

# Increasing productivity with requirements reuse and variant management with **DOORS Next Generation**

*DRM 1946*

*Eran Gery – DE, Rational Systems Solutions*

*Daniel Moul – Sr. Product Manager*

*Brian Steele – RM Architect*

## Innovate2014

The IBM Technical Summit

June 1 – 5 | Orlando, Florida

**< Innovate @ SPEED >**

#ibminnovate



## Please note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Plans are based on best information available and may change in future.

# Outline

- Motivations & Definitions
- Patterns
  - Branch from closest product/component
  - Use common reference assets
  - Negative & positive variability
  - Functional and Temporal Variation
- Enabling capabilities
  - 1. Configuration management
  - Demo
  - 2. Global configurations and product definition
  - 3. Parameterized
  - 4. Integrating feature modeling



# Doing more with less in a customizing world

- Trend toward mass customization and shorter product lifecycles
- More embedded software; more complex connected products
- Need to adhere to safety standards, compliance and regulations



Source: [http://commons.wikimedia.org/wiki/File:ITPB\\_health\\_Club.jpg](http://commons.wikimedia.org/wiki/File:ITPB_health_Club.jpg)



# Some reuse scenarios...

- Managing requirements for a product family e.g.,
  - A vehicle platform
  - A set of insurance claim systems
- Handling supply chain
  - Multiple suppliers with varying components
- Shared requirements across different programs for different customers
- Parallel development of multi-year programs
- Handling requirements for a trade-study prototypes

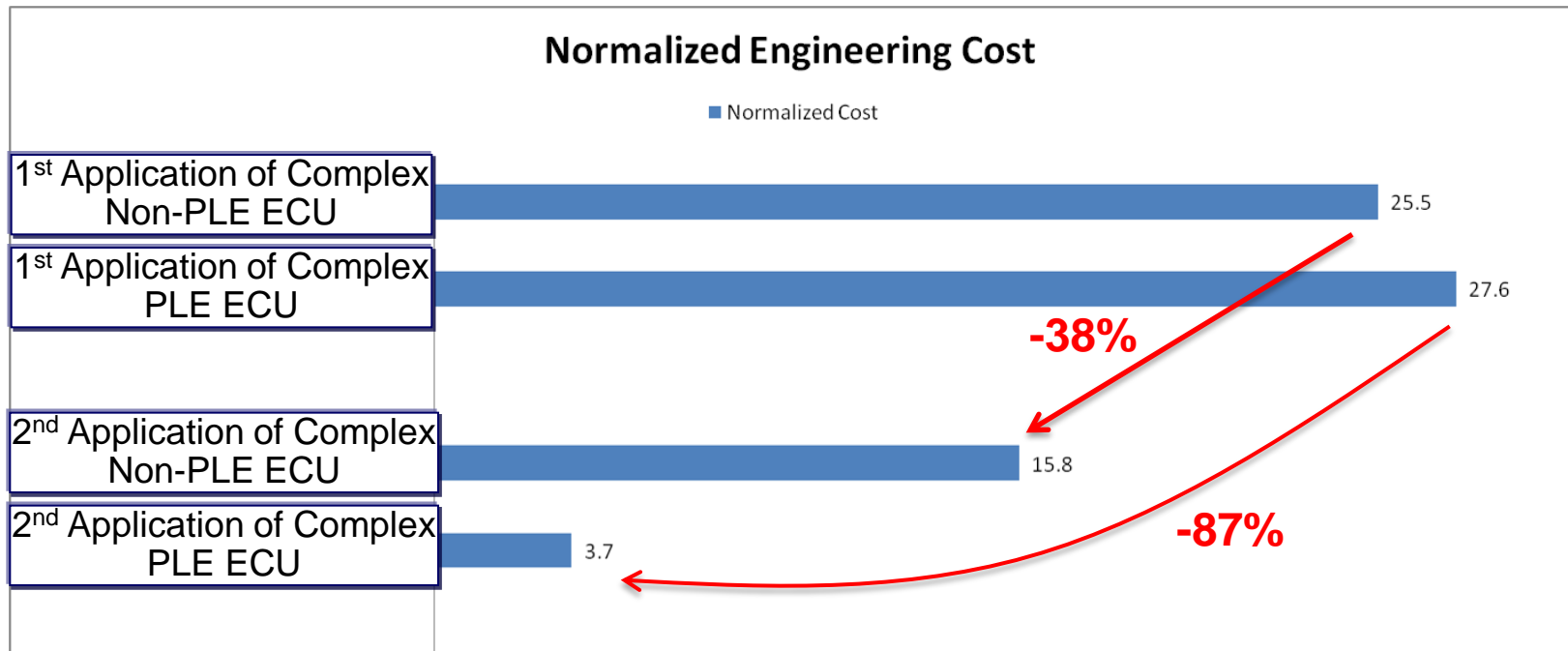


# Product Line Engineering

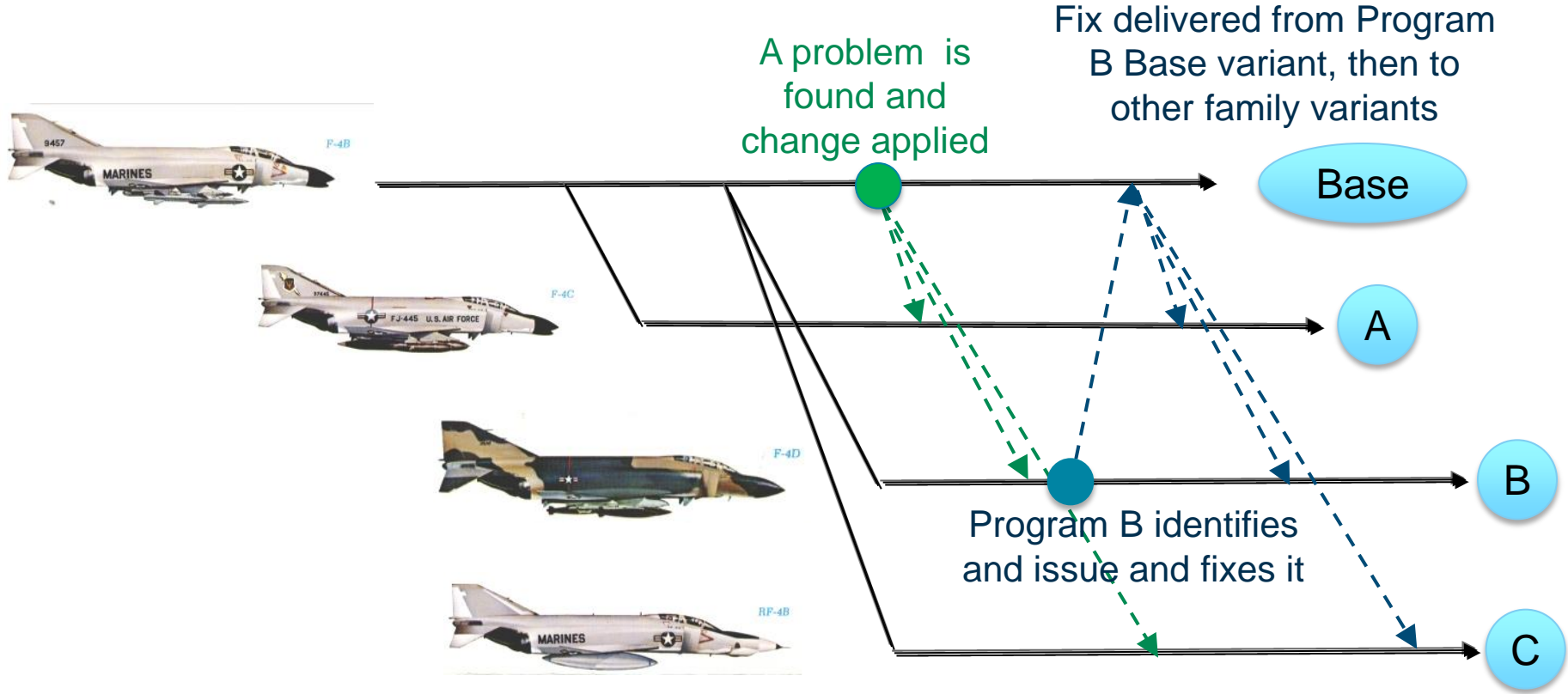
- Business & technical strategy
- Some automotive examples
  - Addressing different geographical markets (OEMs)
    - Safety regulations, Language, Driving side
  - Delivering parts to multiple OEMs (Suppliers)
  - 100s or 1000s of variants ... 1,000,000s of combinations



# GM started a reuse approach (PLE) in software engineering with impressive results:



# Strategic reuse: the conceptual scenario...



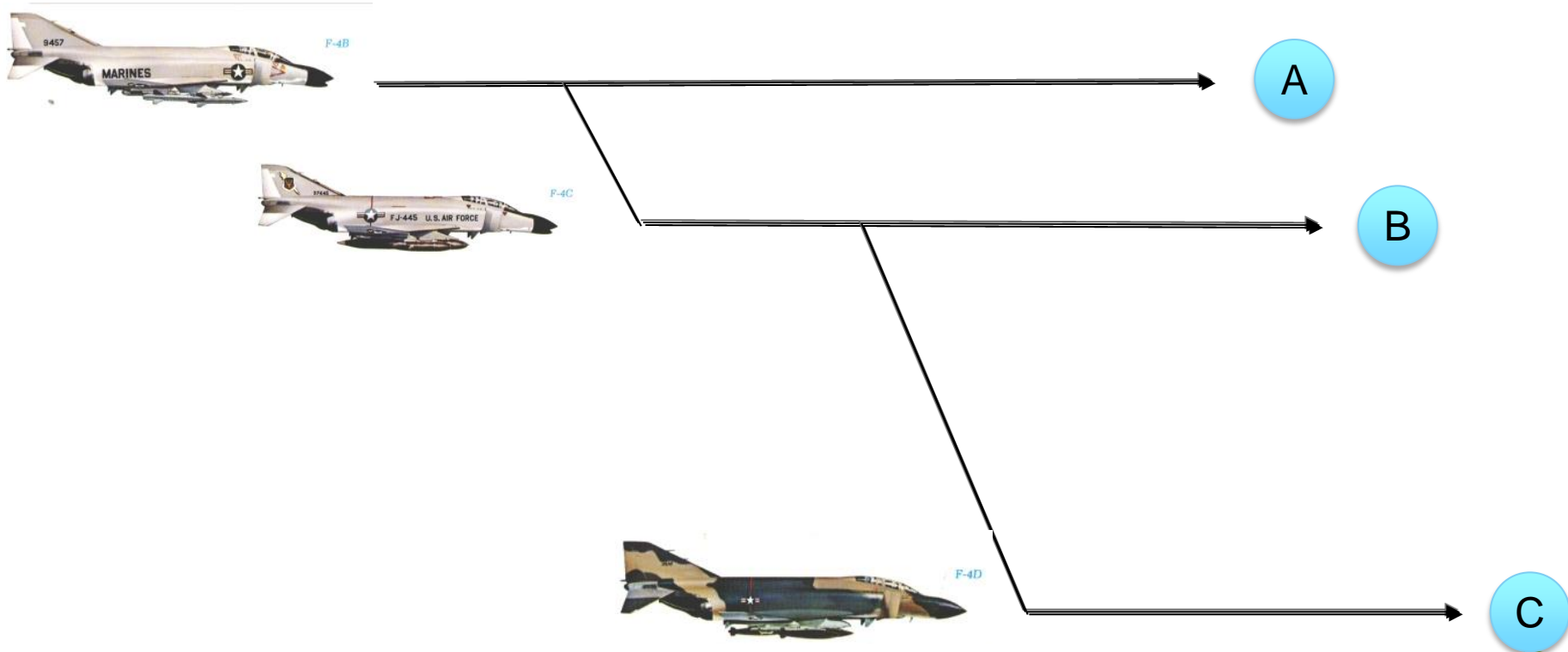
- Management of core platform engineering (“base”)
- Enable parallel engineering of platform variants
- Enable controlled reuse and change propagation downstream and upstream





