy to Implement
- > Can be used to correct errors, but...

### Issues:

No Time Variance – No History

Think About Effect to Summary / Aggregate Tables

# Ralph Kimball – Type 2 SCD

## Adding A Dimension Row – “Keeping Track of History”

- > Most Common Type

### Issues:

Data Fragmentation / Partitions History

No Alternate Views (e.g. Old Data With Current View)

Surrogate Key Maintenance

## Adding A Dimension Column – “Keeping Track of Versions in Parallel”

- > Provides an alternate view (perhaps current with previous)

### Issues:

Default Case Only Allows Two Views  
Maintenance of this can be troublesome

# Ralph Kimball – Hybrid SCD (Type “6”)

## Mixing Elements of Type 2 and 3 Together

- Predictable Changes with Multiple Version Overlays
- Unpredictable Changes with Single-Version Overlay
- More Rapidly Changing Dimensions

> Most Flexible

Issues:

But Maintenance Could Be an Issue

Potentially Can Provide Complexity to The End Users

# Business Case For Time Variance: The Sad Story of Abbots Autos

<Switch to FTK Tutorial 130>

Break

When We Return:

Associative Data Modeling

# Associative Data Modeling

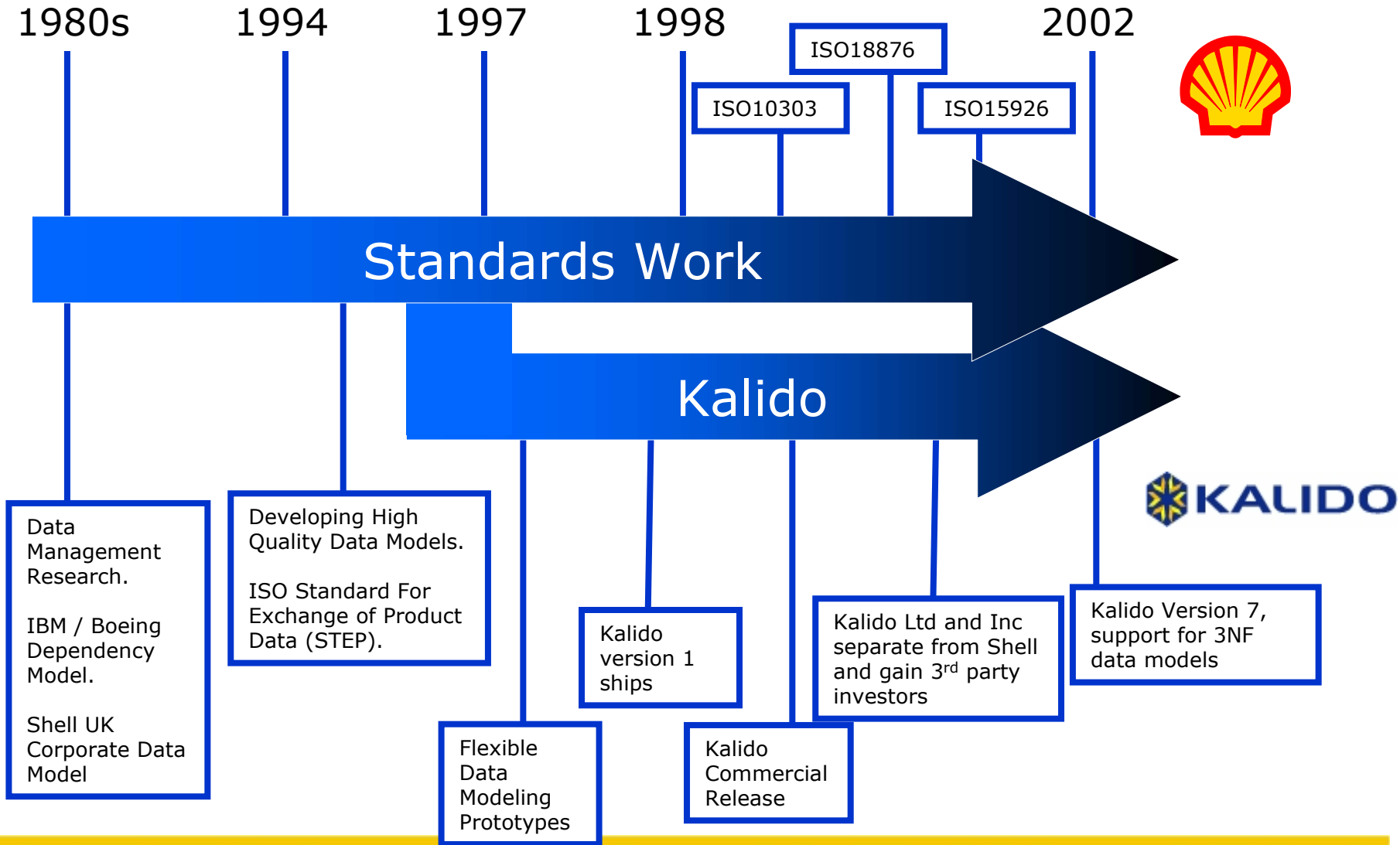
Introduction and Context

Modeling Examples

The Generic Entity Framework and the Principles

Applicability to Data Warehousing

# Kalido History



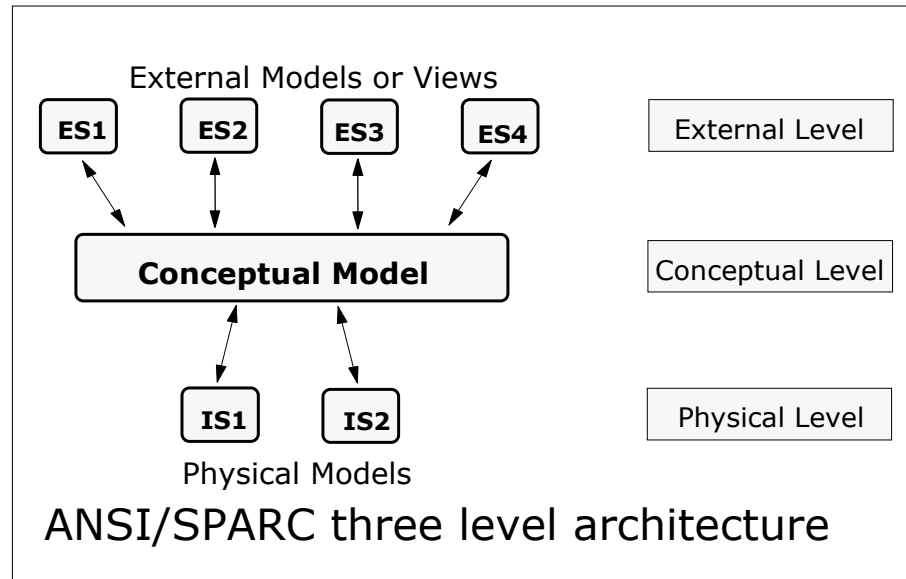
## Shell Standards Work - "Developing High Quality Data Models"

*Systems often cost more than they should, to build, operate, and maintain.*

*They may also constrain the business rather than support it.*

*A major cause is that the quality of the data models implemented in our systems (own built or packages) is poor.*

Professor Matthew West (1994)



Shell Developed the Generic Entity Framework to provide a standardized conceptual model for its business and its partners.

# Issues with conventional approaches

Many problems are found as a result of the way data is held in systems:

- Arbitrary or inappropriate restrictions are placed on the data that can be held.
- History data cannot be held.
- Fudge or false data may be introduced to overcome restrictions.
- Uncontrolled redundancy of data requiring reconciliation of different versions.
- Difficulty in integrating data from different sources due to incompatibility in definitions and format.
- The same data structures may be replicated.
- The same functionality may be replicated.

Examples of financial and time penalties incurred:

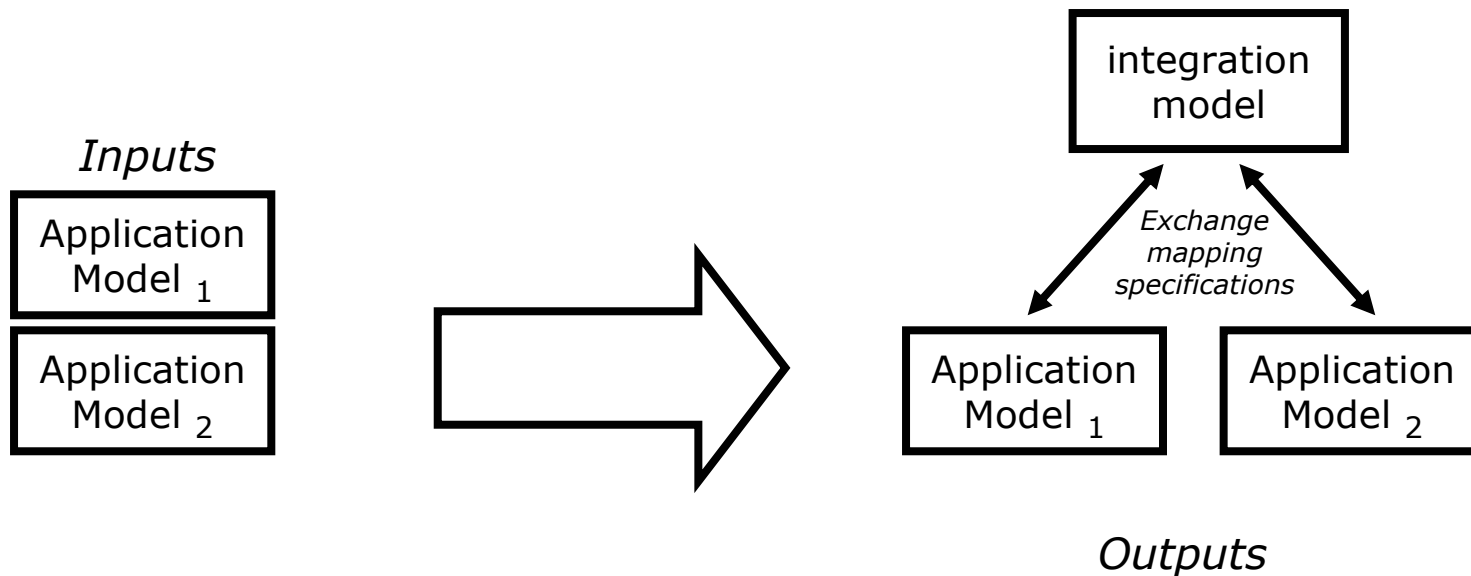
- Translation of data meaning is expensive - it can account for 25-70% of implementation cost.
- The need to translate data means that users of different systems can often only share data sequentially, and not concurrently – increasing the time required for critical business processes.
- Quality suffers as duplication of data is inefficient - inviting errors resulting in inferior decisions.
- Staff time is wasted trying to locate and reconcile data.
- There is a slower response to the need for change in systems.

