



**UGS**

*Transforming the  
process of innovation*

# Teamcenter Engineering Basics and Structure Management Overview

Amy Strucko

Teamcenter Product Management

PLMWorld 2006

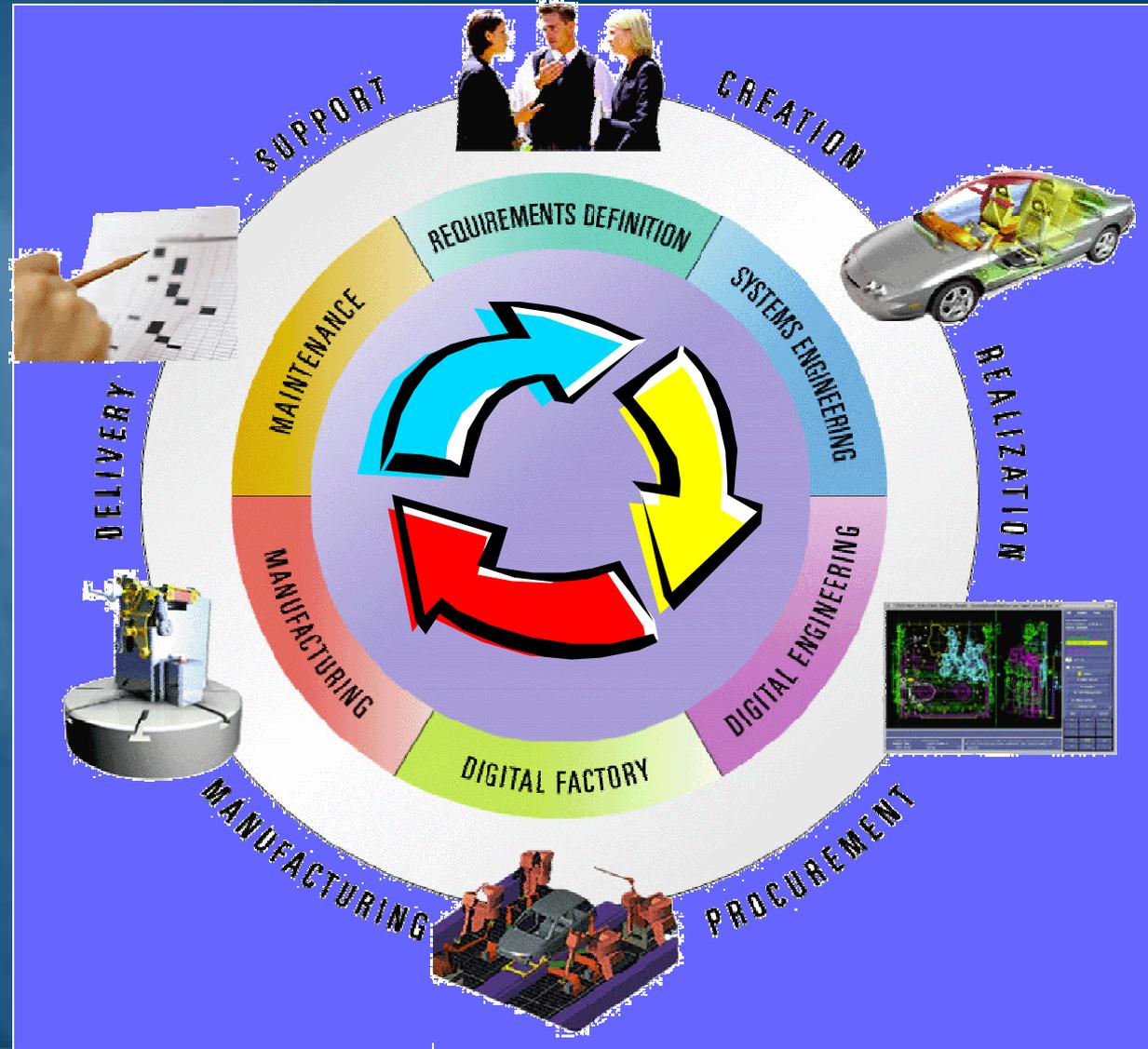
**TEAMCENTER**

- Object Management
  - Revisioning
  - Part Management
- Structure Management:
  - Effectivity Management / Incremental Change
  - Configuration Context
  - Variability Management
- Supersedure / BOM Compare
- Where Used / Where Referenced

# Product Lifecycle Management

“A Product is the sum of its life cycle process, and not its manufactured parts.”

*B. Huthwaite*



# Object Management

- ▶ Items are objects representing the logical control items for the Customer

- ▶ Revision control
- ▶ Workflow
- ▶ Release Status
- ▶ Unique IDs
- ▶ Manage datasets
- ▶ Structure

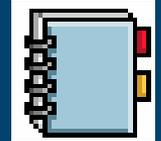
- ▶ Datasets represent bulk data
  - ▶ Datasets can encapsulate one or more files
  - ▶ Files point to physical file system