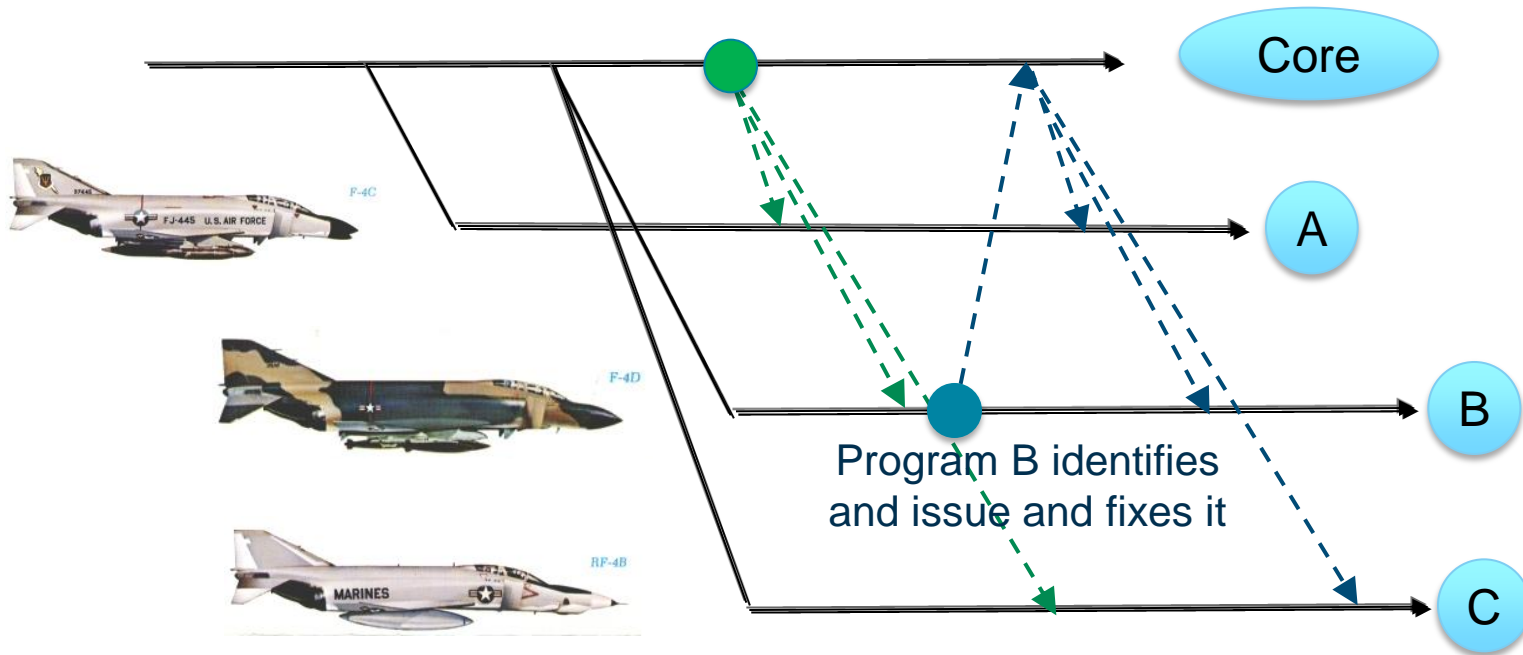
# Patterns of reuse

- “Branching” from closest product / component
- A more tactical reuse approach...



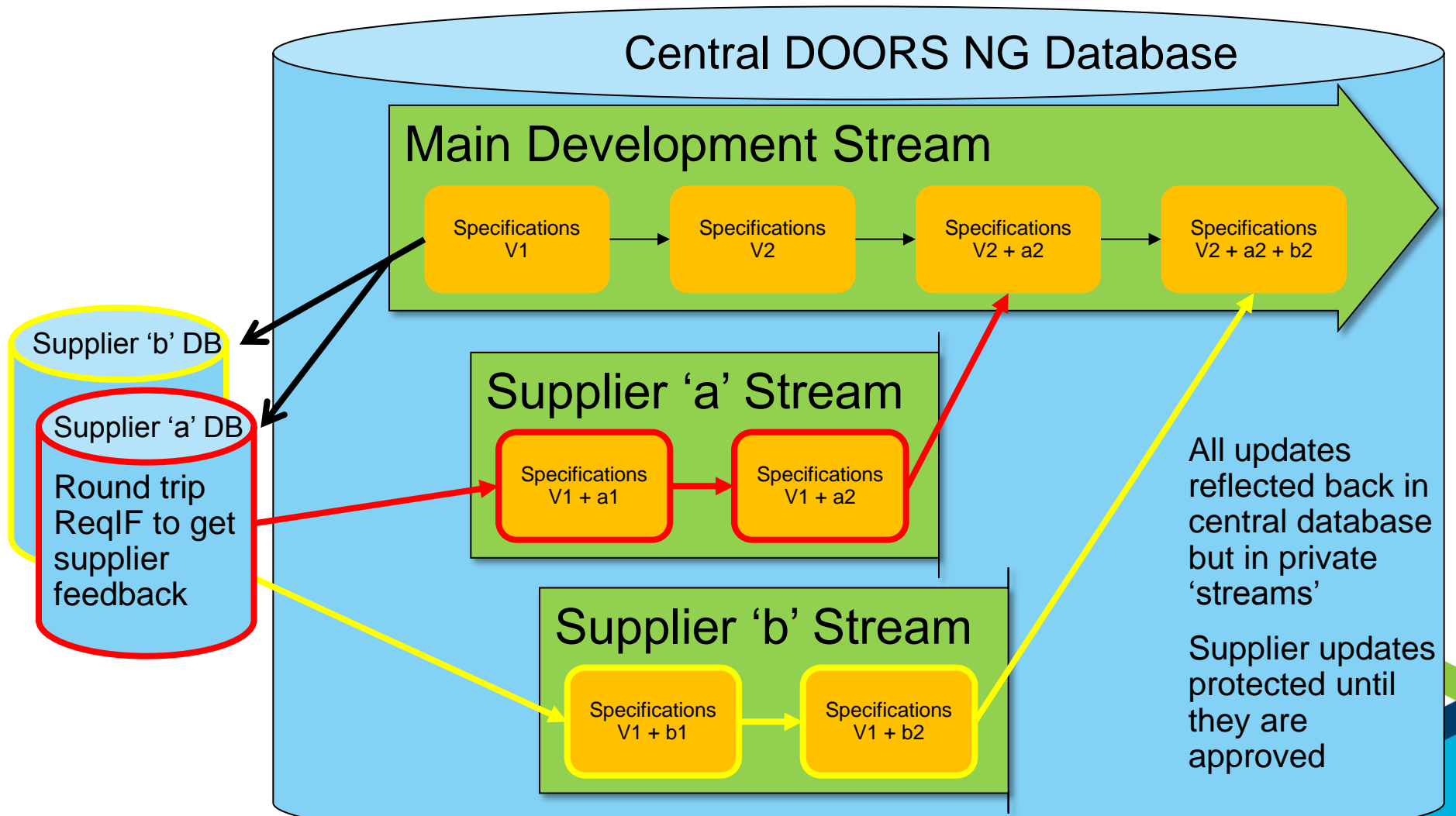
# Patterns of reuse

- Core assets pattern



# Example scenario: supplier communications

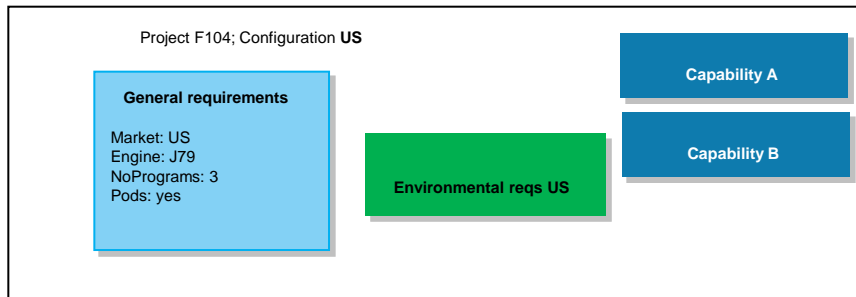
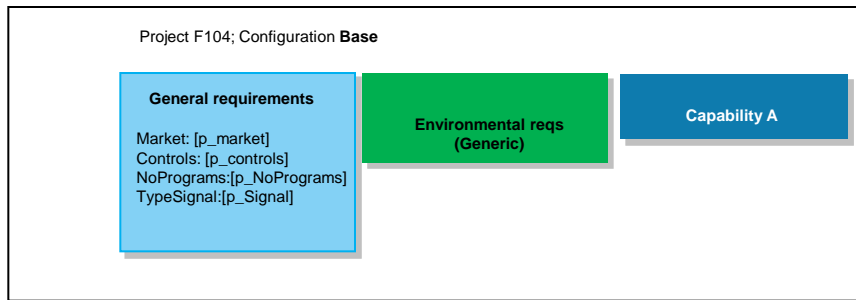
*ReqIF with requirements configuration management*