All of these problems either restrict the way a company does business, or add to the cost of doing business.



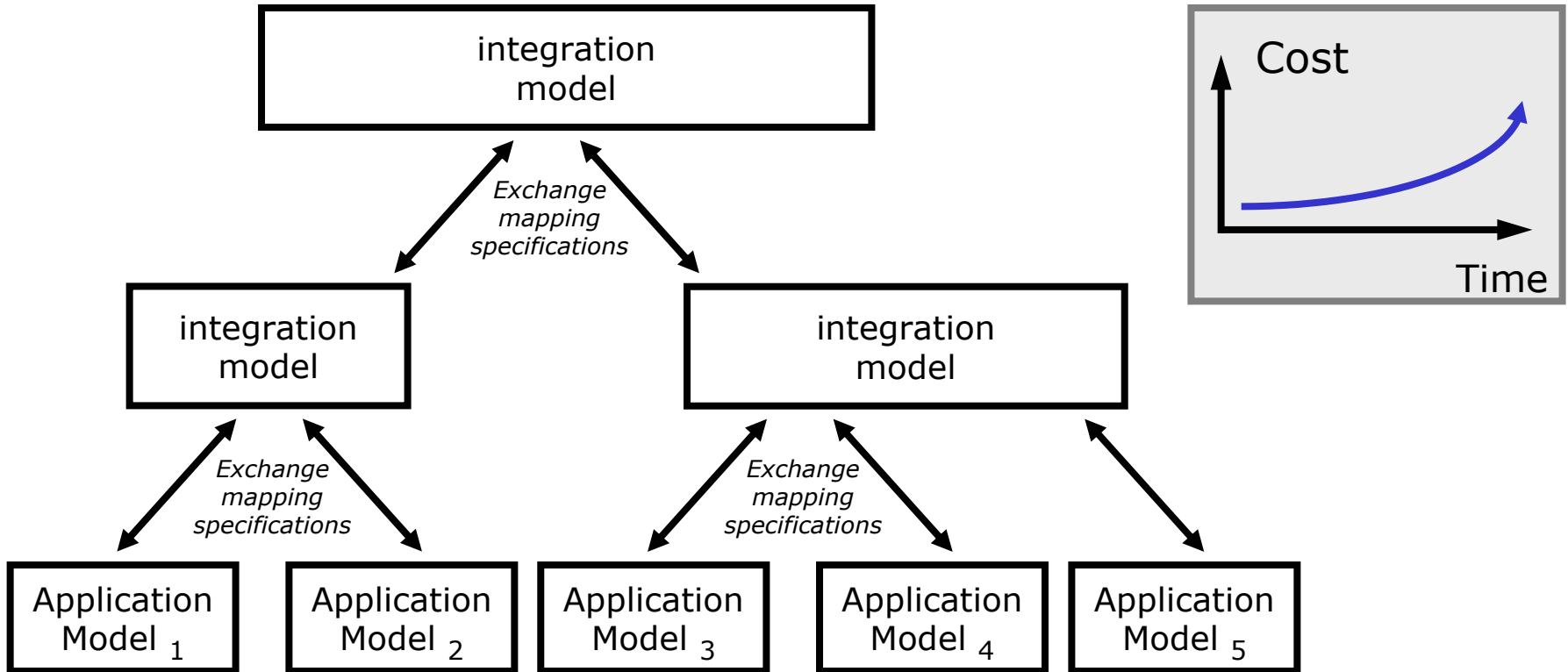
# Integrating 2 models is always possible

It is always possible to create an integration model that is the superset of 2 existing models



But these models are difficult to produce  
How do you resolve conflicts?  
What if the definitions are not the same?  
Can you get agreement?  
Is it too complex?

# However ...



What happens when you integrate more models?

Does this change the integration model?  
Do you need multiple layers of integration model?  
What if the definitions are not the same?

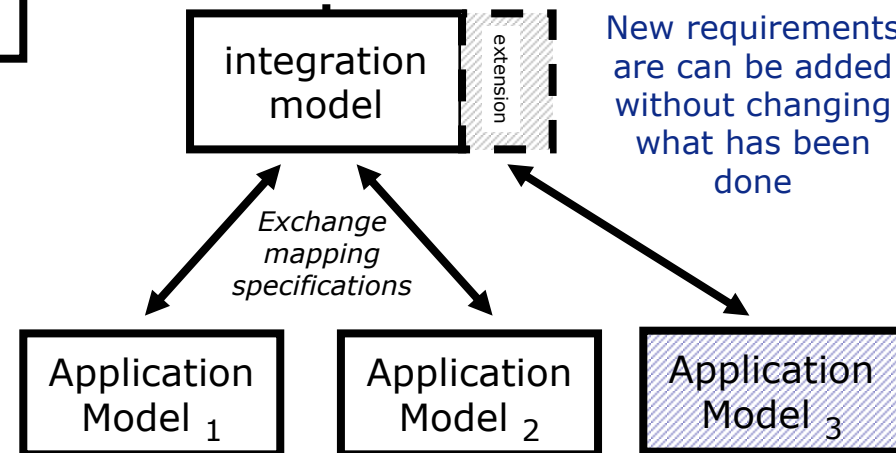
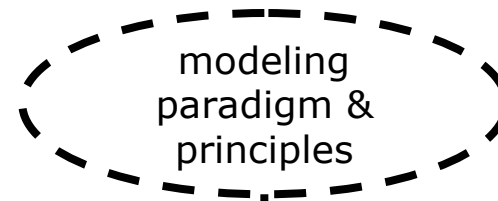
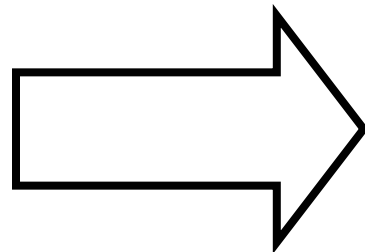
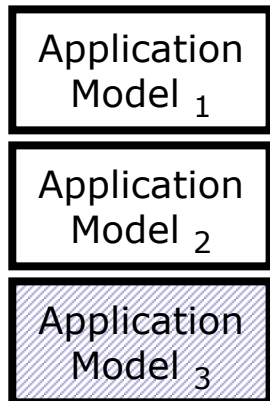
Can you get agreement?  
Is it too complex?  
How do you resolve conflicts?

# Integration Methodology – ISO 18876

An architecture and methodology for integrating of data for exchange, access and sharing data

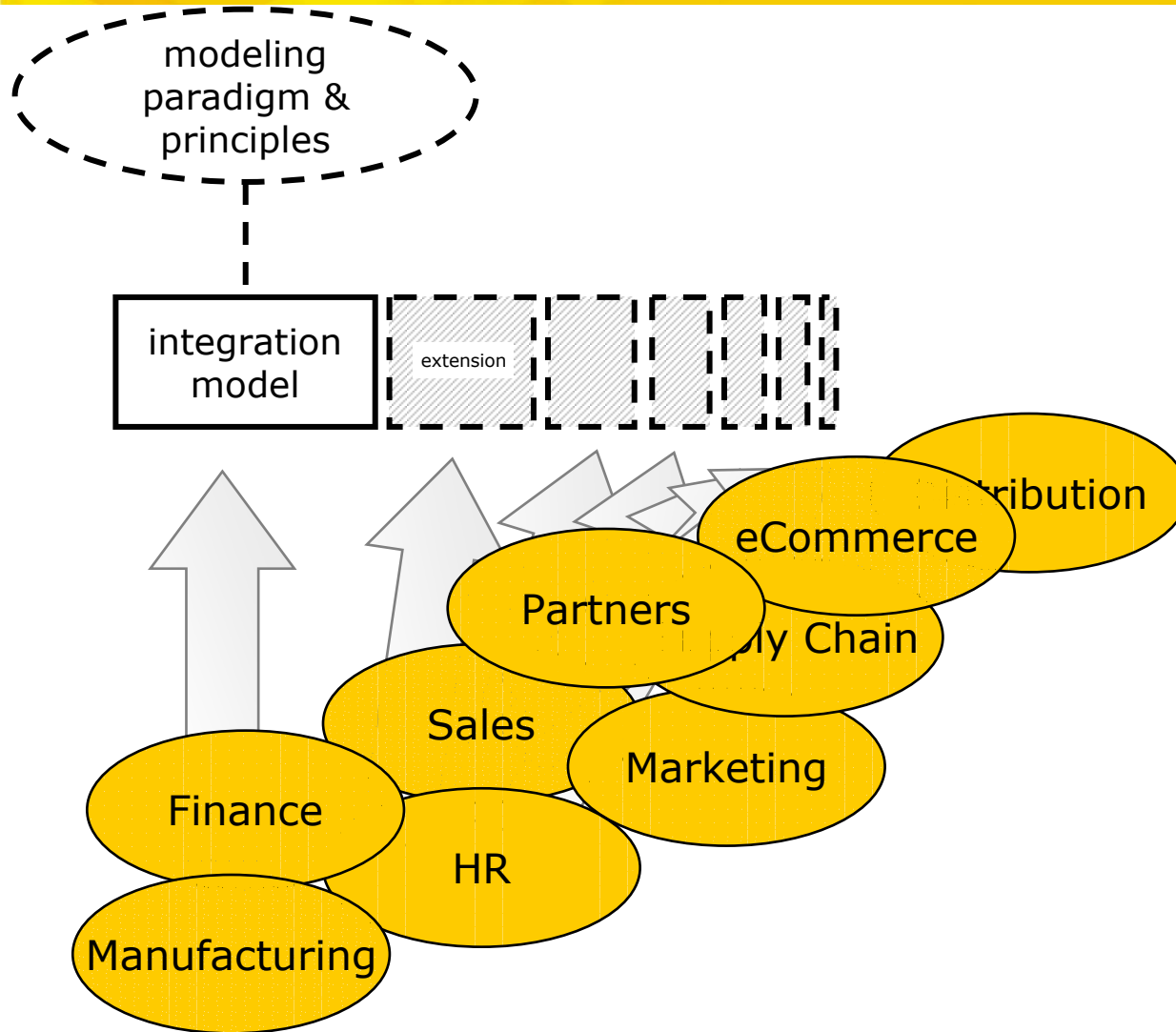
- from different sources or different contexts
- described by different models, or defined in different modeling languages
- resolving conflict between models developed with different objectives
- translating data between different encodings
- translating models between different modeling languages