Document



Module



Document



Assembly



Change Request



Component



PDF File



Word File



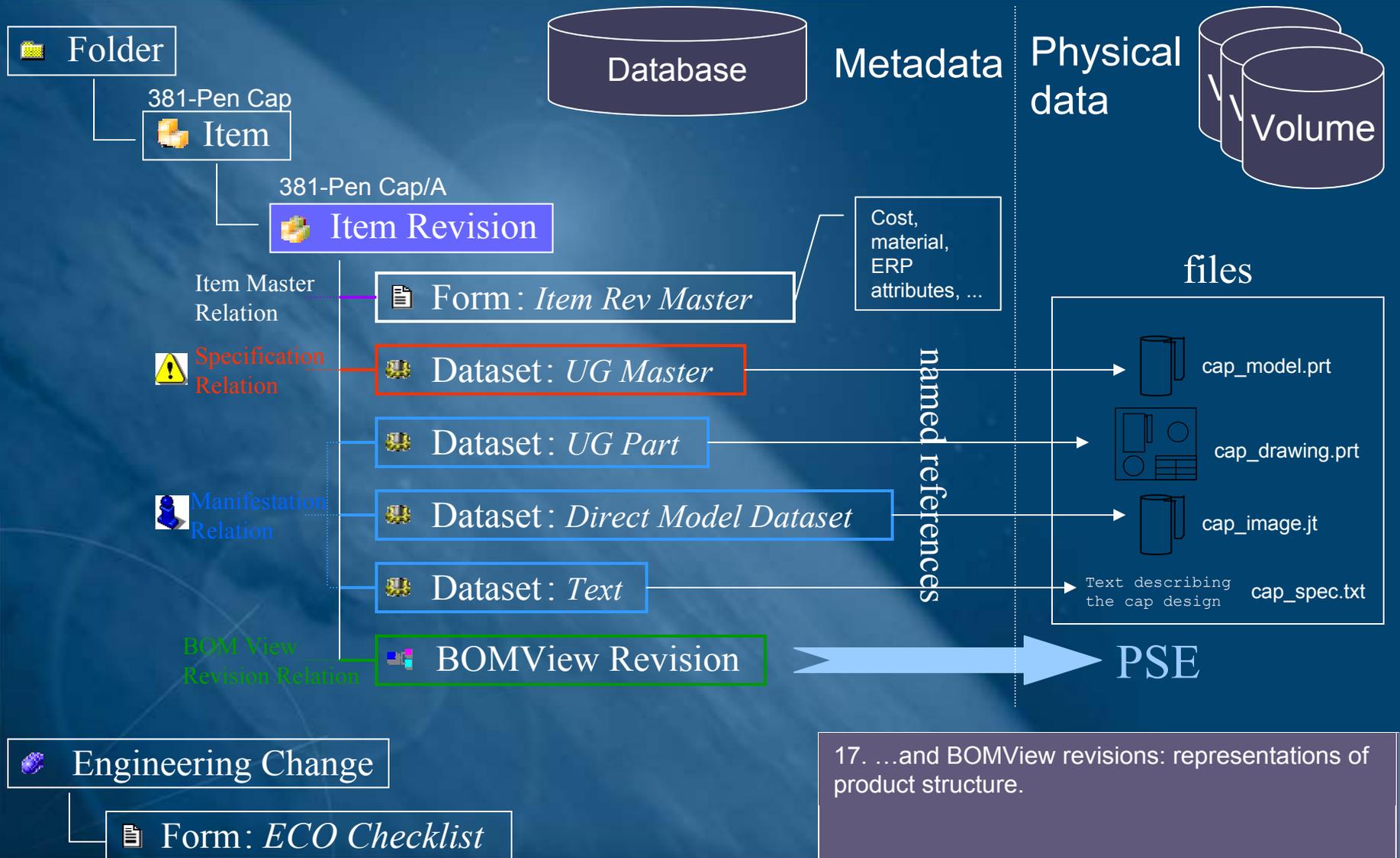
CAD Drawing



# Revisioning

- ▶ A key element to Configuration Management is the ability to manage multiple **iterations** of an item as it evolves over time. Some iterations may be important to preserve and may trigger key business processes. Others are work in progress and can be discarded
- ▶ Iteration schemes include the ability to check-in and out an item therefore producing multiple versions and to freeze/release and then revise an item at key business stages producing multiple **revisions**
- ▶ In a product structure, the validity of each revision is defined according to configuration rules such as effectivity. The structure can be filtered for the appropriate revisions by applying Configuration Context (e.g. revision effectivity, release status, latest/working etc)

# Teamcenter Engineering Data



17. ...and BOMView revisions: representations of product structure.

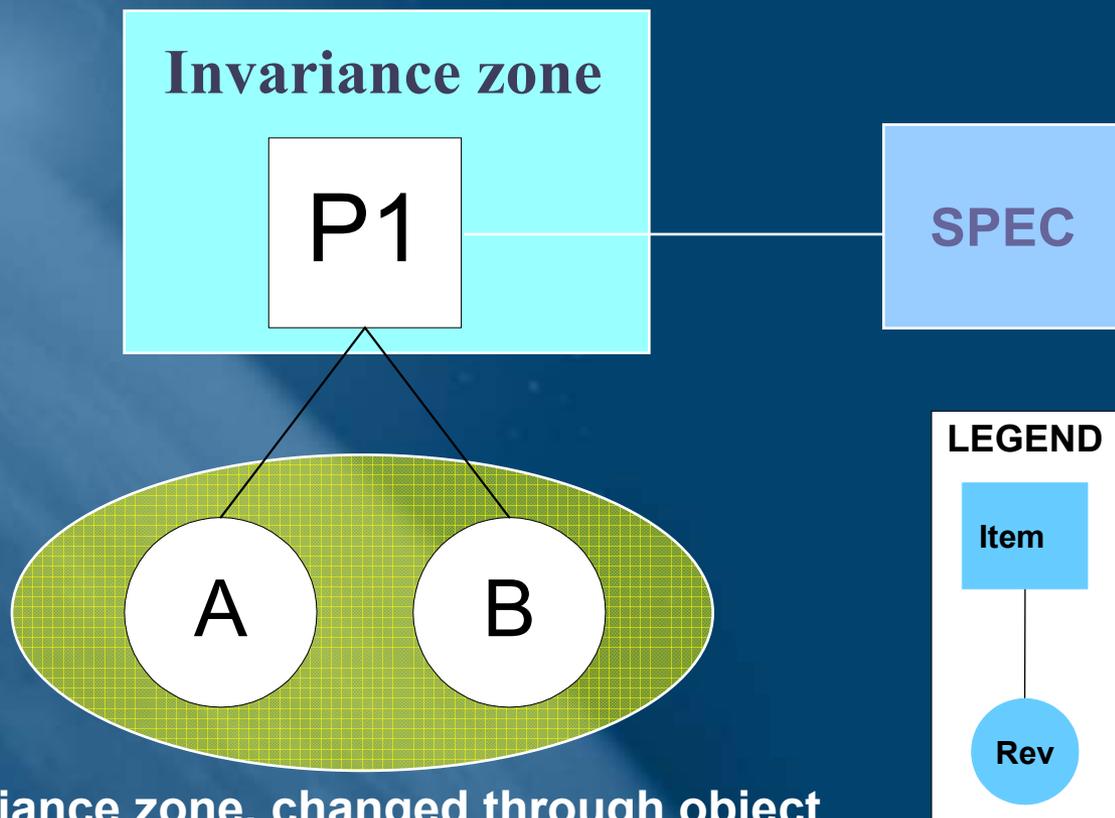
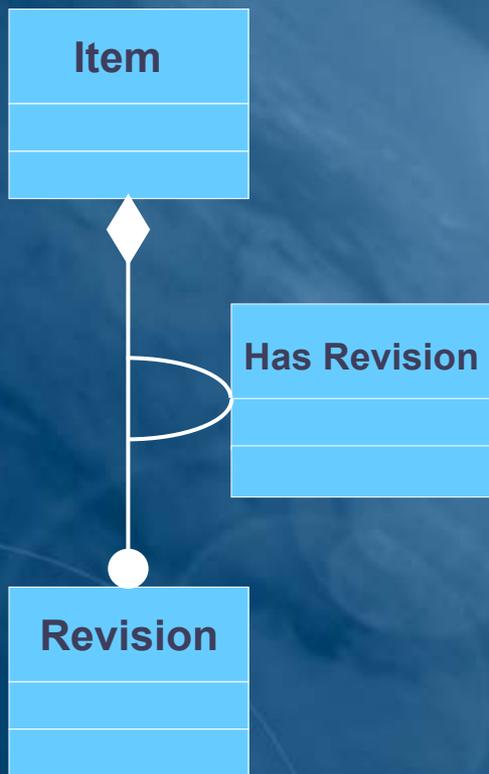


# Item and Item Revision

- ▶ For many Items, an interface for the object is defined (for example form, fit and function) and separated into its own Item Object. The Item applies globally and may have multiple associated Revision Objects each conforming to the same master definition
- ▶ Many kinds of Business Objects in Teamcenter are modeled with Items and Revisions
  - ▶ E.g. parts, documents, change requests
- ▶ Items insulate related objects from
  - ▶ **WIP (work in process) dynamics**
  - ▶ **interchangeability**