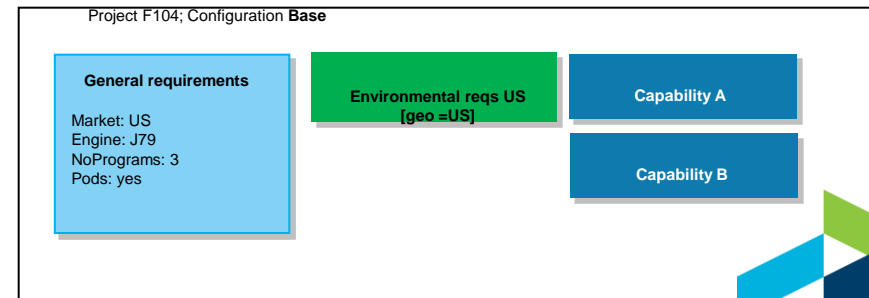
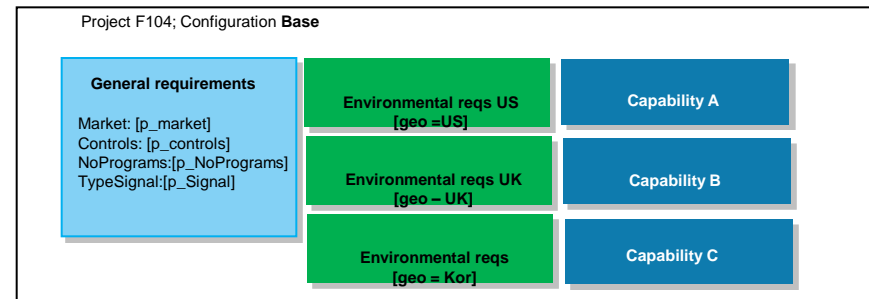
# Patterns of reuse (3)

- How are core assets reused: negative vs. positive variability

Positive: Branch, Add, Modify

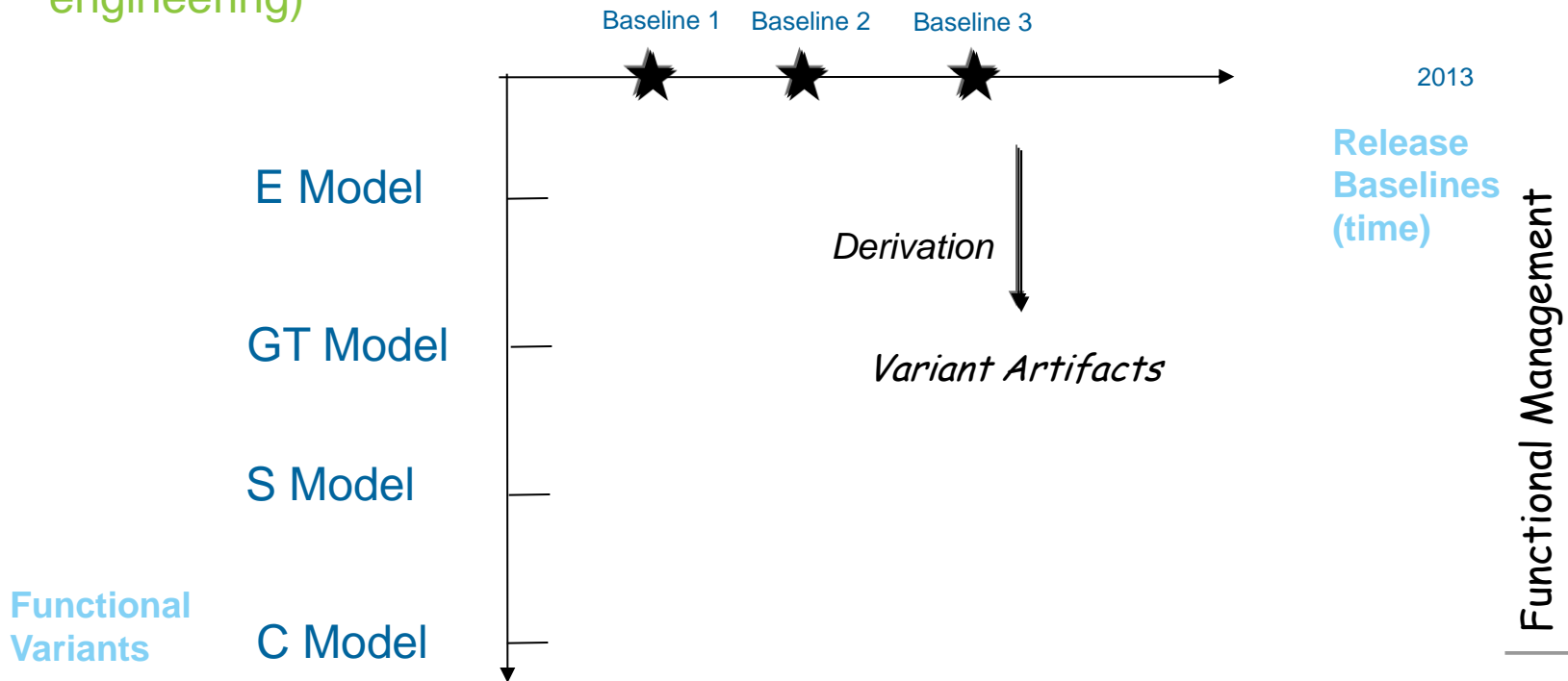


Negative: Filter, Branch + Derive



# Variability dimensions: functional variability

Product  
Development  
(AKA product  
engineering)

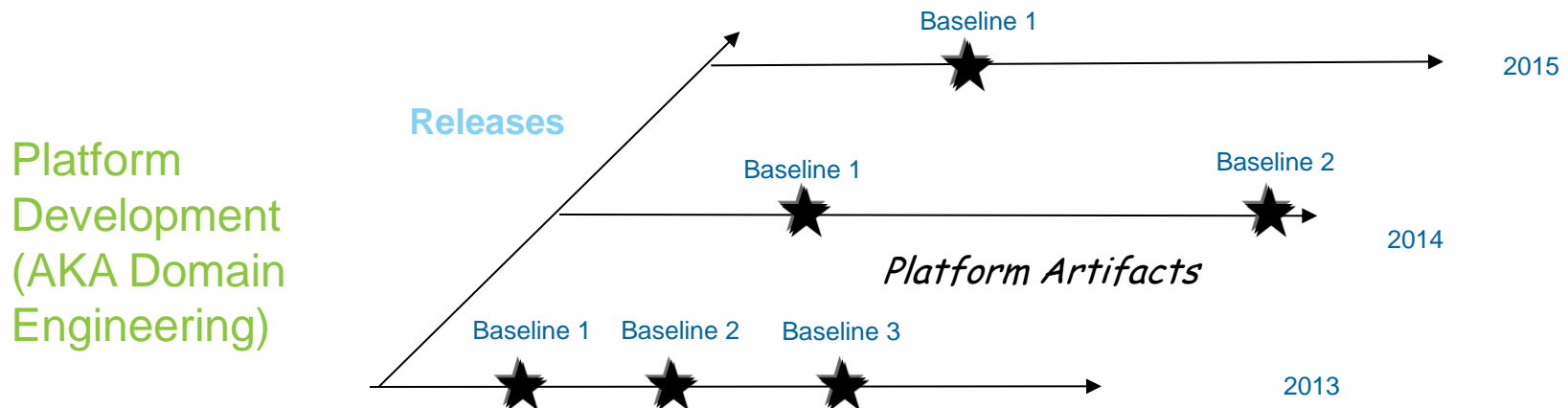


Note: Feature models & profiles evolve over time and are also temporally managed



# Variability dimensions: temporal variability

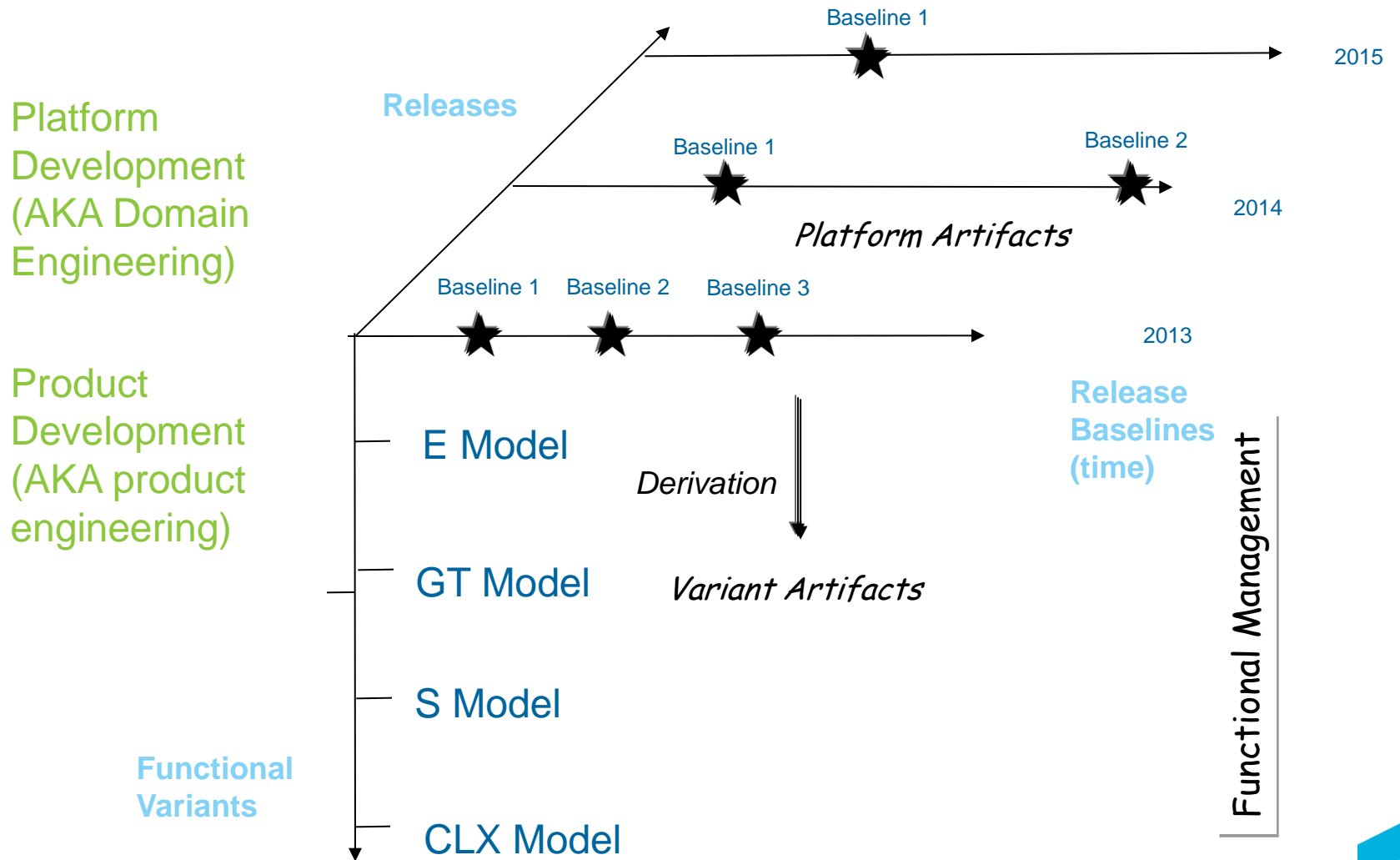
- Parallel configurations representing different temporal plans
- E.g – different annual plans, different iterations



The temporal dimension is needed if there is parallel engineering overlap between temporal targets. This is not always the case for requirements engineering. Sometimes it is needed for the V&V info.