*Inputs*



*A conceptual framework for Integration Models*

# An iterative approach to the enterprise model



The Integration model approach allows iterative development

The model can be evolved over time

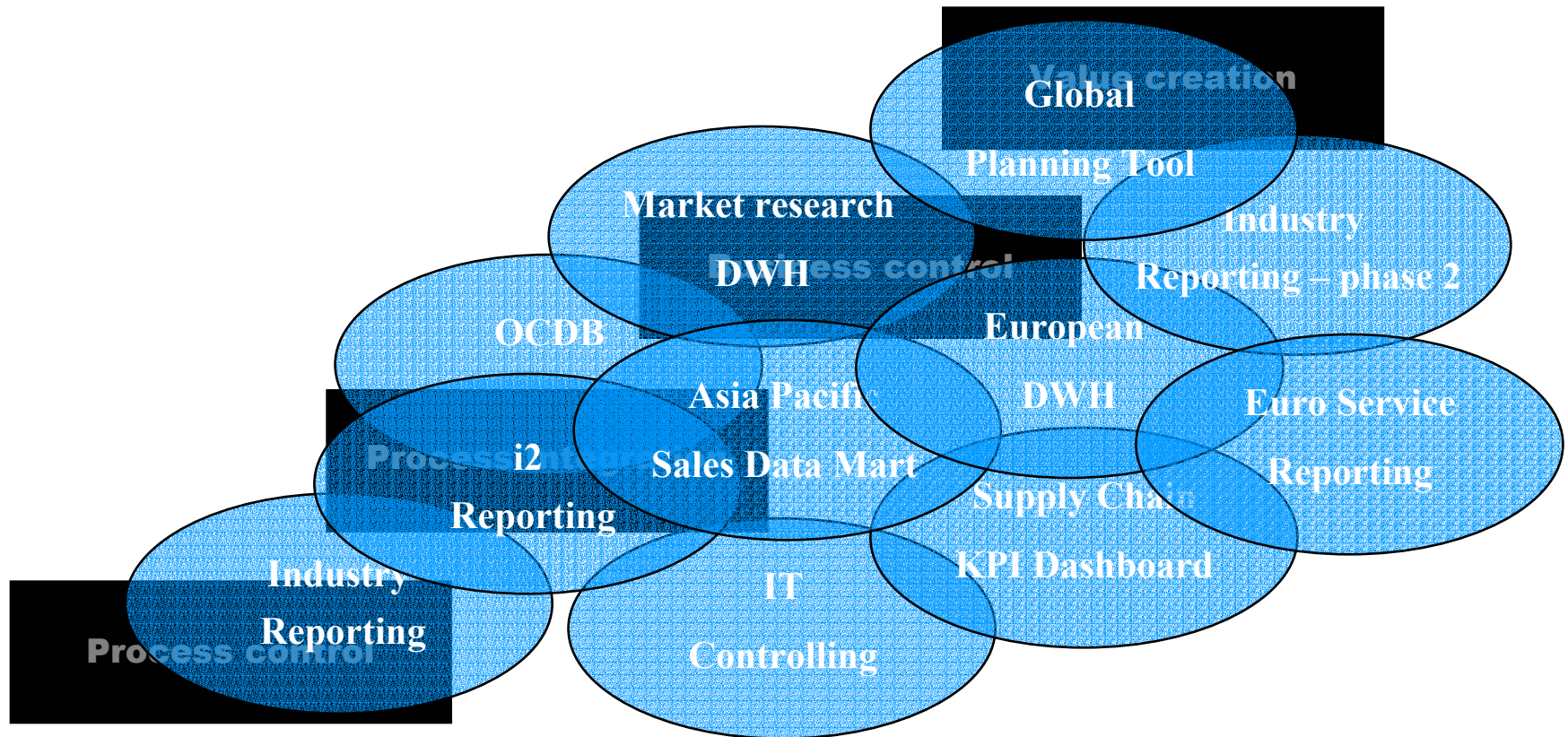
Based on the principles, extensions get simpler with each extra iteration

Finally, new subjects are incorporated completely within the model

# A real example – Philips Consumer Electronics

Evolve from Tactical Reporting to Strategic Value Creation

**PHILIPS**

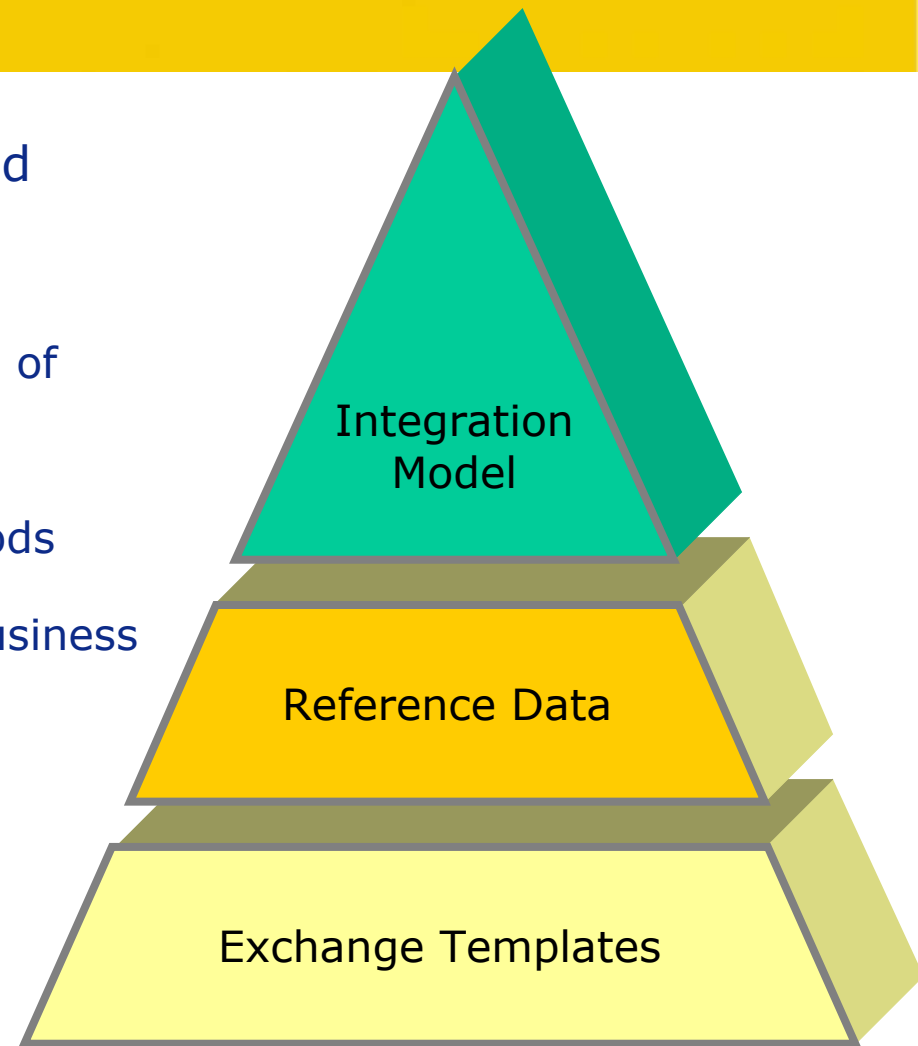


Iterative, evolutionary, repeatable, reusable, low risk solution

# A new approach

Enterprise information integration and exchange relies on 3 components

- > An integration data model capable of handling multiple business models simultaneously, over all time periods
- > Reference data that defines the business model, and content
- > Exchange templates to allow systems to map to the integration model and exchange data



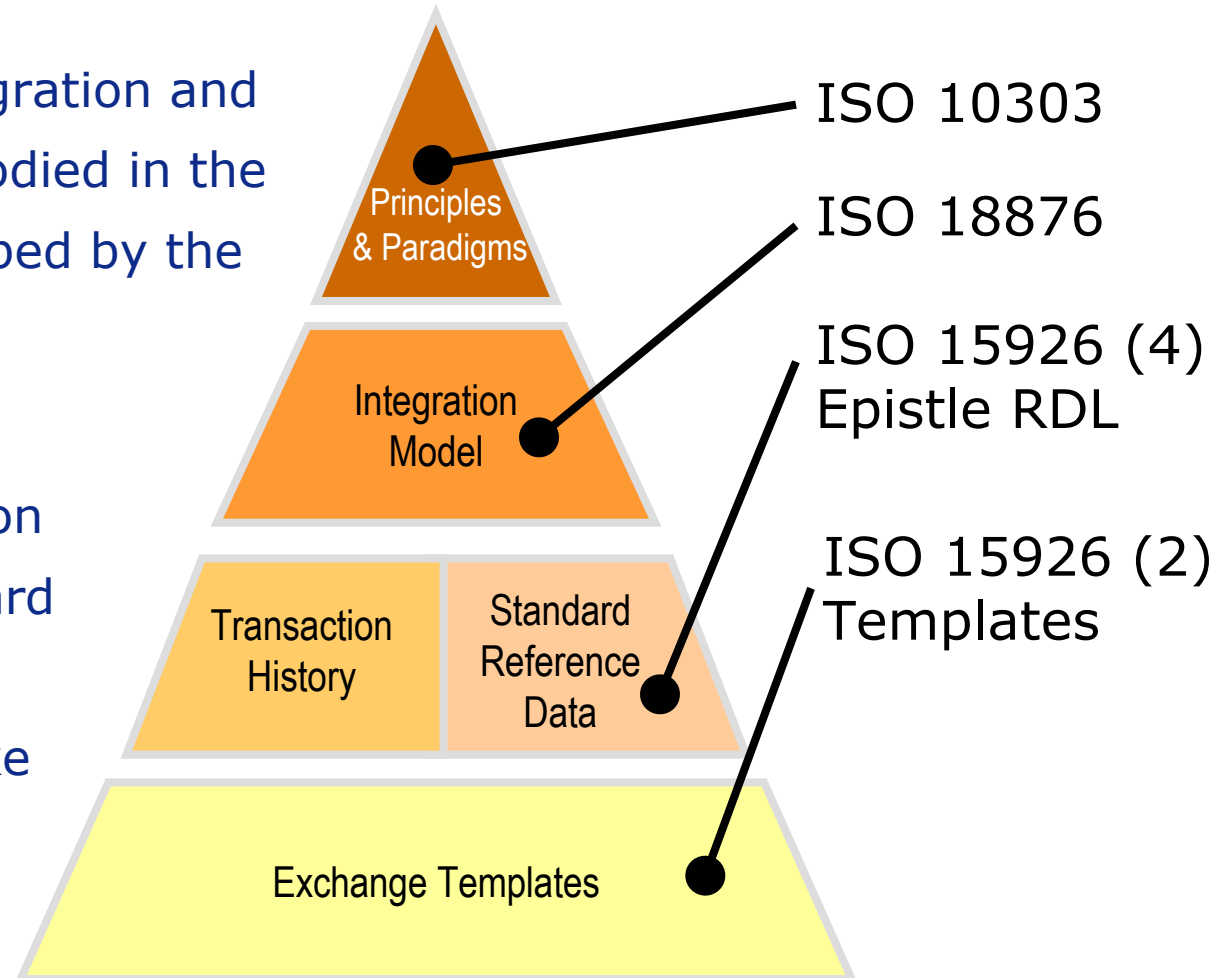
ISO 15926 Architecture

# Integrated and standards based approach

Best practice in integration and exchange is embodied in the standards developed by the process industry