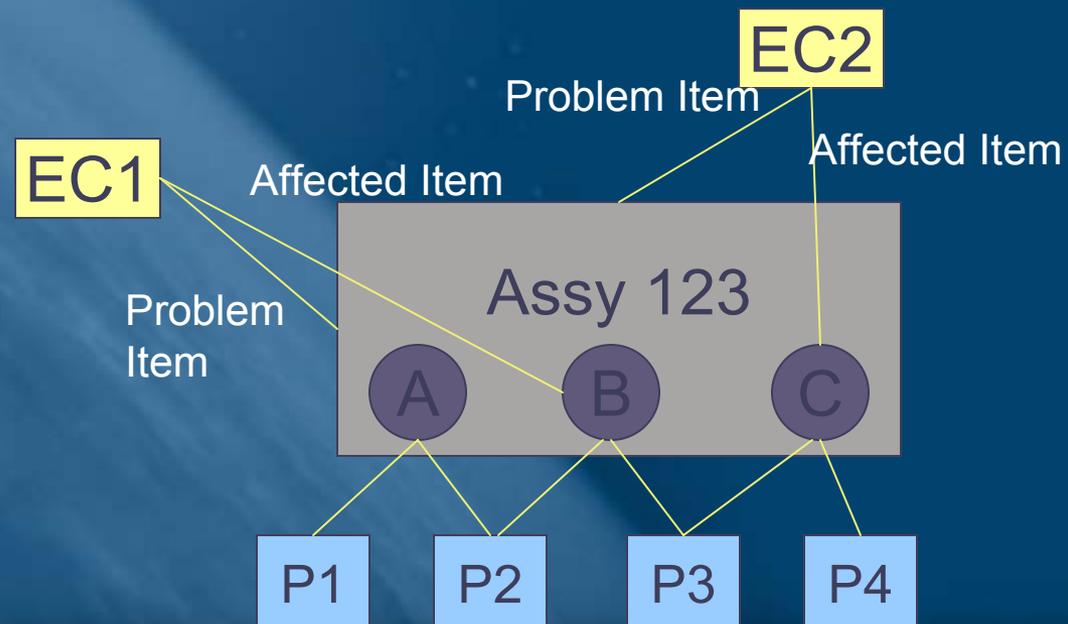
# Item-Revision Model



Variance zone, changed through object acquisition for definition

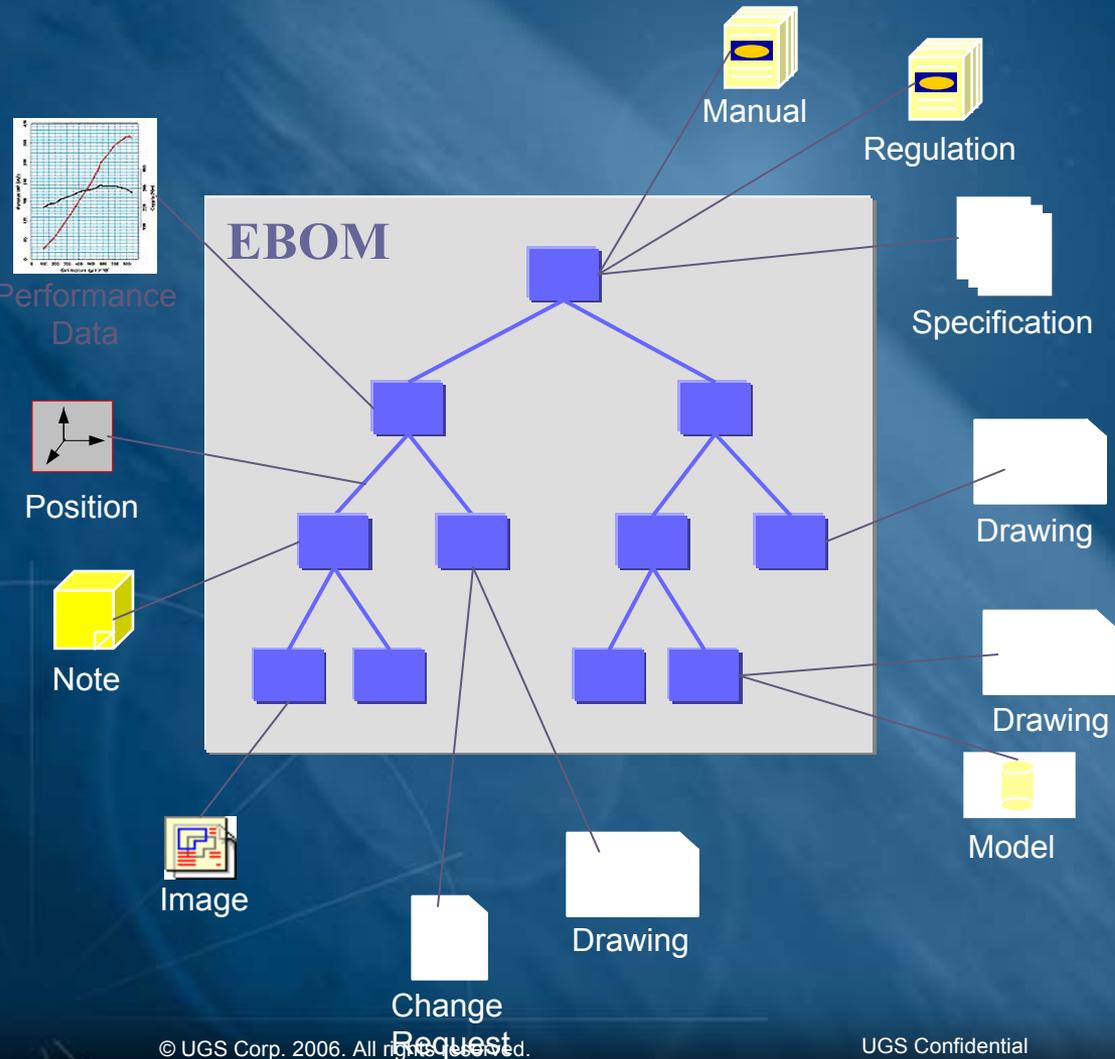
# Revision Based Configuration Management

- ▶ **Revision based configuration management** assumes all previous changes accumulate into the latest revision – there is no representation of the change itself, just the result of applying it to the latest configuration
- ▶ Later we will discuss an alternative approach to revision based configuration management called **Incremental Change**



# What is Product Structure?

- ▶ An integrating mechanism for lifecycle data management



- It's constantly changing throughout the product's life
  - Change in customer demand
  - Engineering changes
  - Work in progress
  - Release states
- It needs to be configuration managed to ensure:
  - User access to accurate information
  - Collaboration
    - As shared information changes users can be notified