Note: Feature models & profiles evolve over time and are also temporally managed

# Combining variants with parallel development...

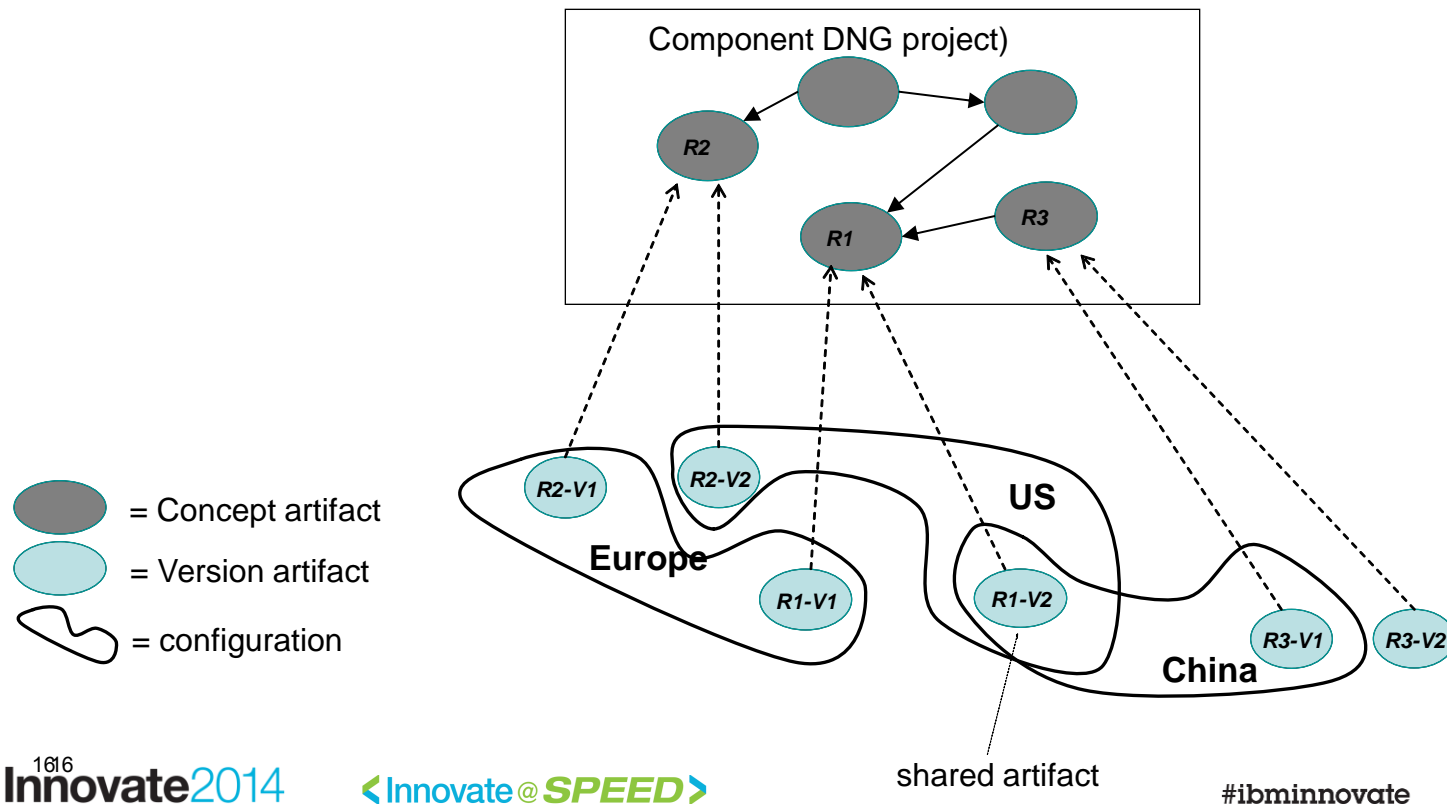


Note: Feature models & profiles evolve over time and are also temporally managed



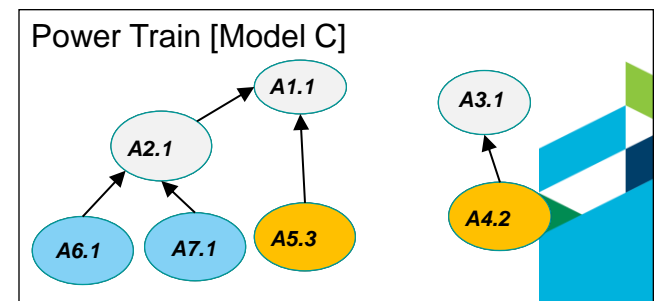
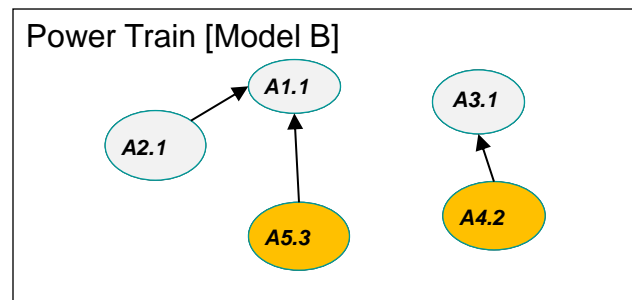
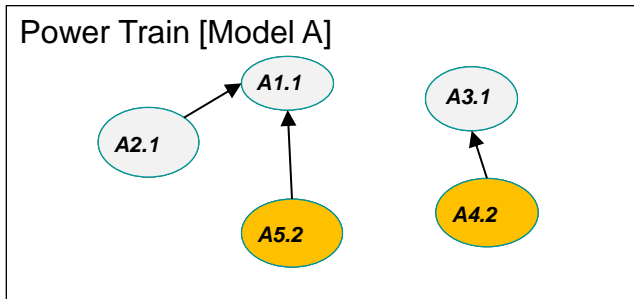
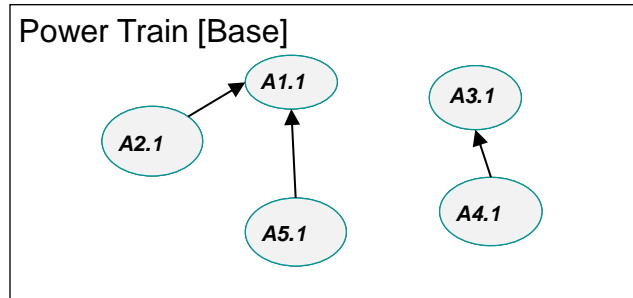
# Components and configurations

- Components are collection of logically related artifacts from a particular domain
- Artifacts have versions
- A (component) configuration specifies the included artifacts and their versions
- Configurations can share common artifacts and manage variability of other artifacts
- Configurations can be mutable (stream) or immutable (baseline)





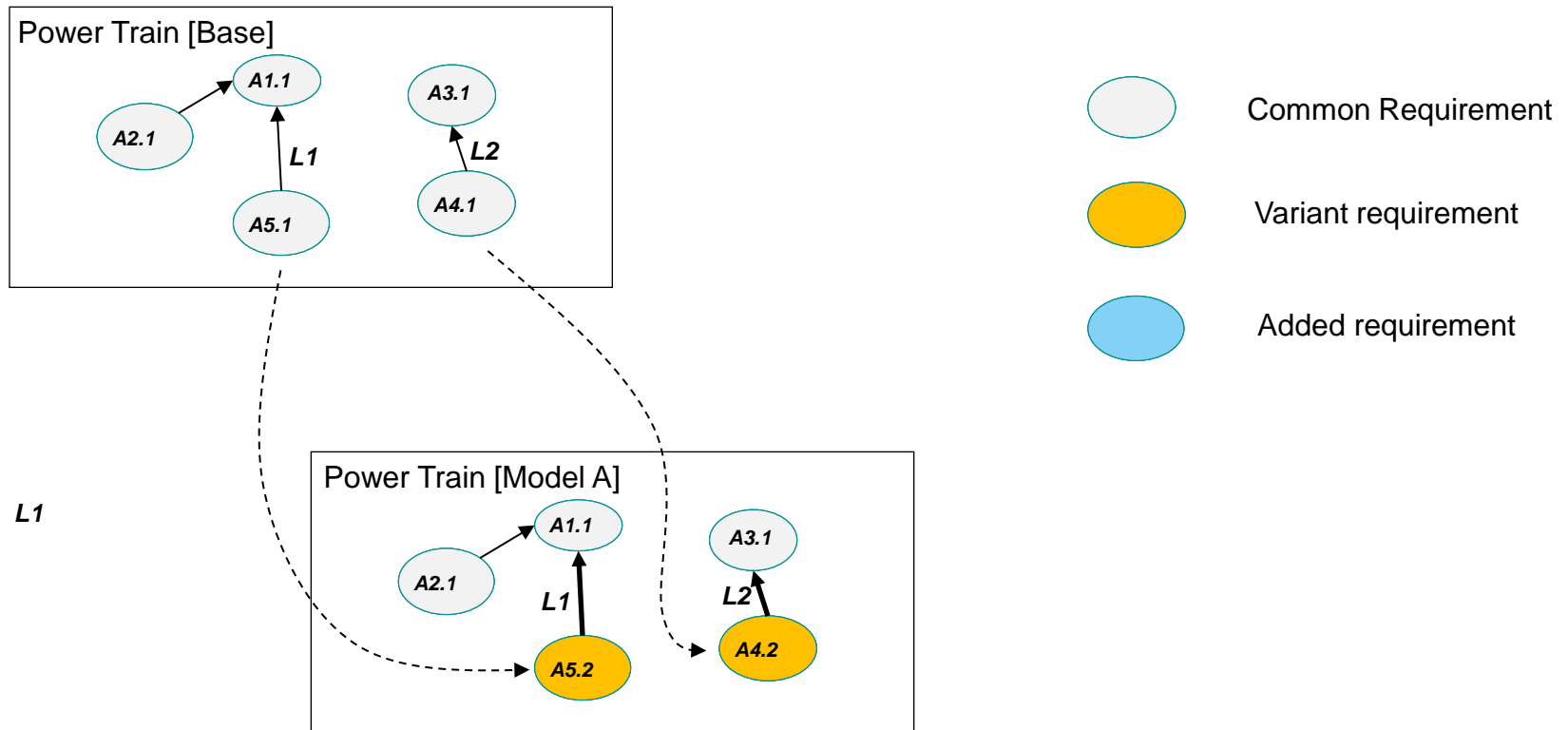
# Example: Components and configurations



#ibminnovate

# Conceptual Links

- What happens to links when we create new versions of requirements?
  - Conceptual links are defined relatively to conceptual requirements e.g. A1, A2, A3, A4, A5