Without an integration model and standard reference data, interfaces can take 3-4 times more to create

(Prof Matthew West 2003)



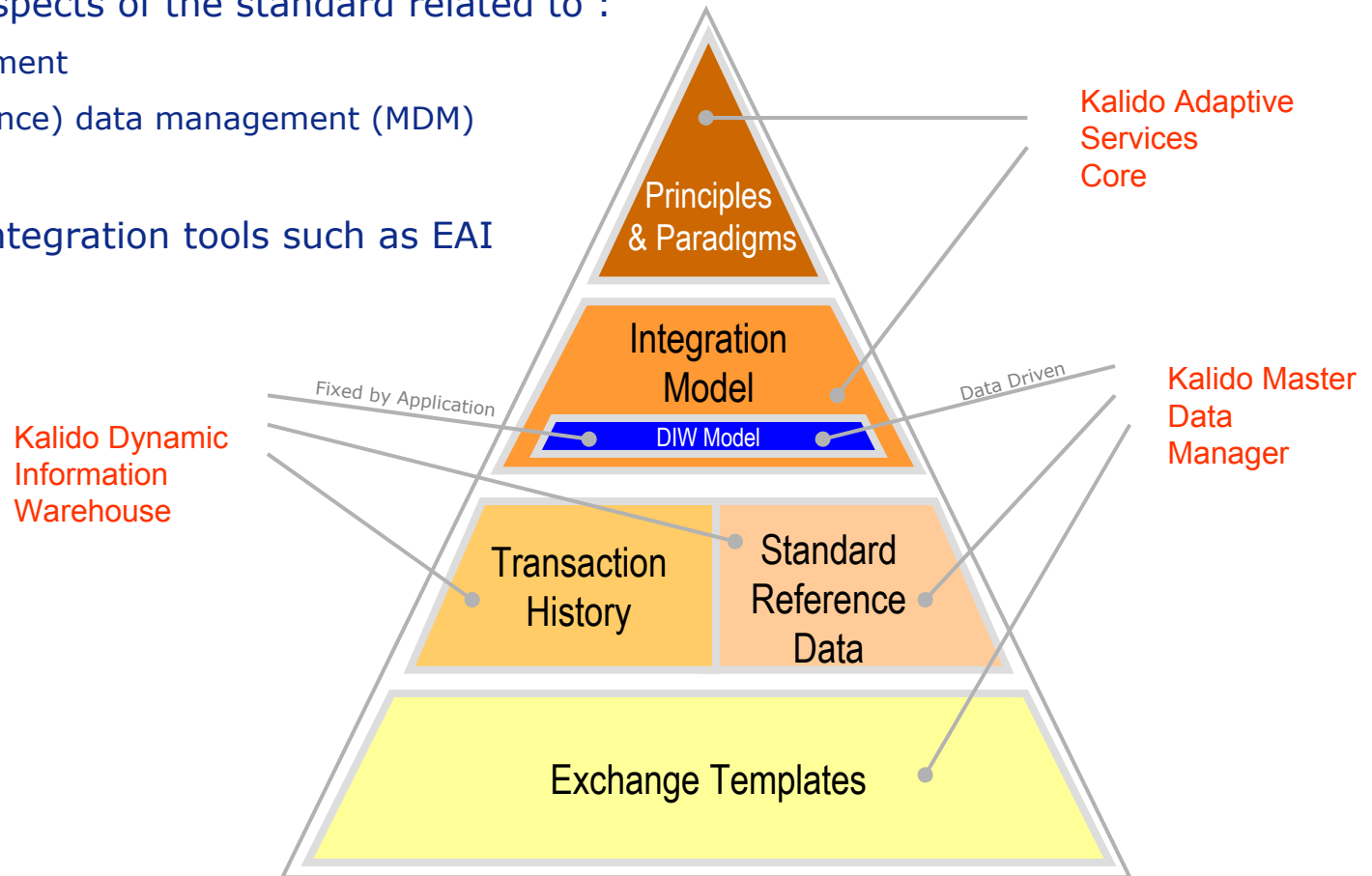
# Integrated and standards based approach

Kalido uses the best practice approach to deliver integration

The products are based on the same paradigms and principles and implement aspects of the standard related to :

- Model Management
- Master (Reference) data management (MDM)
- Analysis (DIW)

The MDM can drive integration tools such as EAI



## Integration Models – ISO 10303

A standard for the computer interpretable representation and exchange of product data.

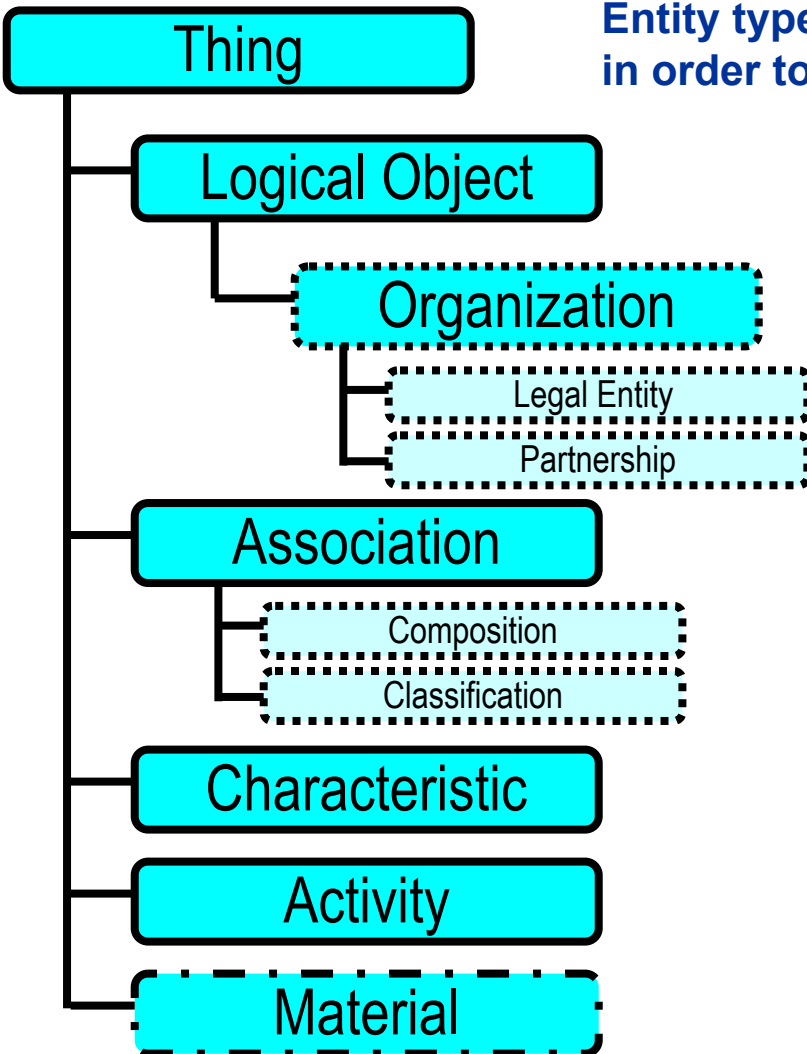
A mechanism for describing product data throughout the lifecycle of a product, independent of any particular system.

The nature of the description makes it suitable not only for neutral file exchange but also a basis for implementing and sharing product databases and archiving.

# Integration Model Principles ISO 15926

**Entity types should be part of a subtype/super type hierarchy in order to define a universal context for the model.**

- The answer lies in a subtype/ super type hierarchy of generic entity types.
- This hierarchy must be universal in its context, which is guaranteed if it consists only of generic entity types, even if it is not complete.
- Care must be taken to ensure that the level of detail is appropriate.
  - Too high a level of subtyping means that entity types could mean almost anything.
  - Too low a level of subtyping means that you get lost in the detail, or it is too specific
- With this approach, the model is layered
- The model is best handled as data