# Customer Demands for Product Structure



- Collaborative Design
- Digital Mock-Up
- Multi-CAD Integration

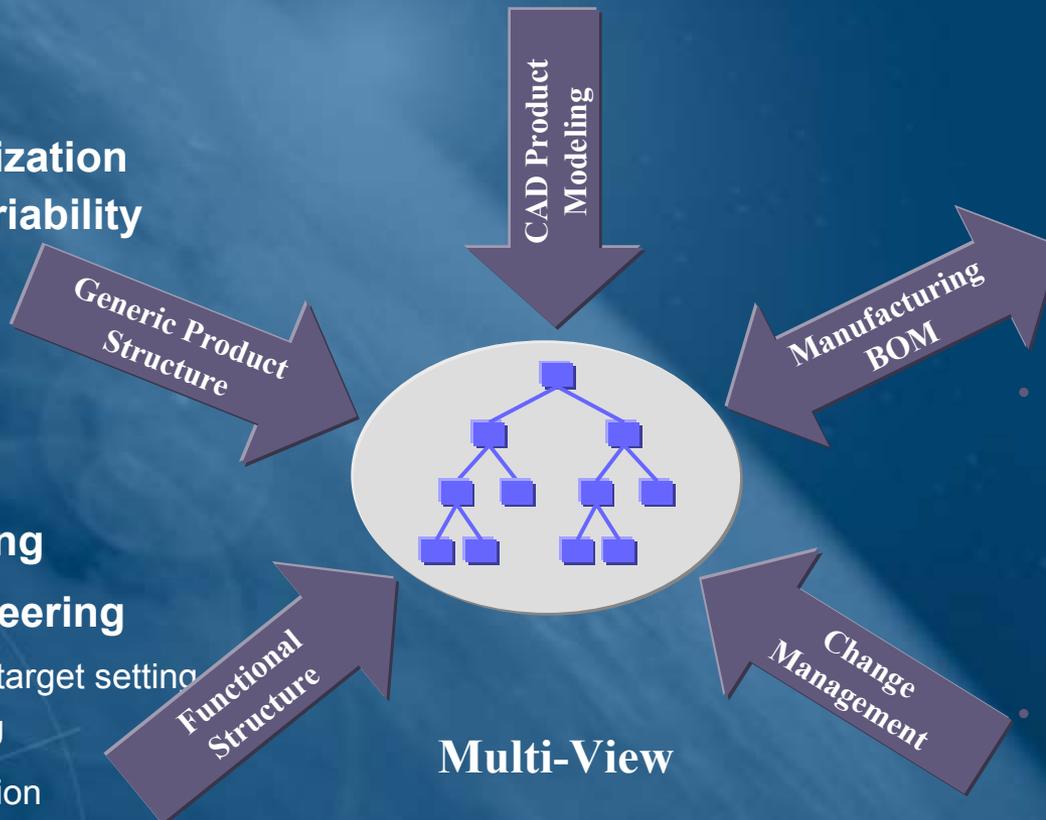
- Mass Customization
- Design for Variability

- MRP Integration
  - MBOM
  - Cost
- BOM Accountability
- Single Source of Product Data

- Product Planning
- Systems Engineering

- Performance target setting
- Concept DMU
- Space allocation
- Reuse

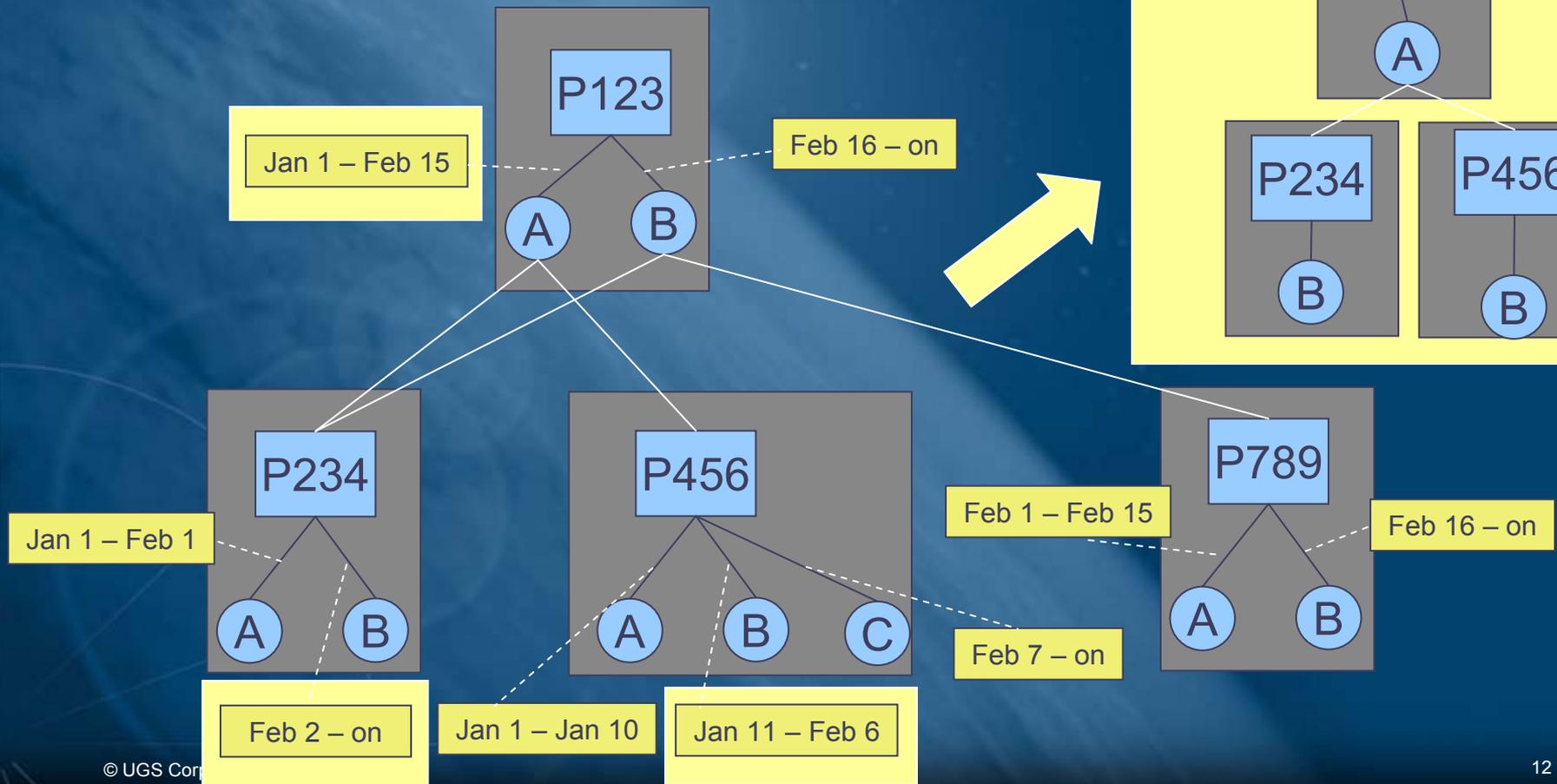
• (new program startup)