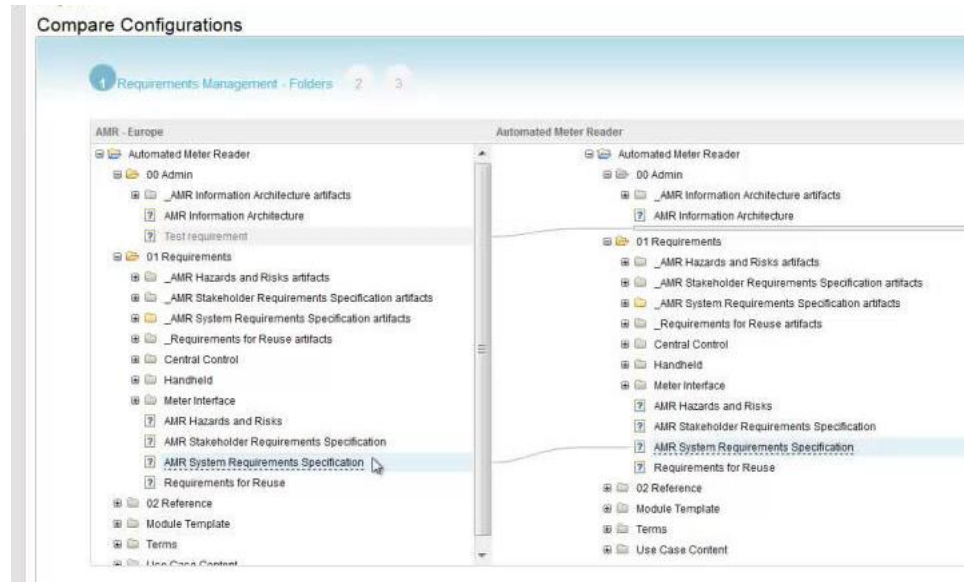


Conceptual Links persist when versions of objects are replaced in a configuration



# Example: Requirement configurations DOORS NG

- Diff contents of two configurations
- Diff contents of module in two configurations



## Compare Configurations

Changes from AMR - Europe to Automated Meter Reader

**615: AMR System Requirements Specification**

Content	Attributes
289	The handheld device shall provide a means to automatically (electronically) read the meter
522	Leak diagnostic data, when successfully uploaded to the handheld device, shall be immediately available for display on the handheld device.
319	<b>3.1.2 Meter Interface Unit</b>
248	The meter interface unit shall operate using walk-by, mobile (vehicle-based), and mesh network collection platforms
466	The meter interface unit shall sample water flow every 15 minutes in a 24 hour period to determine leakage.
444	The meter interface unit shall support all data collection functions (data reading, time-triggered operation, and management) of the AMR system
440	The meter interface unit shall employ two-way communications down to the endpoint making it possible for operators to push internal data requests, firmware updates, new capabilities and updated monitoring schedules via the network.
264	The meter interface unit shall be compatible with the existing meter models in use for the area covered by this project.
248	The meter interface unit shall be powered by a replaceable long lasting battery (lithium or other).
458	The meter interface unit shall capture usage data hourly and store this consumption data for up to 365 days. This hourly consumption data is considered usage profile data.

458 The meter interface unit shall capture usage data hourly and store this consumption data for up to 365 days. This hourly consumption data is considered usage profile data.

468 Meter usage data and leak diagnostic data shall be retrievable on demand from any meter interface via the network or a handheld.

430 When connected to a fixed network, the meter interface unit shall 'wake up' and communicate for 4 seconds every 30 minutes, synchronizing all clocks and configuration information. Between these transmissions, the unit remains in a low power state, conserving battery life.

367 The meter interface shall detect water leaks and record leak status with the account data

466 The meter interface unit shall sample water flow every 15 minutes in a 24 hour period to determine leakage.

221 Show changes

**3.1.3 Fixed Network Automated Meter Reading System**

255 The system shall have a permanently installed network to capture meter readings

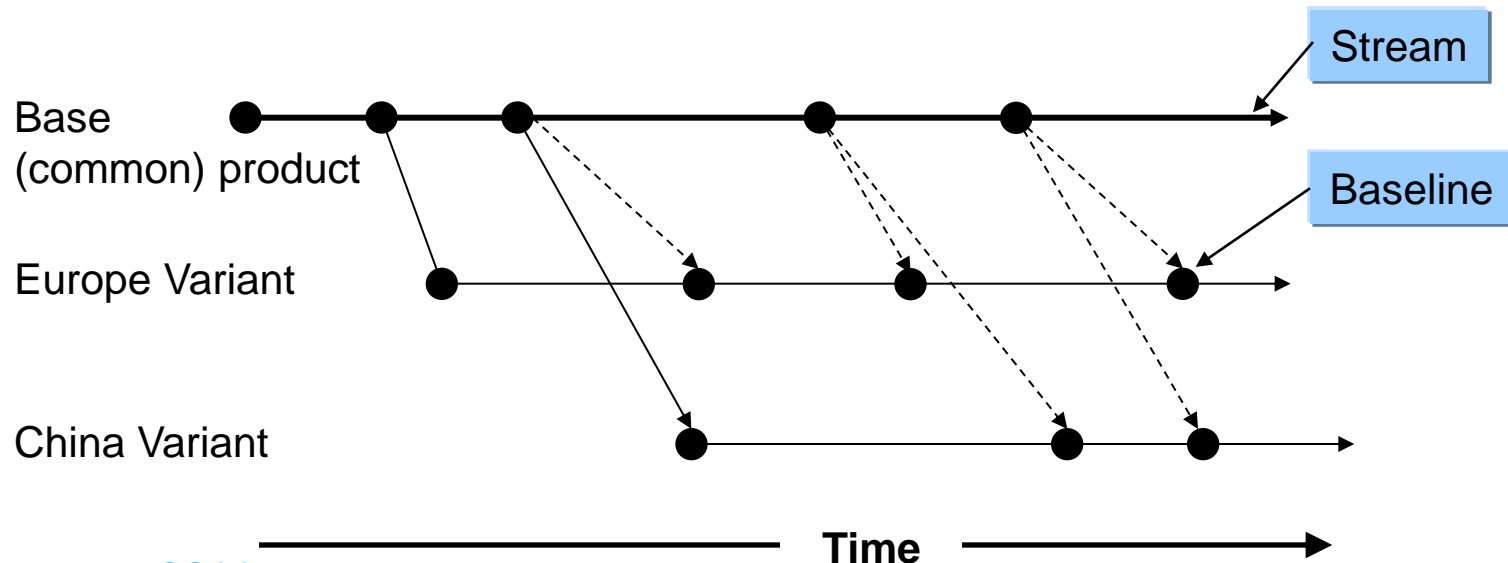
265 The system shall collect transmissions of meter readings from AMR capable meters and get the data to a central computer without a person in the field to collect it.

352 The systems shall forward a reading from a more remote area back to a main collector without actually storing it.

247 The system shall consist of a series of antennas, towers, collectors, repeaters, or other permanently installed infrastructure to collect transmissions of meter readings from AMR capable meters and get the data to a central computer without a person in the field to collect it.

# Realizing variant management with streams

- Each product variant is a branch of evolving artifacts
  - A stream or “workspace” – A mutable configuration
  - Streams are associated with baselines – Immutable configurations
- Common artifacts are shared across branches
- On one branch, evolution is a sequence of baselines
- New variants can be branched from existing variants
  - Can evolve in parallel
- Branches can update other branches using workspace delivery



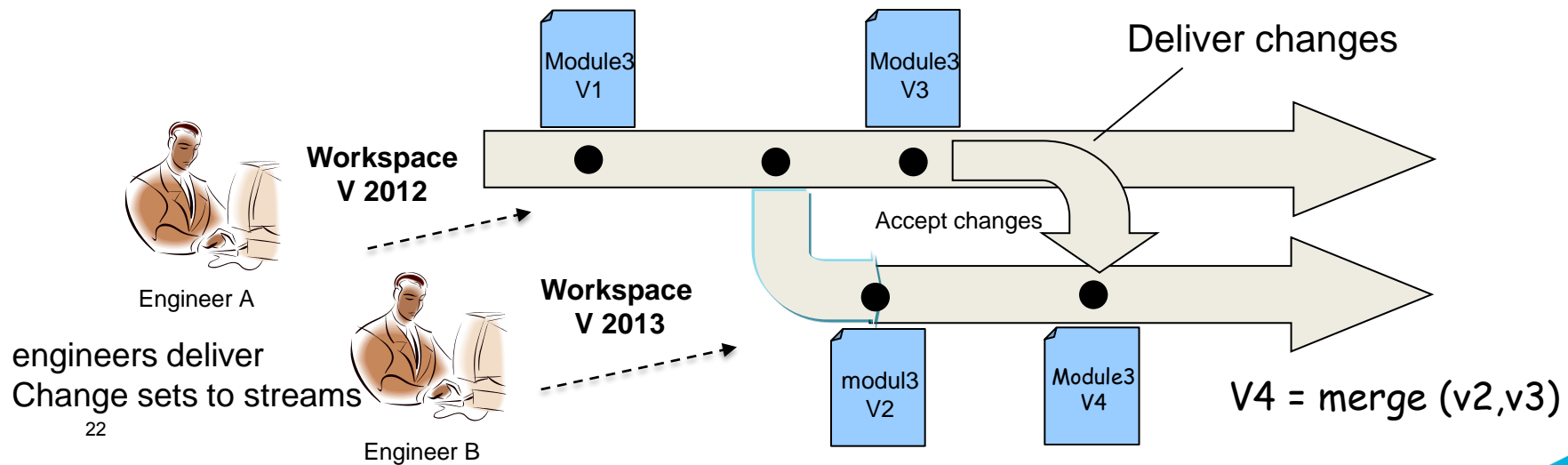
# Some essentials for reuse scenarios

- Changsets – a logical grouping of requirement changes that can be associated – e.g. with a change request
- Changeset Delivery, Delivery Targets
- Rebasing
- Key reuse patterns
  - Creating a new variant – create a child stream
  - Updating common requirements from the base stream to a variant
    - Rebasing
  - Updating the base with changes already in the variant – deliver changes
  - Looking at a difference between two variants – compare streams
  - Handling “conflicting” changes - merge



# Parallel development of components with streams

- Streams are mapped to **workspaces** in the various domain tools
- Changes to artifacts are shared into streams as **change-sets**
- Changes can be delivered across streams
- Deliveries may result in conflict detection that leads to a **merge**