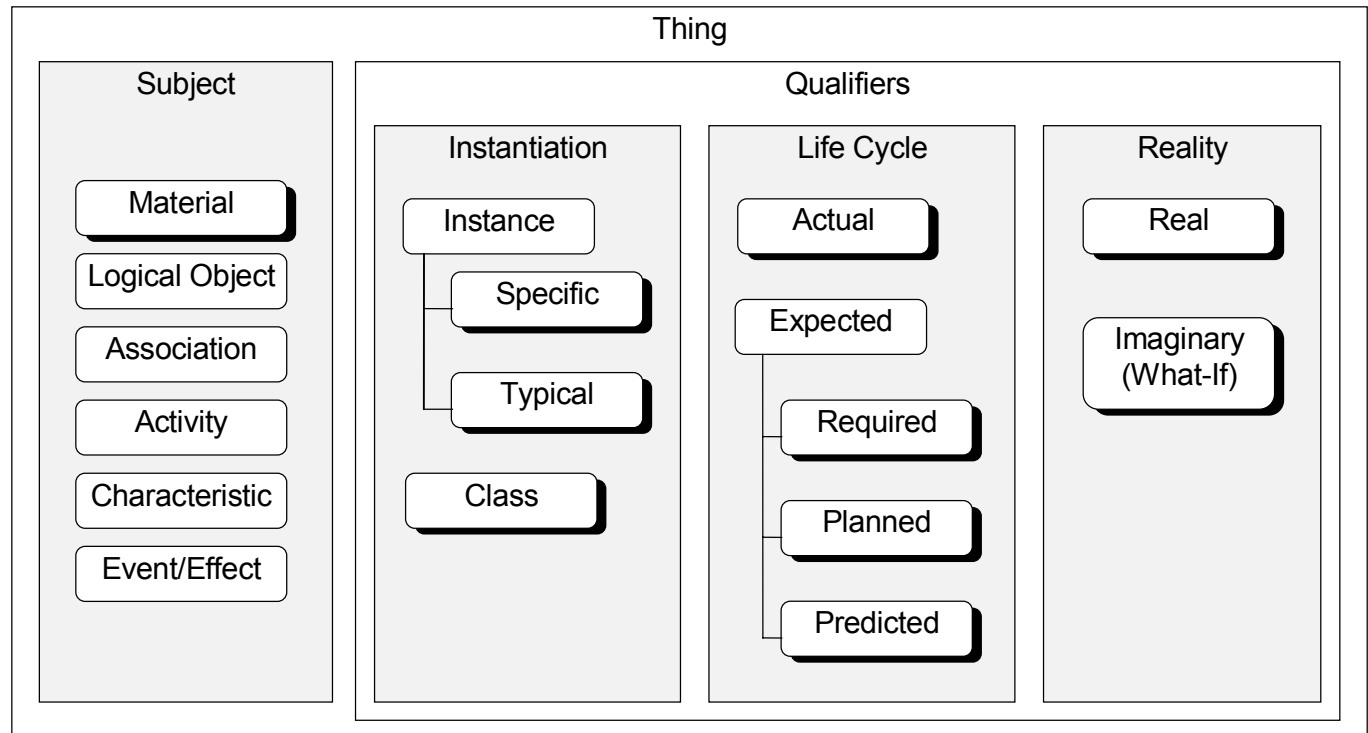


# Integration Models - The Generic Entity Framework

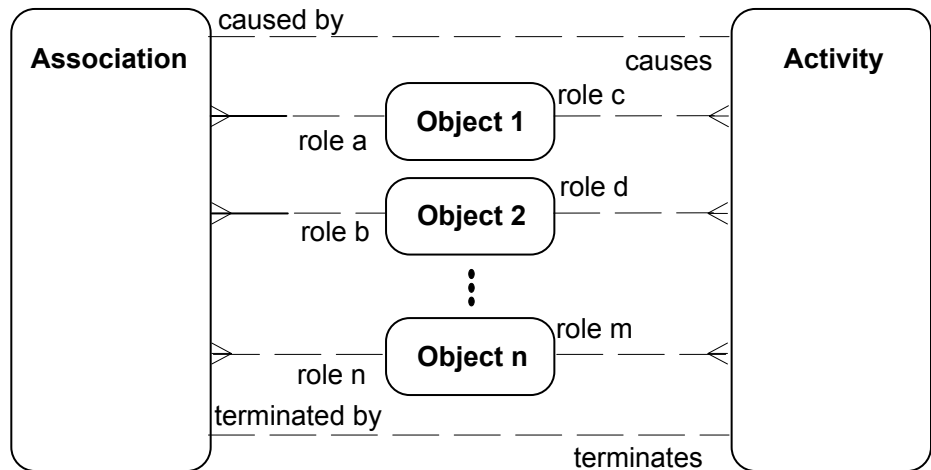
A series of templates and entity definitions that can be used to construct business data models.

Orthogonal subtypes of Things with qualification by:

- ◆ Instantiation
- ◆ Life Cycle
- ◆ Reality



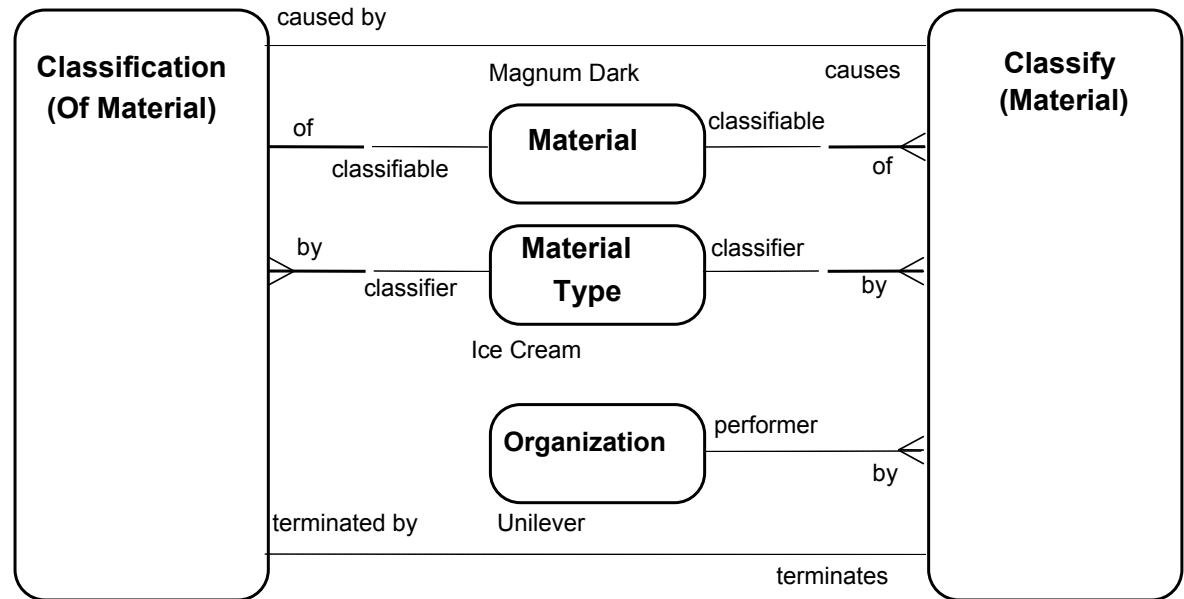
# Integration Model Fundamentals



## Data Model Templates for objects and relationships

Examples for major relationship and activity types.

Introduces **Classification** as a generalized model for identifying a "type" for data, rather than the standard relational approach of classification by **Entity Type**.



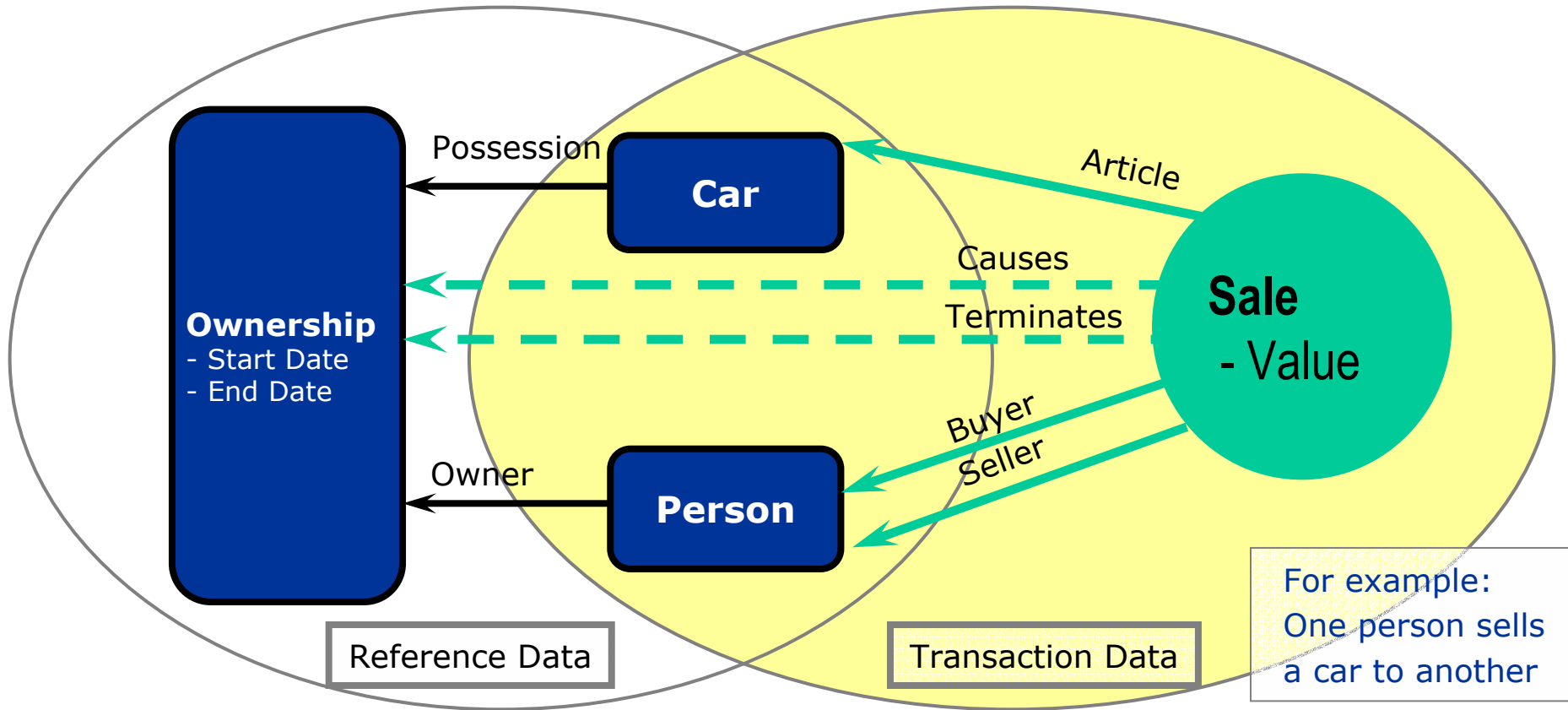
Magnum Dark is classified as Ice Cream

Unilever classified Magnum Dark as Ice Cream on 1/1/1990

# The relationship between transaction and reference data

Most reference data is caused by transactions (business events), and all transactions create reference data

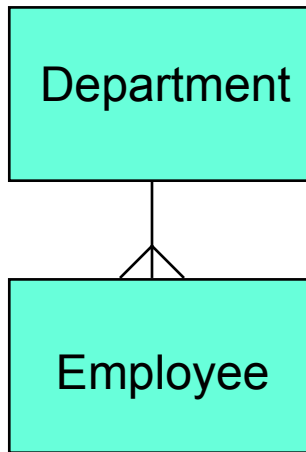
- > Often models omit one or other part
- > "One persons reference data is another's transaction data"



# Another view: Dependence vs. ER

This ensured that things that could exist in their own right independent of other things.

It also ensured that things which were really dependent on other things would be represented as such...

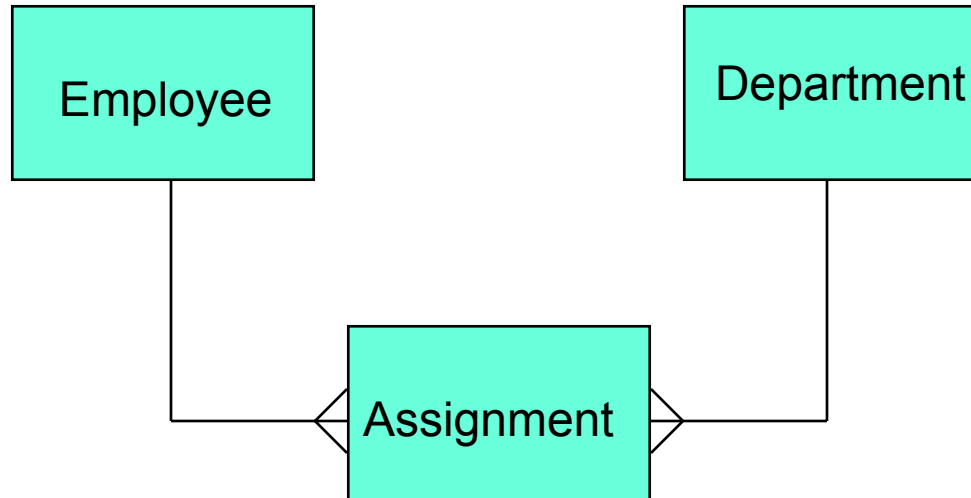


This model says that an employee is dependent on the existence of his or her department.