- Managing Product Change
  - ECO

# Revision Effectivity

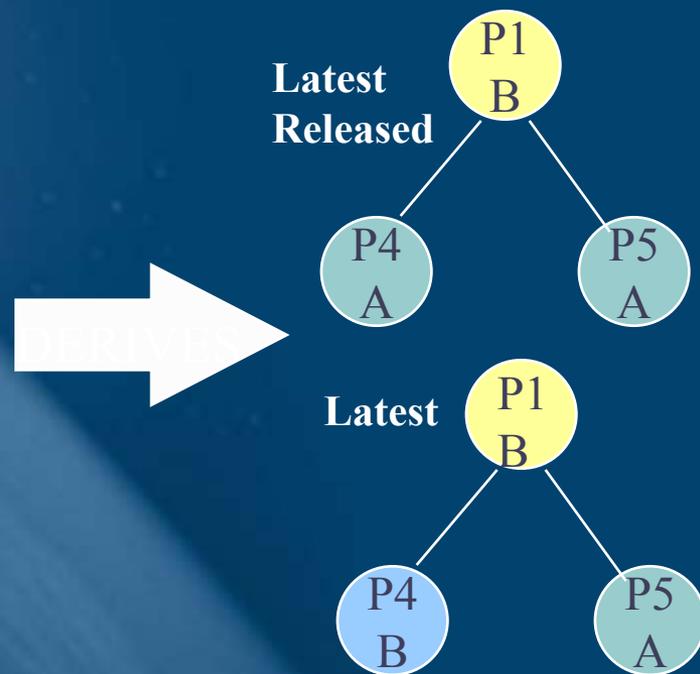
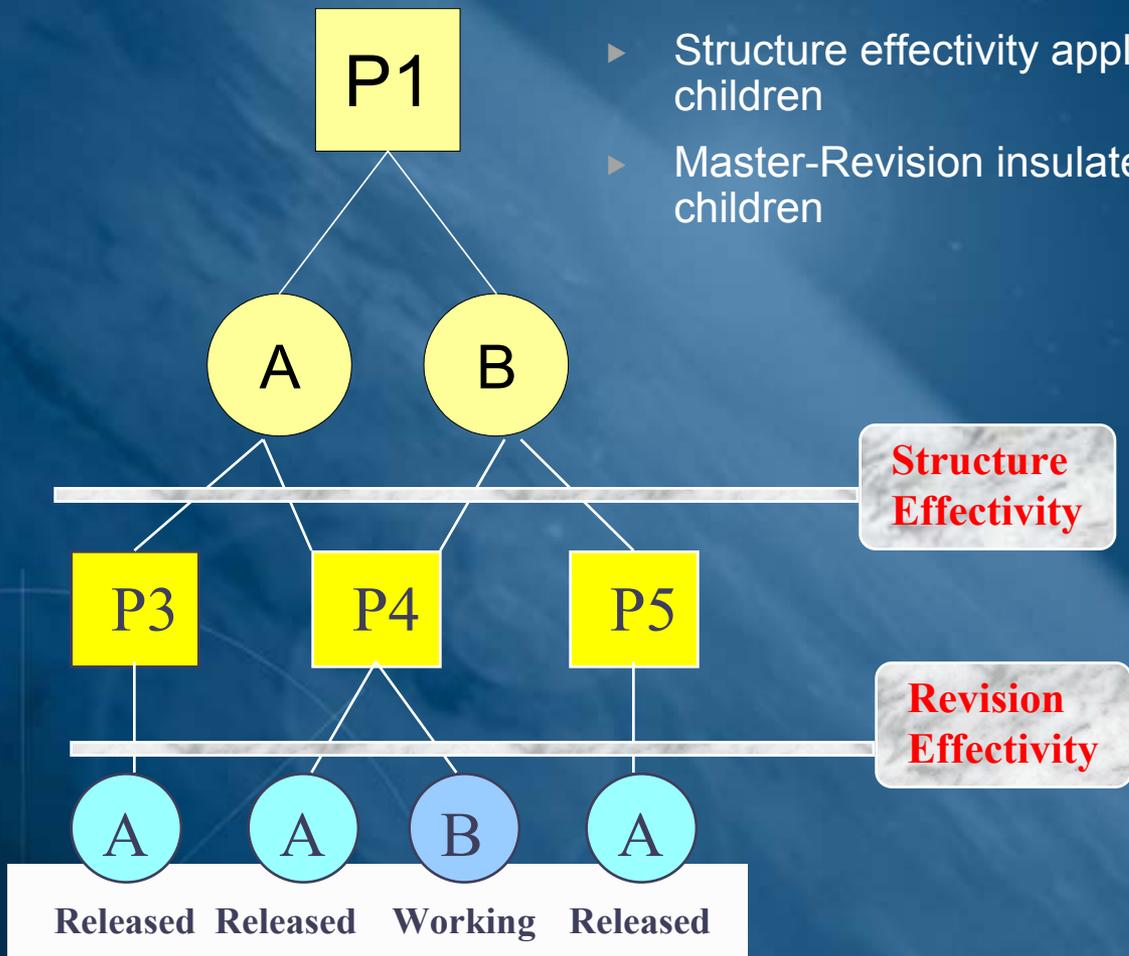
- Each revision of an Item can be qualified by an effectivity range





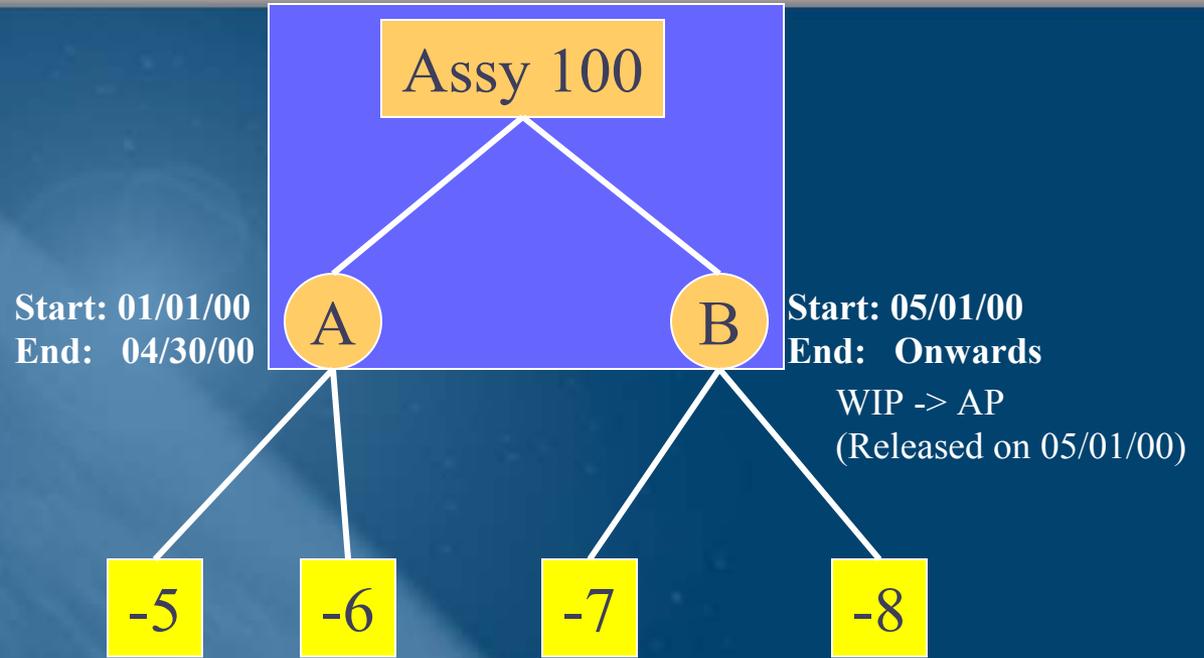
# Structures and Effectivity

- ▶ Revision effectivity applies between an Item and its revisions
- ▶ Structure effectivity applies between a parent and its children
- ▶ Master-Revision insulates parent from updates to its children