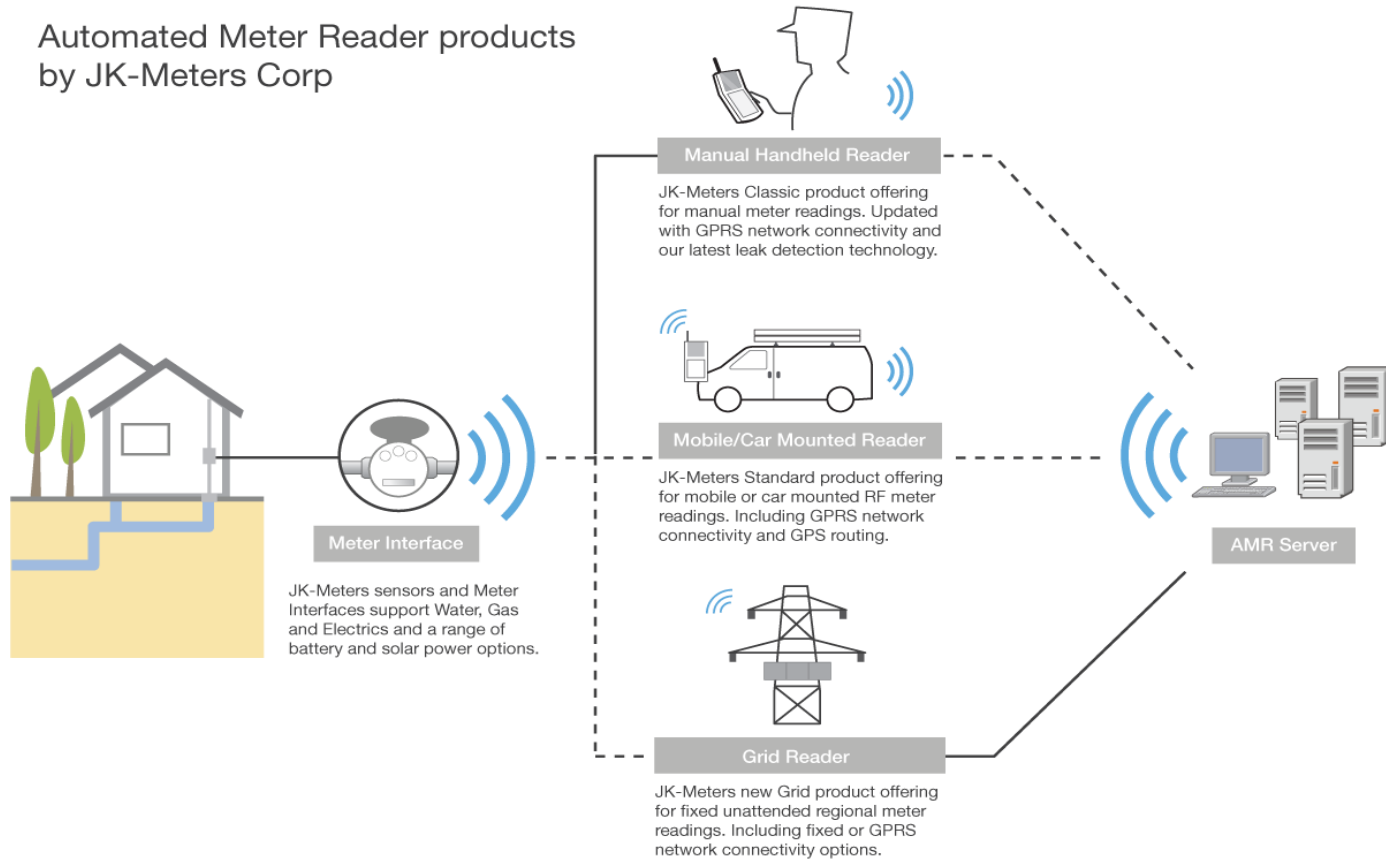


# DEMO



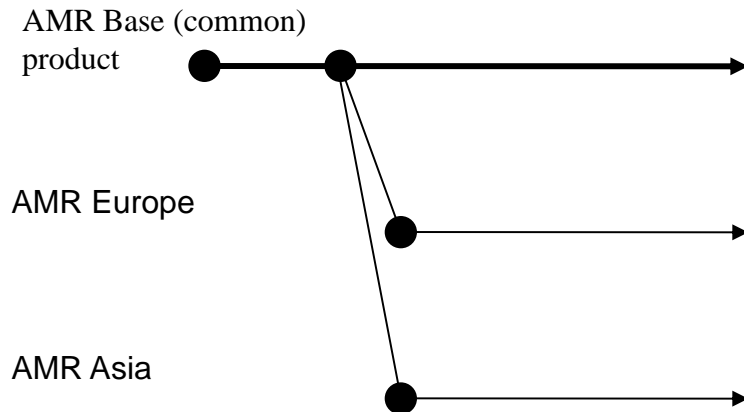
# Example: Automated Meter Reader Scenario

Automated Meter Reader products  
by JK-Meters Corp





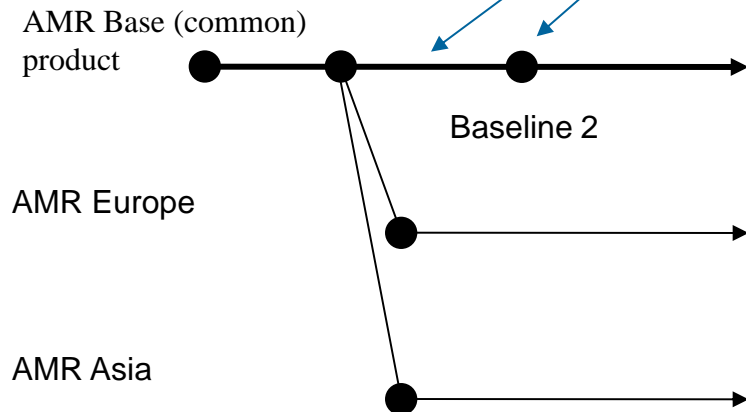
# Demonstration scenario - 1



- 1. Initial setup: AMR Base, AMR Europe, AMR Asia**
- Add new Requirements in the AMR Base Configuration. Create an explicit baseline.
  - Compare the new baseline with the old one
- In AMR Europe, rebase configuration on the new baseline from step 2
  - Show that changes are present in AMR Europe, not present in AMR Asia.
- Derive a new configuration - AMR South America using the baseline from step 2
  - Show that AMR Europe and AMR South America are identical
- In AMR Europe, make changes to a common requirement using a change set linked to a work item and deliver change to AMR Base.
- Baseline AMR Europe.
- Baseline AMR Base.
- Rebase AMR South America to new Global Baseline.

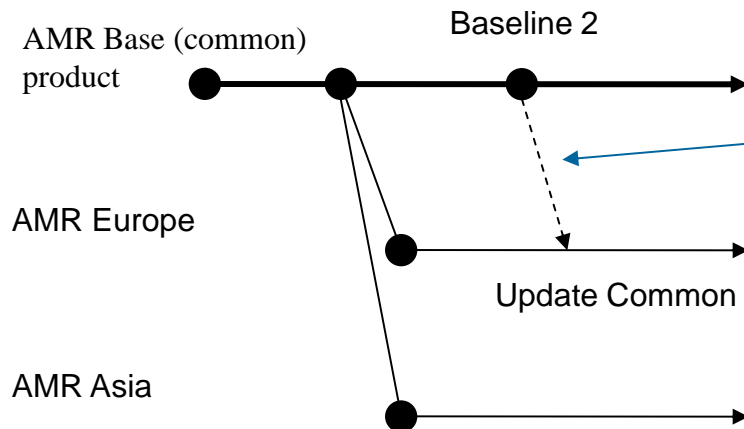


# Demonstration scenario - 2



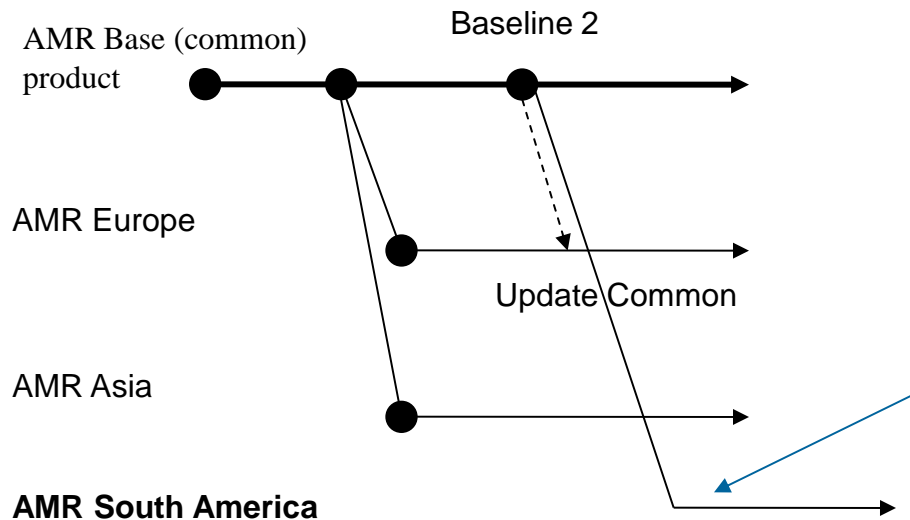
1. Initial setup: AMR Base, AMR Europe, AMR Asia
- 2. Add new Requirements in the AMR Base Configuration. Create an explicit baseline.**
  - **Compare the new baseline with the old one**
3. In AMR Europe, rebase configuration on the new baseline from step 2
  - Show that changes are present in AMR Europe, not present in AMR Asia.
4. Derive a new configuration - AMR South America using the baseline from step 2
  - Show that AMR Europe and AMR South America are identical
5. In AMR Europe, make changes to a common requirement using a change set linked to a work item and deliver change to AMR Base.
6. Baseline AMR Europe.
7. Baseline AMR Base.
8. Rebase AMR South America to new Global Baseline.

# Demonstration scenario - 3



1. Initial setup: AMR Base, AMR Europe, AMR Asia
2. Add new Requirements in the AMR Base Configuration. Create an explicit baseline.
  - Compare the new baseline with the old one
- 3. In AMR Europe, rebase configuration on the new baseline from step 2**
  - Show that changes are present in AMR Europe, not present in AMR Asia.
4. Derive a new configuration - AMR South America using the baseline from step 2
  - Show that AMR Europe and AMR South America are identical
5. In AMR Europe, make changes to a common requirement using a change set linked to a work item and deliver change to AMR Base.
6. Baseline AMR Europe.
7. Baseline AMR Base.
8. Rebase AMR South America to new Global Baseline.