Unfortunately it is not true. Employees can exist without a department or even be in two departments

# The dependence model

The dependence model would look like this



- An employee can exist without a department and
  - A department can exist without any employees
- BUT...**
- An assignment can only exist if there is a department and an employee

# Other Learnings That Fed The Research

1. Multiple Classifications
2. Attributes
3. Life Cycle

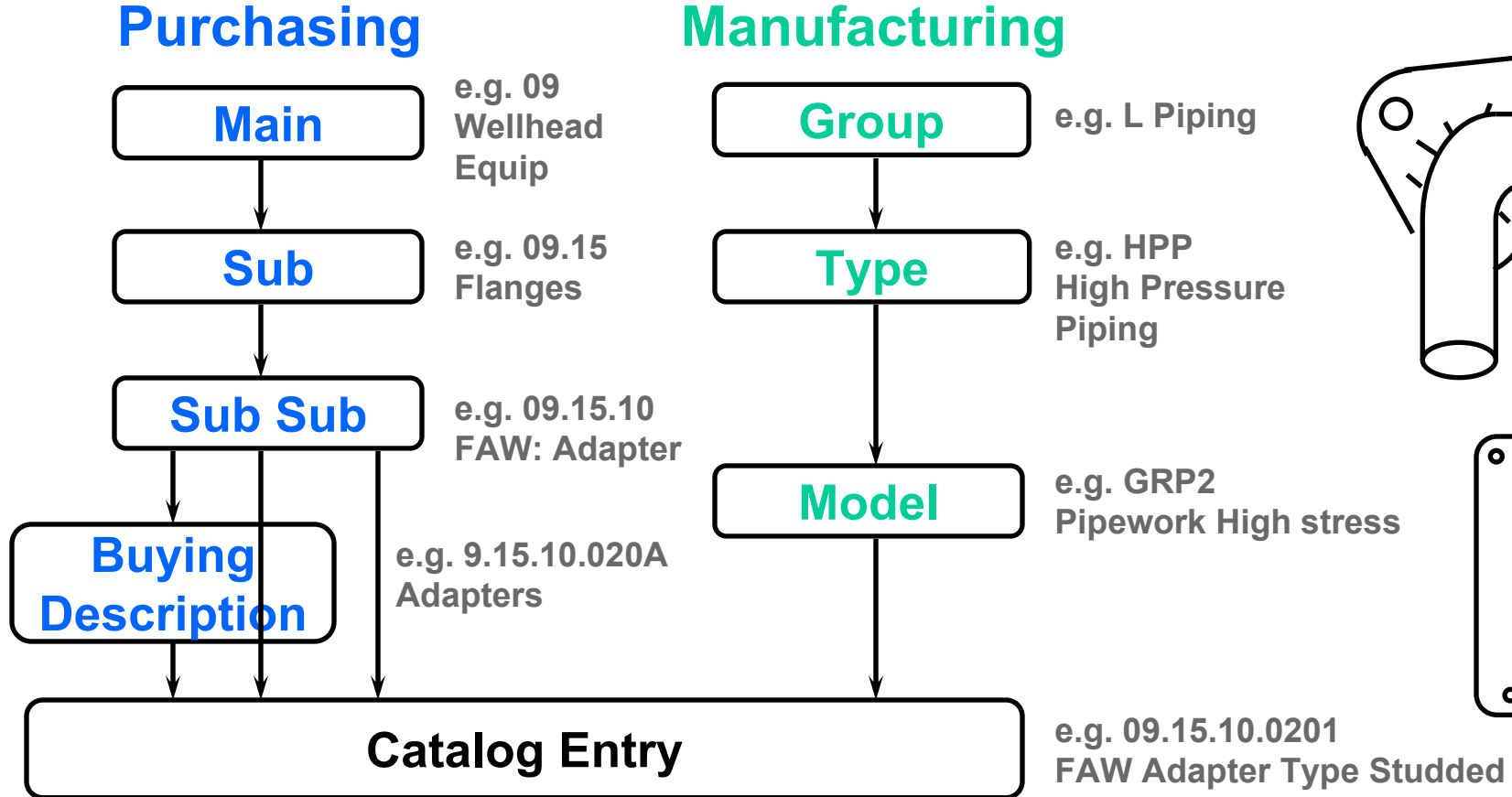
# Classification

We worked next on the engineering maintenance system and a purchasing system in the early 1990's

We discovered that:

**Things can be classified in many ways**

# Multiple Classifications



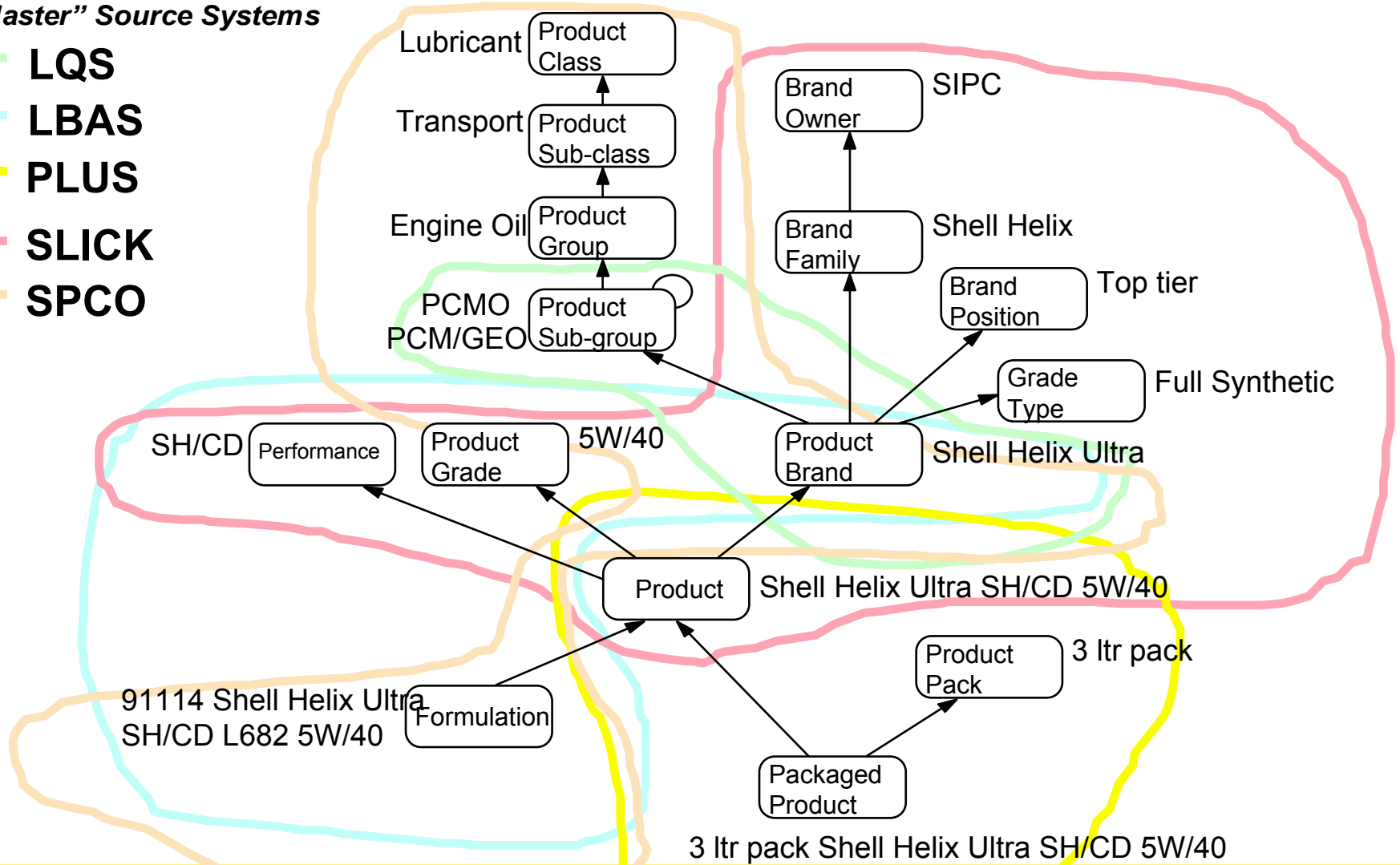
Different ways of classifying the same thing.  
Manufacturing had a different classification scheme from Purchasing

# Example Integration Model

## Shell Lubricants Database Model - 1995

### "Master" Source Systems

- LQS
- LBAS
- PLUS
- SLICK
- SPCO



Multiple master systems merged with an integration data model. Implemented in a Kalido prototype during 1995 using the Generic Storage model

# How to Standardize

Changing to a single standard is resisted because although the schemes are often similar:

- > There are subtle variations in the groupings that different aspects of the business require
- > People who use these classifications are used to them and don't want to change
- > Systems that use them are costly to change

A generic design enables all hierarchies to be supported simultaneously. Merged businesses don't have to adopt one structure immediately. Standardization can then be carried out as business needs are identified, rather than as the system dictates.

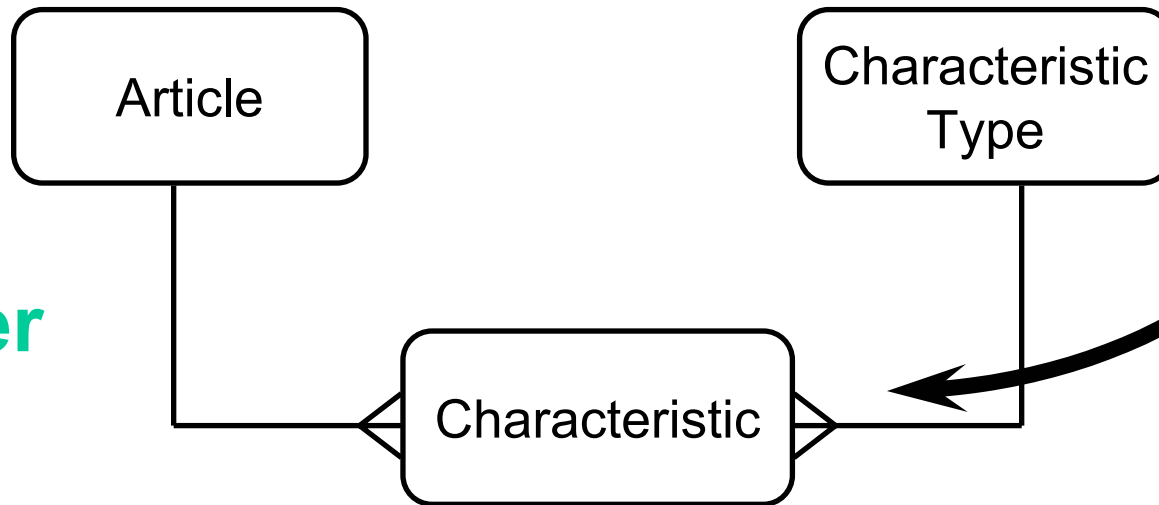
# How Many Attributes

A typical table

How Many Attributes Does a Sales Article Have?