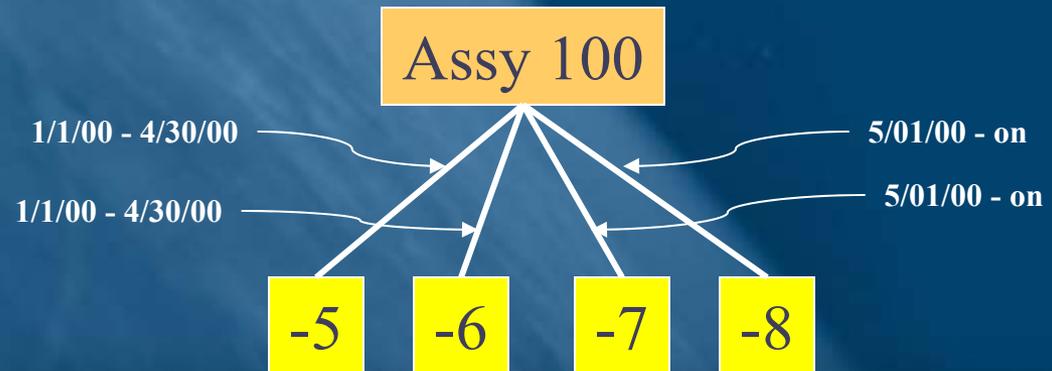


# Revision Effectivity vs Structure Effectivity

## Revision Effectivity

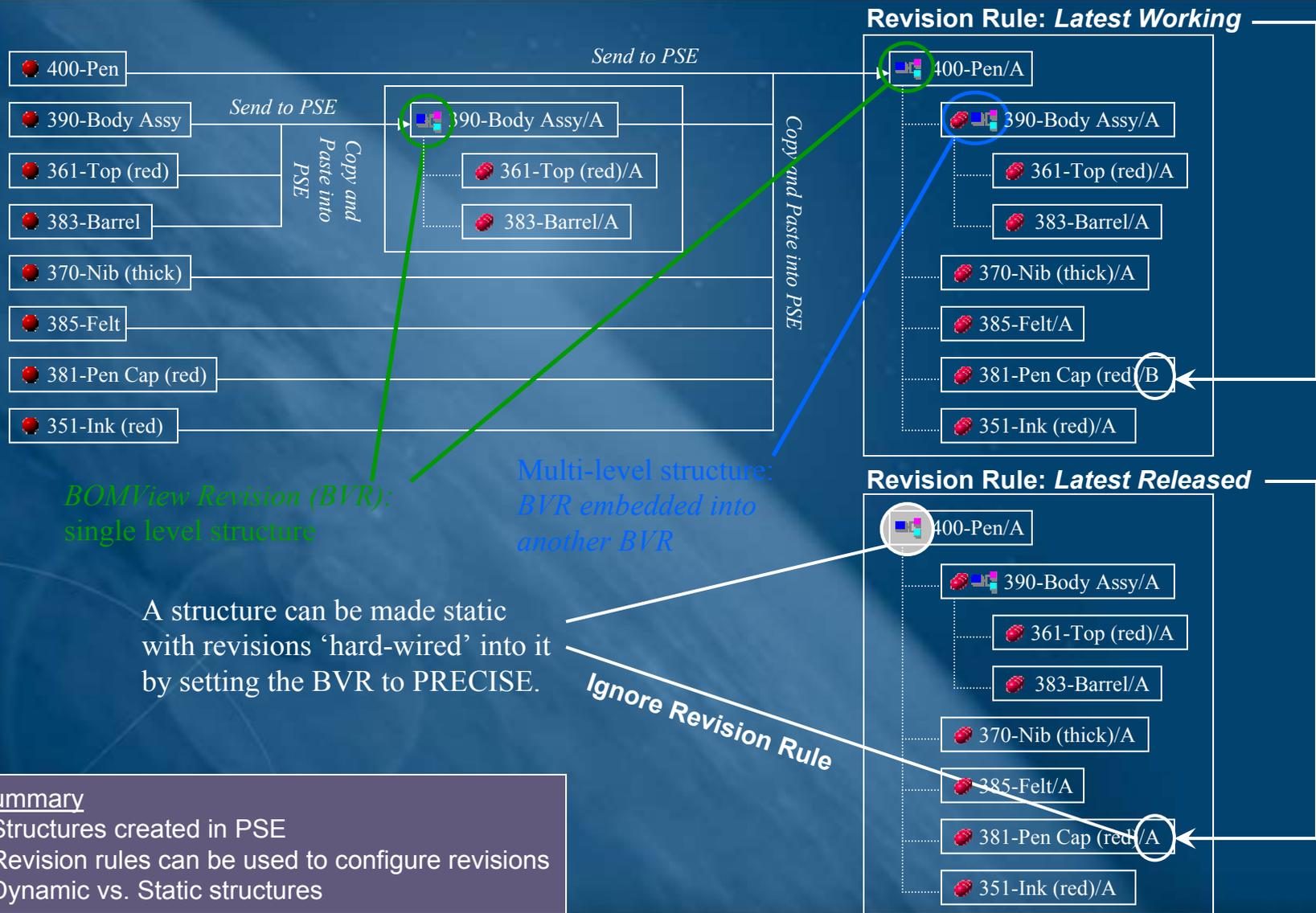


## Structure Effectivity





# Product Structure and Revision Rules



A structure can be made static with revisions 'hard-wired' into it by setting the BVR to PRECISE.

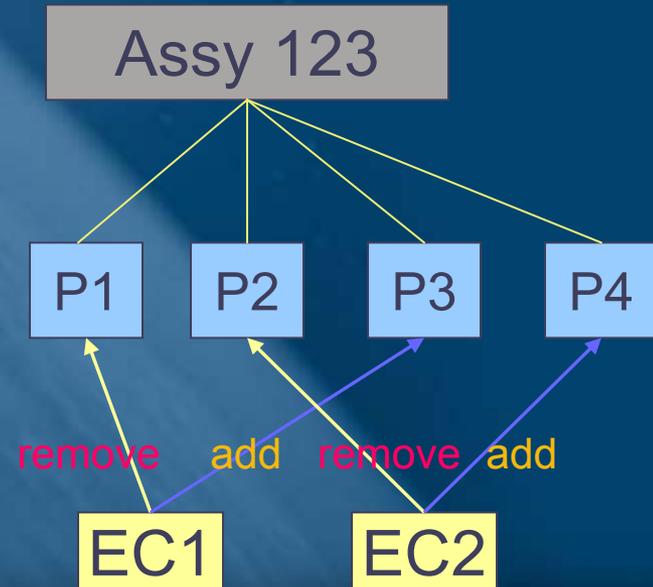
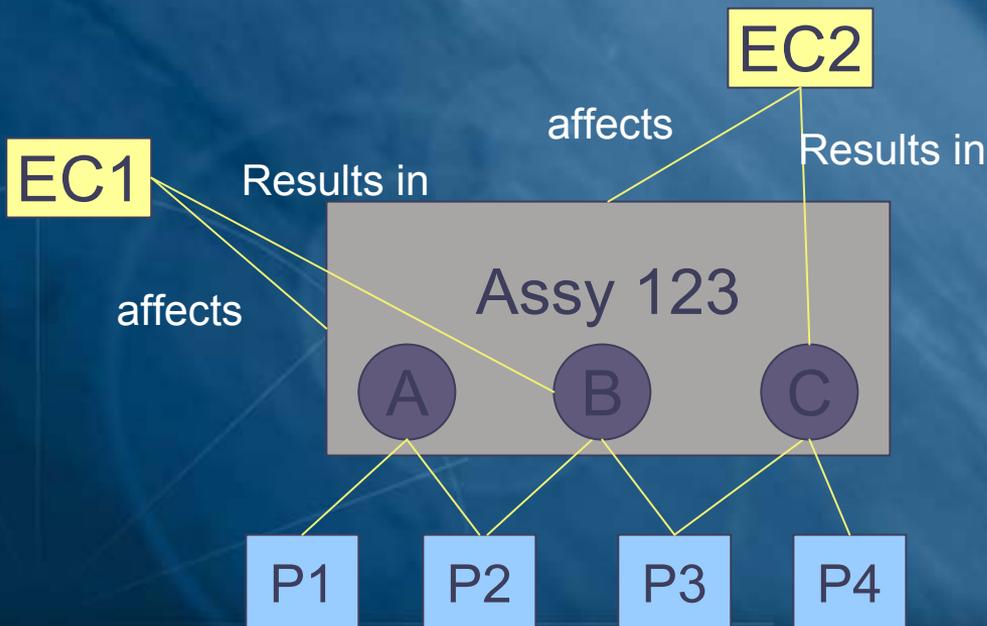
- Summary**
- Structures created in PSE
  - Revision rules can be used to configure revisions
  - Dynamic vs. Static structures