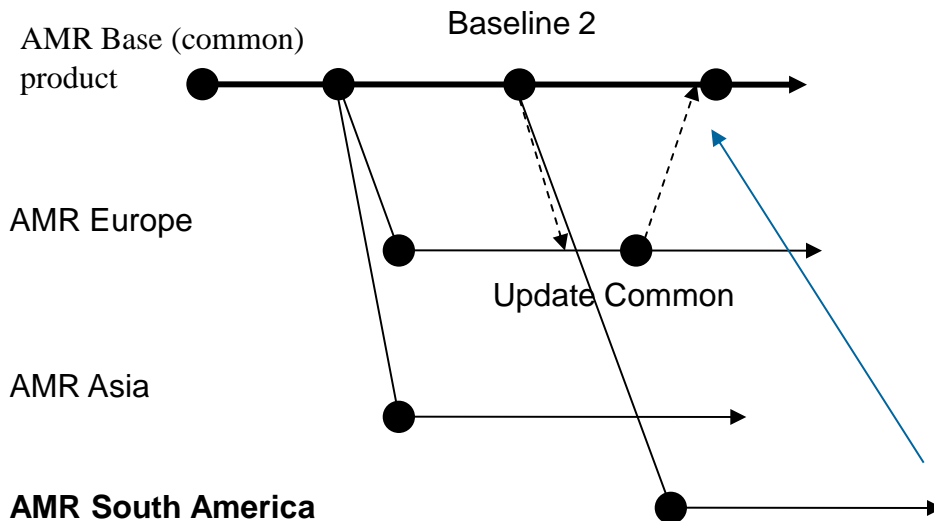
# Demonstration scenario - 4



1. Initial setup: AMR Base, AMR Europe, AMR Asia
2. Add new Requirements in the AMR Base Configuration. Create an explicit baseline.
  - Compare the new baseline with the old one
3. In AMR Europe, rebase configuration on the new baseline from step 2
  - Show that changes are present in AMR Europe, not present in AMR Asia.
4. **Derive a new configuration - AMR South America using the baseline from step 2**
  - Show that AMR Europe and AMR South America are identical
5. In AMR Europe, make changes to a common requirement using a change set linked to a work item and deliver change to AMR Base.
6. Baseline AMR Europe.
7. Baseline AMR Base.
8. Rebase AMR South America to new Global Baseline.



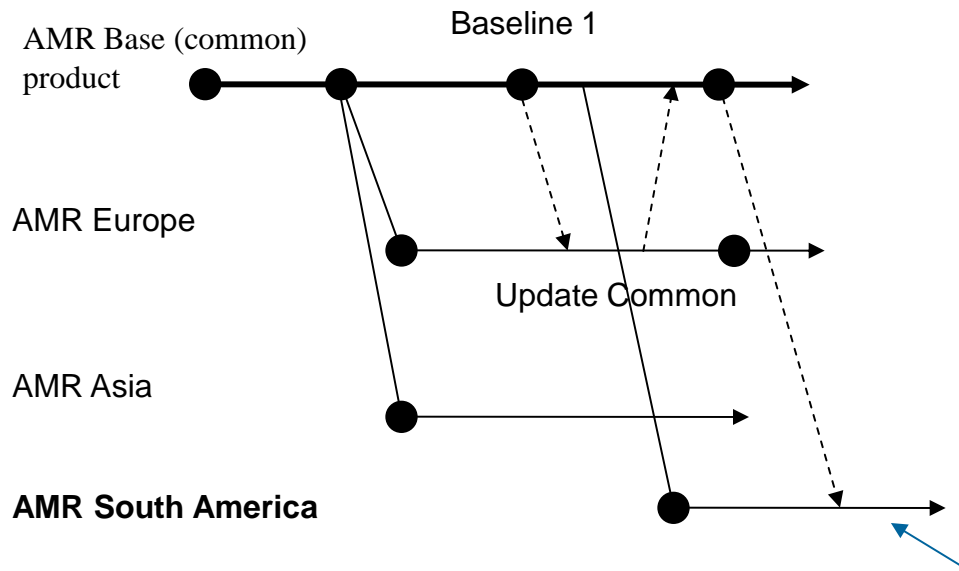
# Demonstration scenario - 6



1. Initial setup: AMR Base, AMR Europe, AMR Asia
2. Add new Requirements in the AMR Base Configuration. Create an explicit baseline.
  - Compare the new baseline with the old one
3. In AMR Europe, rebase configuration on the new baseline from step 2
  - Show that changes are present in AMR Europe, not present in AMR Asia.
4. Derive a new configuration - AMR South America using the baseline from step 2
  - Show that AMR Europe and AMR South America are identical
5. In AMR Europe, make changes to a common requirement using a change set linked to a work item and deliver change to AMR Base.
- 6. Baseline AMR Europe.**
7. Baseline AMR Base.
8. Rebase AMR South America to new Global Baseline.



# Demonstration scenario - 7

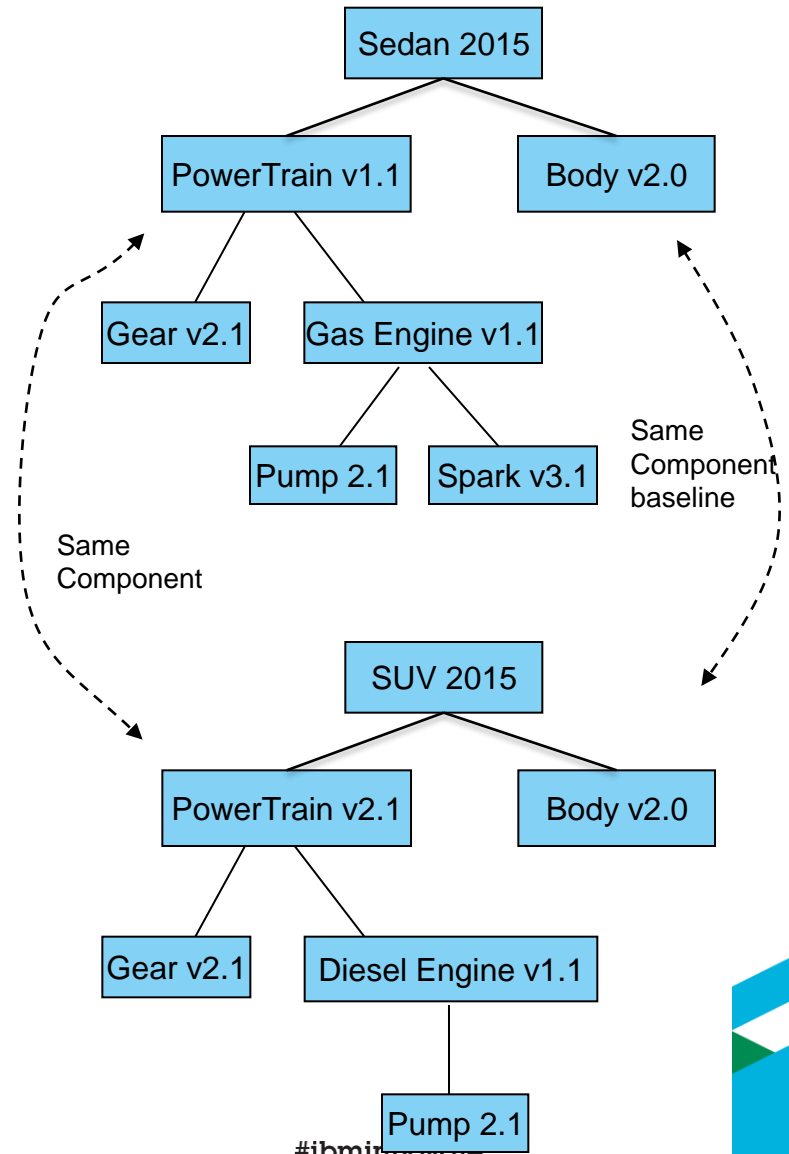


1. Initial setup: AMR Base, AMR Europe, AMR Asia
2. Add new Requirements in the AMR Base Configuration. Create an explicit baseline.
  - Compare the new baseline with the old one
3. In AMR Europe, rebase configuration on the new baseline from step 2
  - Show that changes are present in AMR Europe, not present in AMR Asia.
4. Derive a new configuration - AMR South America using the baseline from step 2
  - Show that AMR Europe and AMR South America are identical
5. In AMR Europe, make changes to a common requirement using a change set linked to a work item and deliver change to AMR Base.
6. Baseline AMR Europe.
- 7. Baseline AMR Base.**
- 8. Rebase AMR South America to new Global Baseline.**



# Handling multiple components - component dependencies

- To enable higher reuse, it is useful to organize requirements in multiple *components*
- Initially, DNG uses projects as component boundaries
  - To be refined in 2015
- Component configurations are linked using dependencies
  - Essentially imply a hierarchical structure
- A “component” can be part of multiple products
  - At same or different baseline