Article Number	Article Name	Length	Style	Quality	Max Wattage	Color	etc
CP 862	Ladies Skirt	27 ins	Straight			Blue	tc
GB 084	Table Lamp		Antique		60 Watts	Bronze	
XT 031	Tool Kit			Stainless			

Any  
Number



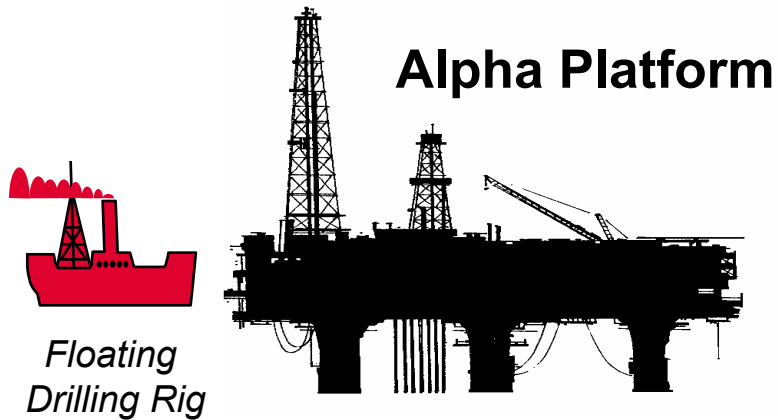
In 1994 we were working on a  
Cost Management System  
for North Sea platforms

## Everything kept changing

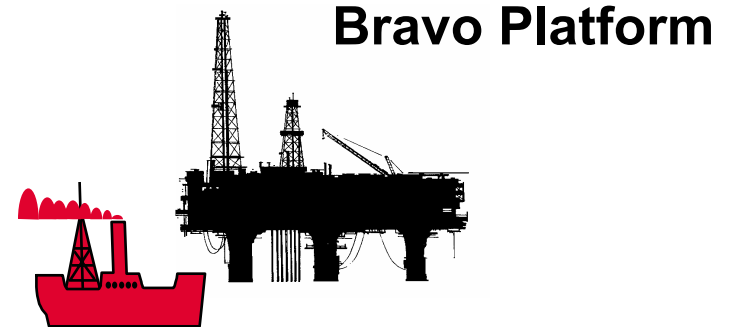
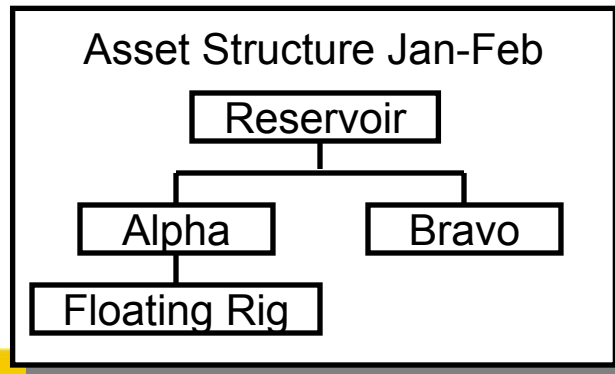
We discovered  
Everything has a life

# Changing Hierarchies

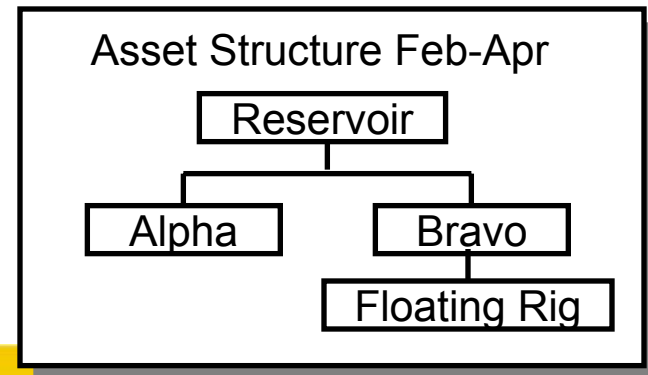
Costs of Activities Change during the Normal Course of Business Operations



**Costs of drilling  
allocated to Alpha  
from Jan to mid Feb**

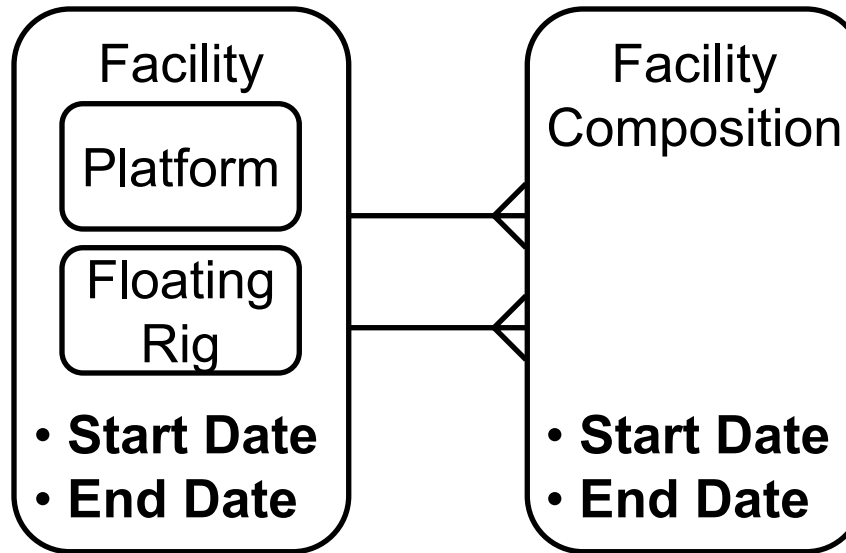


**Costs of drilling  
allocated to Bravo  
from mid Feb to April**



# Handling Changing Hierarchies

**This is true  
for all  
Entities**



**Include a start and end date on the association to define the period for which the association is valid**

**This is true  
for all  
Hierarchies**

**A Facility Composition table**

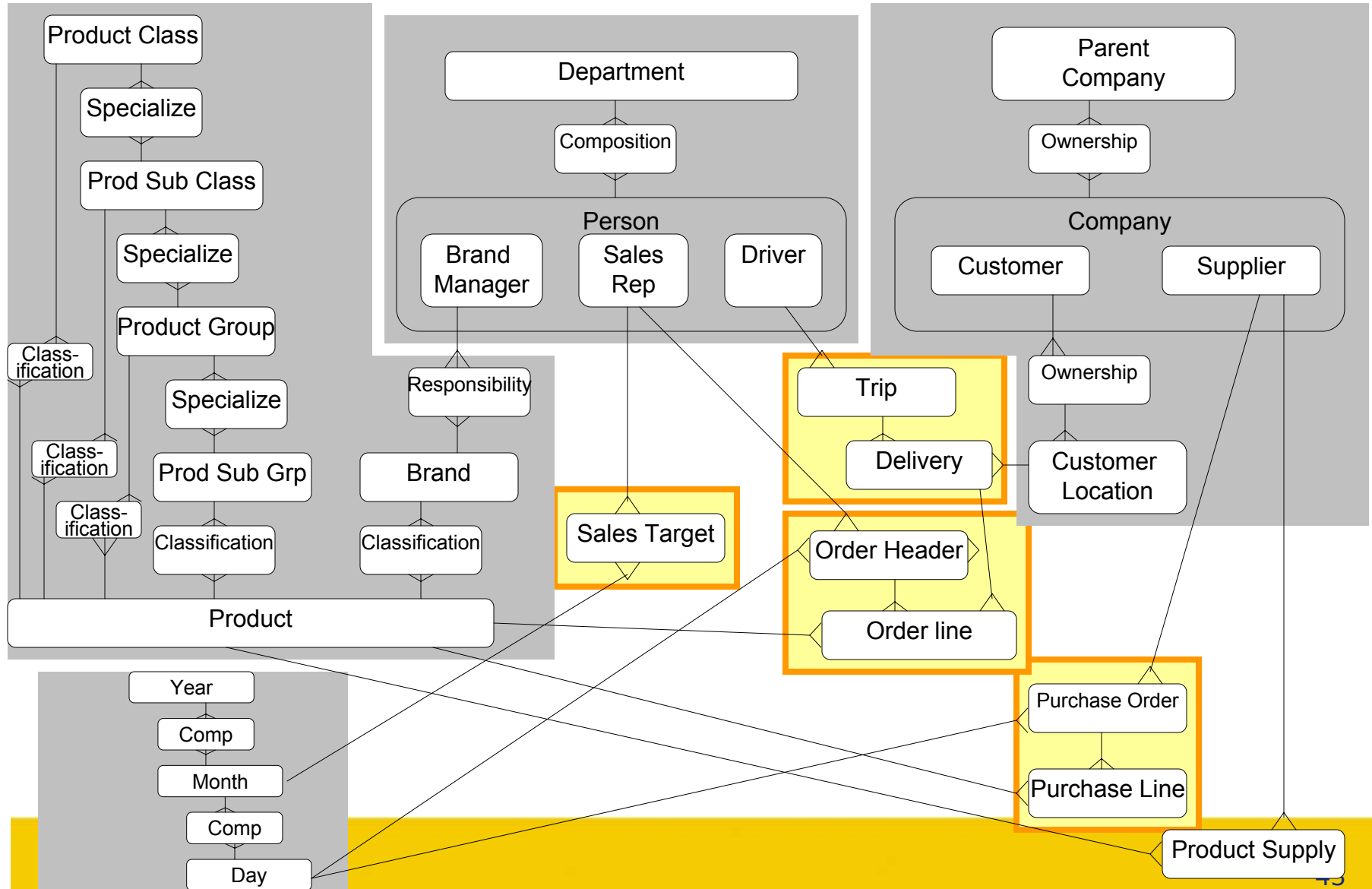
Parent Facility	Child Facility	Start Date	End Date
Alpha Bravo	Floating Rig Floating Rig	1st January 16th February	15th February 30th April