# Revision configuration vrs Incremental Change

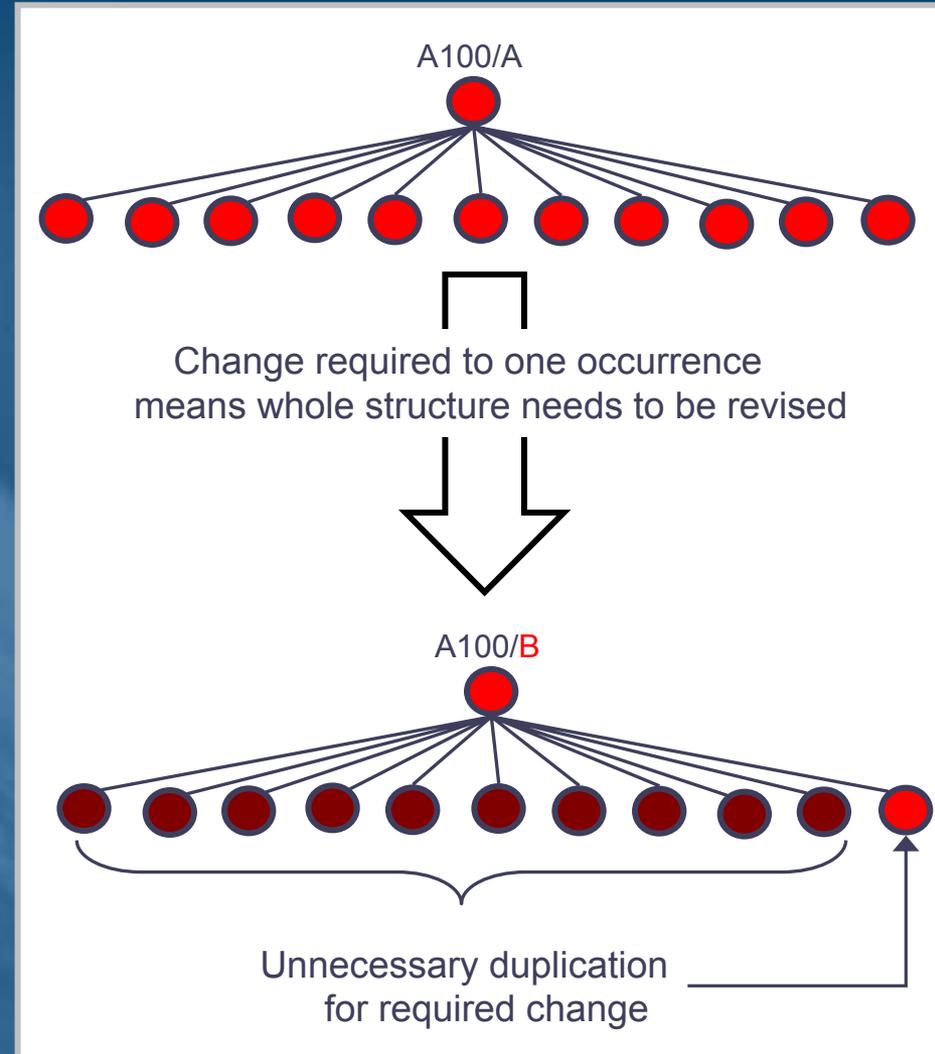
- ▶ The traditional **revision based configuration management** assumes all previous changes accumulate into the latest revision
- ▶ There is no representation of the change itself, just the result of applying it to the latest configuration

- ▶ In contrast, an **incremental change** is a change that defines the evolution of an object or group of objects by documenting the **differences** between two states of the object
- ▶ Supports changes to structure, attributes, related data, etc



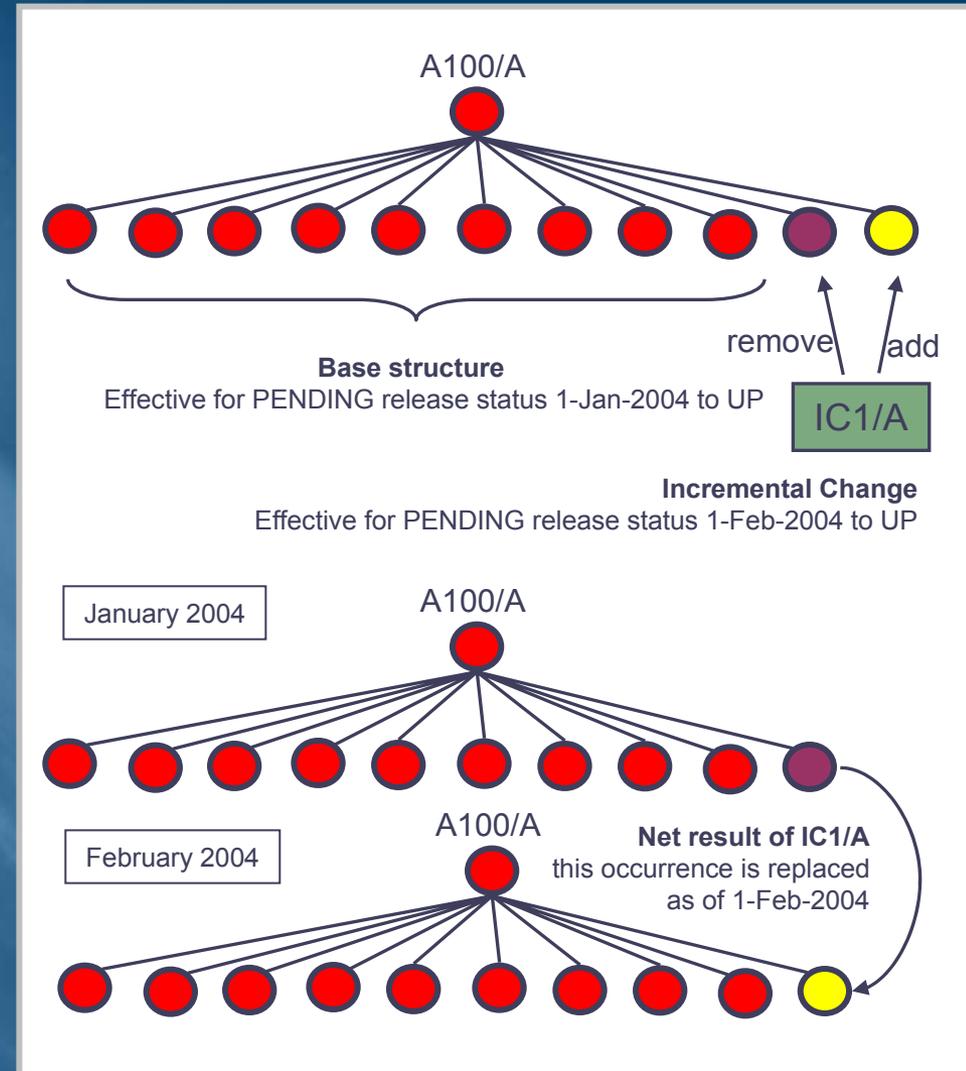
# Revision Configuration of Flat Structures