# Principles for Conceptual Models

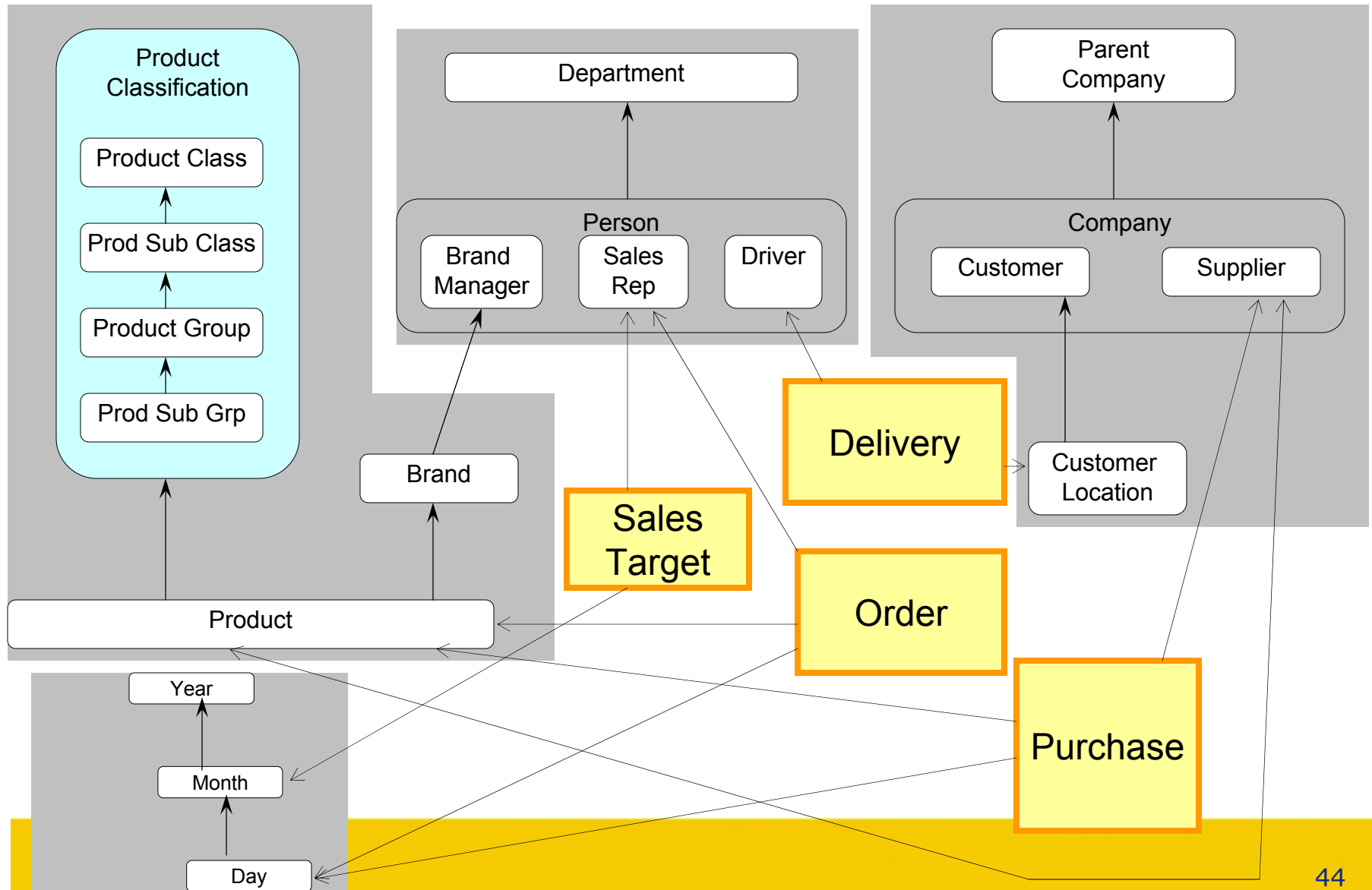
1. Entity types should represent the underlying nature of an object.
2. Entity types should be part of a framework which provides a universal context.
3. Activities and associations should be represented by entity type
4. Relationships should only be used to express the involvement of something with an activity or association.
5. Attributes should be suspected of representing relationships to other entity types.
6. Entity types should have a single attribute as their primary unique identifier.



# Warehouse Design

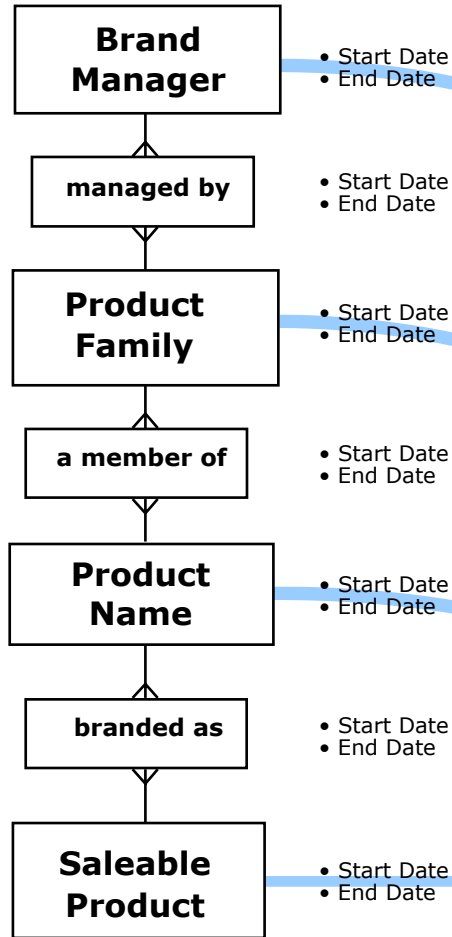
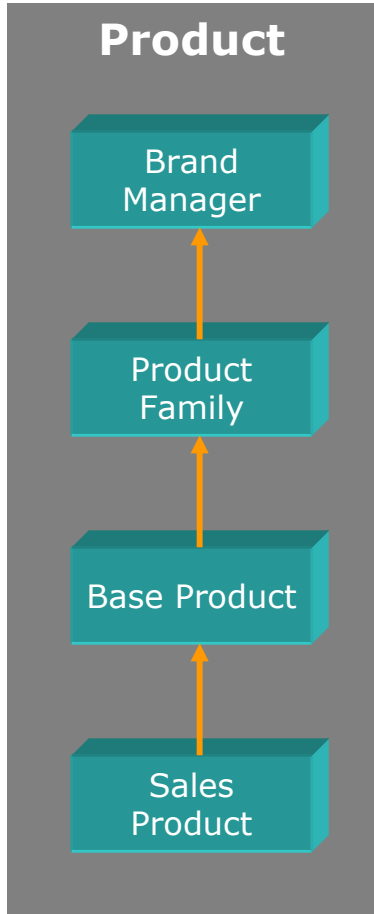


# Warehouse Design



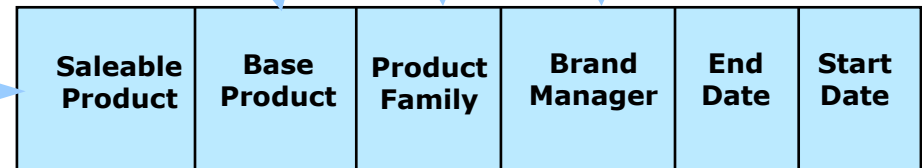
# Flexibility Plus High Performance

## >> Logical Model



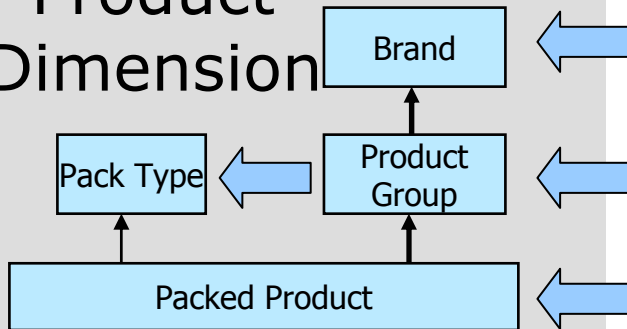
## >> STAR SCHEMA

>> Automatic and Highly Optimized Generation and Maintenance of Hybrid Slowly Changing Dimensions

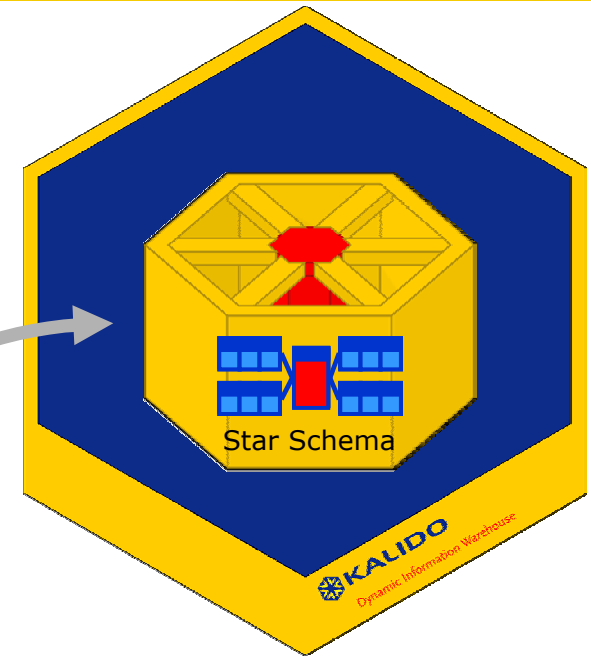


# Star Schema Generation

## Product Dimension



- Reusable tables with multi granularity data
- Time Effectiveness on all data
- Optimum efficiency using minimum snow flaking
- Handles cross dimensional joins with ease



Change of Packaging in 1997

Mapping Table

All	Brand	Product Group	Pack Type	Packed Product	Start Date	End Date
Super Deluxe	Super Deluxe	~	~	~	1990	
Ice Cream	Super Deluxe	Ice Cream	~	~	1990	
½ Litre Tub	~	~	½ Litre Tub	~	1990	
Economy Pack	~	~	Economy Pack	~	1990	
Choc Delight	Super Deluxe	Ice Cream	½ Litre Tub	Choc Delight	1990	1997
Mint Delight	Super Deluxe	Ice Cream	½ Litre Tub	Mint Delight	1990	
Choc Delight	Super Deluxe	Ice Cream	Economy Pack	Choc Delight	1997	

# Conclusions

Generic models give the benefit of ER models

- > with all history
- > without duplication of data
- > are very stable when requirements change

There are techniques to make it perform as well as any star schema design

So, you get the best of both worlds

**(This is the foundation of Kalido. It has been used to integrate information in many Global 2000 companies. Examples in the following presentation.)**

Questions?

Stephen Pace  
stephen.pace@kalido.com