- ▶ Traditionally, when a change to a structure is required, then the parent is revised
  - ▶ This copies the entire structure to the new rev
  - ▶ If the structure has many children, then for a simple change, there is a lot of unnecessary duplication of data
- ▶ Large, flat structures tend to have many changes applied to them
  - ▶ Lots of revisioning, which is difficult to manage
- ▶ The user may also want to track the changes made and associate data with the change.



# Incremental Change (IC) Basics

- ▶ IC is a collection of 'Adds' and 'Removes' that together represent a change. These are incremental change elements
- ▶ Change is configured into a structure using a revision rule
  - ▶ Adds are displayed
  - ▶ Removes are filtered out
- ▶ Changes can be carried out where they are needed without revising entire structure





# Configuration Context

- ▶ A **Configuration Context** represents the collection of qualification parameters needed to configure a representation
- ▶ **Configuration Recipes** define the conditions under which product lifecycle data is valid
- ▶ Allows the application of certain selection criteria when navigating a product structure
- ▶ Note: Combinations of these parameters can be applied together e.g. “latest released” or “effective date 2/1/02 in Engineering View”

Configuration Recipe	Sample Configuration Context parameters
Revision	Latest/ All
Effectivity	Specified date or unit
Maturity State	e.g. Working or Released
View	e.g. Design or Manufacturing
Variant Conditions	SOS (Selected Option Set) e.g. LX Model with Manual Transmission, Air Conditioning and Cruise control
Incremental Change	e.g. ECO1 and ECO3 have been incorporated but not ECO2

# Variant Configuration

Pen Variant: **Red** with **Thick** Nib

**Option Defaults:**  
Values of options for default model

**Rule Checks:**  
Value y of option b is incompatible with value x of option a

400-Pen/A

390-Body Assy/A

361-Top (red)/A

383-Barrel/A

370-Nib (thick)/A

385-Felt/A

381-Pen Cap (red)/A

351-Ink (red)/A

**Variant Rule:**  
Red with Thick Nib:  
Colour = **Red**  
Nib = **Thick**

Generic Pen Structure

400-Pen/A

390-Body Assy/A

361-Top (red)/A

360-Top (blue)/A

383-Barrel/A

370-Nib (thick)/A

371-Nib (thin)/A

385-Felt/A

381-Pen Cap (red)/A

380-Pen Cap (blue)/A

351-Ink (red)/A

350-Ink (blue)/A

**Options:**  
Colour (**Red** or **Blue**)  
Nib (**Thick** or **Thin**)

Colour=**Red**

Colour=**Blue**

Nib=**Thick**

Nib=**Thin**

Colour=**Red**

Colour=**Blue**

Colour=**Red**

Colour=**Blue**

**Variant Conditions:**  
circumstances under which a line is shown

Pen Variant: **Blue** with *Thin* Nib

400-Pen/A

390-Body Assy/A

360-Top (blue)/A

383-Barrel/A

371-Nib (thin)/A

385-Felt/A

380-Pen Cap (blue)/A

350-Ink (blue)/A

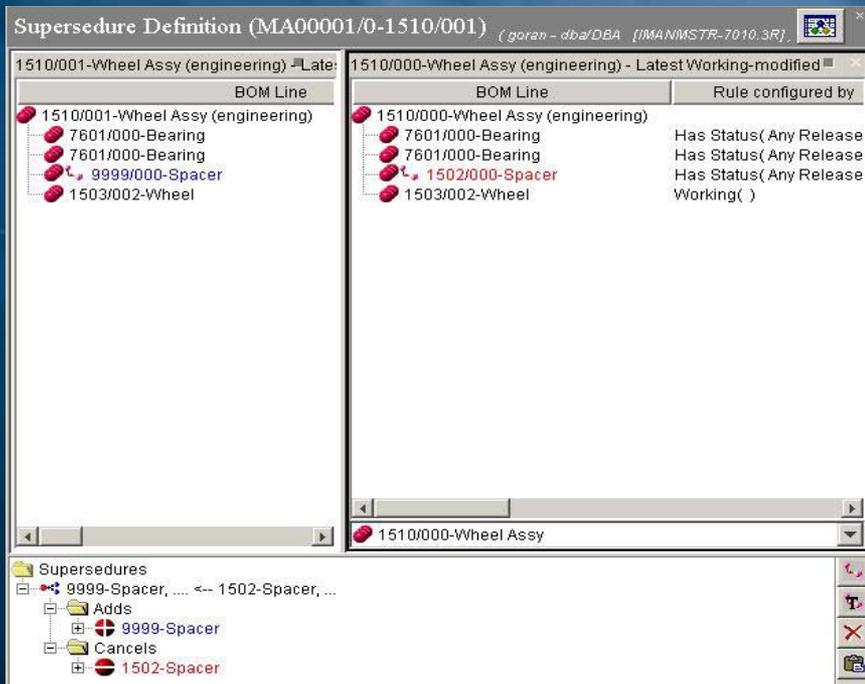
**Variant Rule:**  
Black with Thin Nib:  
Colour = **Blue**  
Nib = *Thin*

## Summary

Variant rules are used to configure specific models of a generic structure based on values of different options.

# Supersedure and Structure Compare

- ▶ Supersedure shows a structure BEFORE and AFTER a change is applied
- ▶ Supersedure highlights changes both graphically and in the structure tree view
- ▶ Structure Compare enables user to compare two revisions of a structure or two different structures.
- ▶ Compare Modes:
  - ▶ Single level
  - ▶ Lowest Level
  - ▶ All Levels



Insert PSE Compare Screen Clip





# Summary



**UGS**

*Transforming the  
process of innovation*

**Questions?**

**TEAMCENTER**