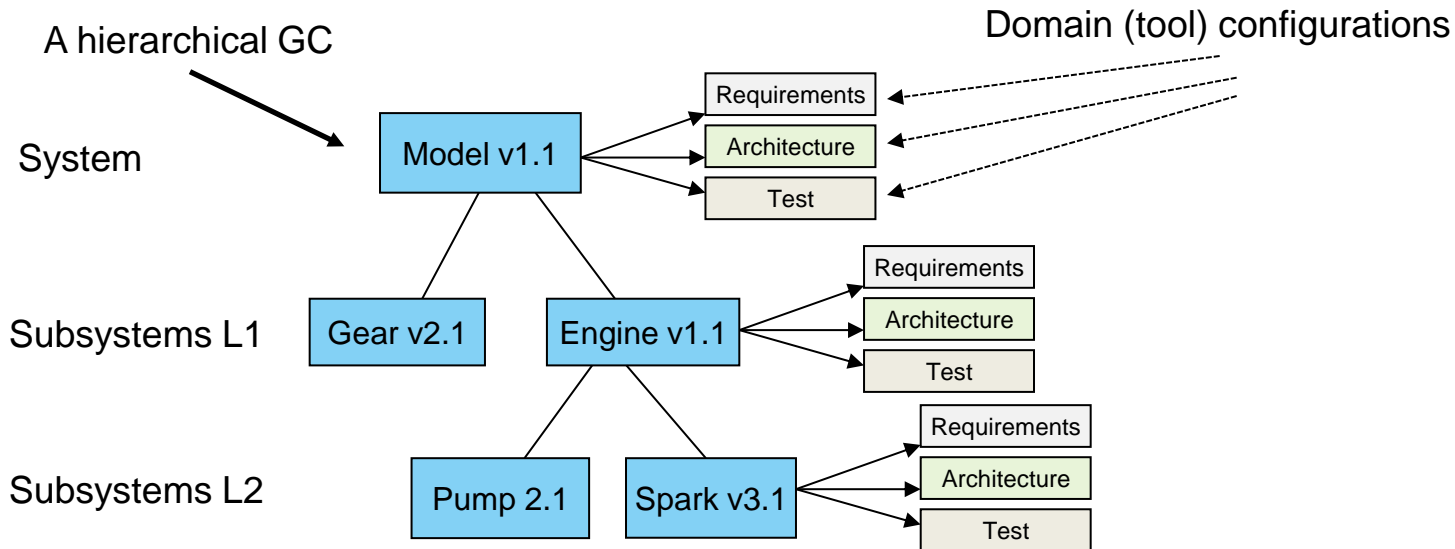




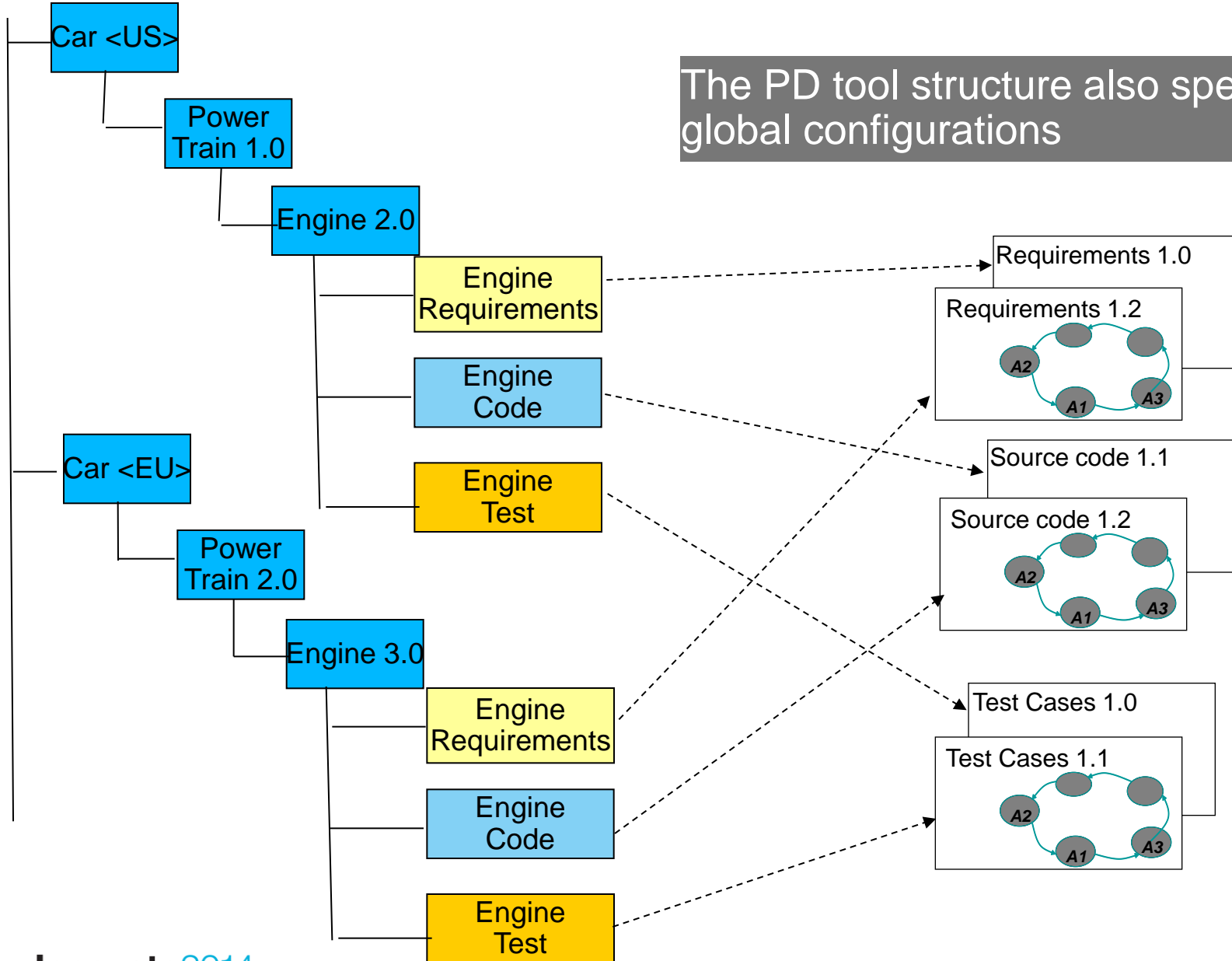
# The bigger picture – global configurations



- How do we configure *requirements* along with the respective *tests*, *architecture*, and *code*?
- Global configurations (GCs) create compositions of configurations into multi-domain composite configurations
- GCs are part of OSLC configuration management
- GCs can be hierarchical
- GCs can also be mutable (global streams) or immutable (global baselines)



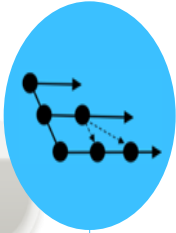
# Using the product definition tool to manage global configs



The PD tool structure also specifies global configurations



# Future outlook: more on reuse patterns



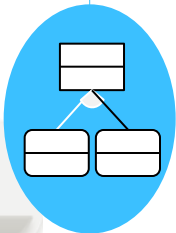
## Multi-stream

- Using requirements versions and component streams to realize variants
- What we've discussed so far...



## Parameterized

- Parameterizing components and artifacts for reuse



## Feature-driven

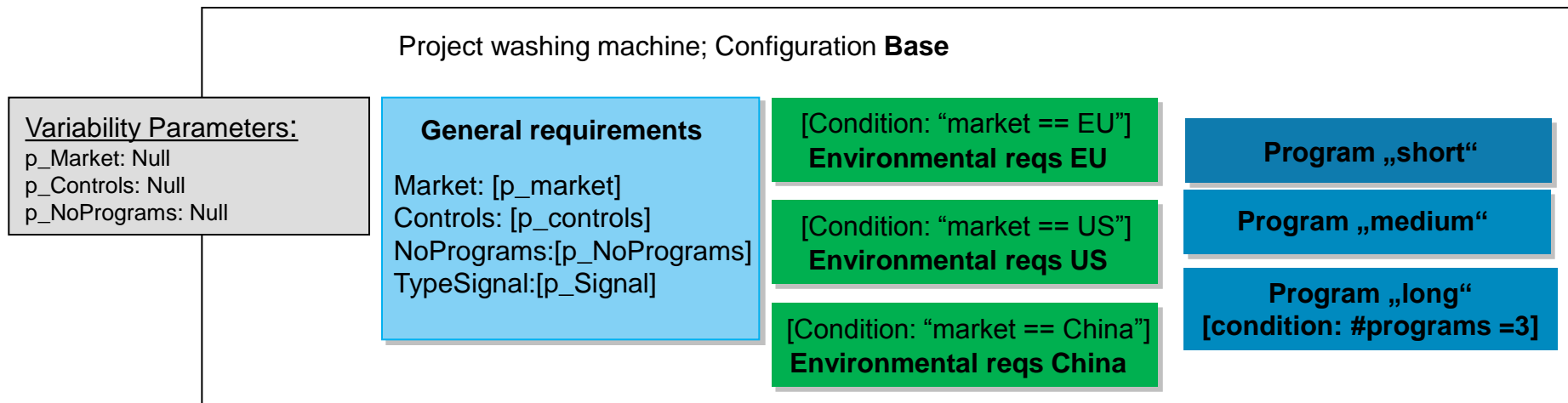
- Deriving requirements configurations by feature selection
- Integrate with feature modeling tools



# Parametric Variant Management

- Parametric components enable artifact reuse by parameterization
- Actual parameter settings derive concrete configurations of the parameterized artifacts
  - Conditional inclusion
  - Value substitution

Example: a parameterized component



# Example: Washing machine – derivation using parameters

## Project WashingMachine; Configuration **Europe**

### Variability Parameters:

p\_Market: Europe  
p\_Controls: Both  
p\_NoPrograms: 2

### General requirements

Market: Europe  
Controls: Both  
NoPrograms: 3

### Environmental reqs EU

### Program „Short“

### Program „Medium“

## Project WashingMachine; Configuration **US**

### Variability Parameters:

p\_Market: US  
p\_Controls: Sensor  
p\_NoPrograms: 3  
p\_Signal: Both  
p\_Color: Black

### General requirements

Market: US  
Controls: Sensor  
NoPrograms: 3

### Environmental reqs US

### Program „short“

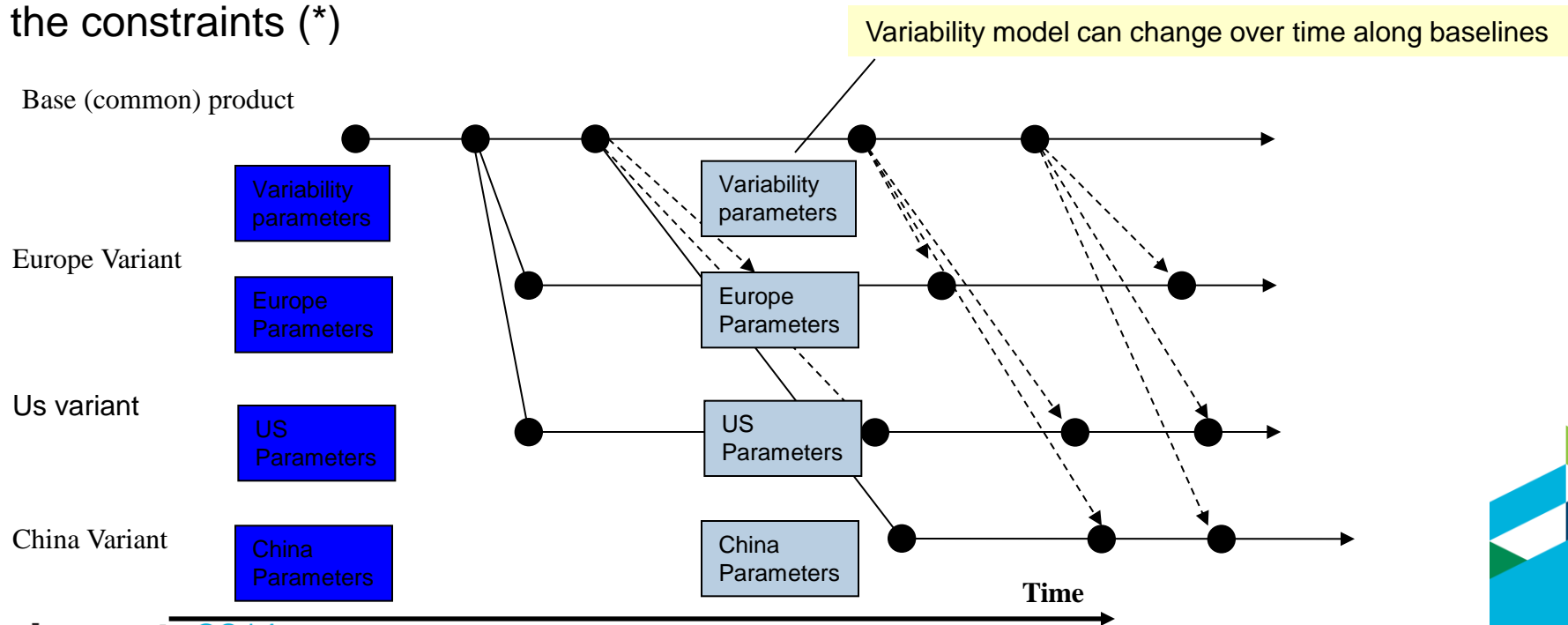
### Program „Medium“

### Program „Long“



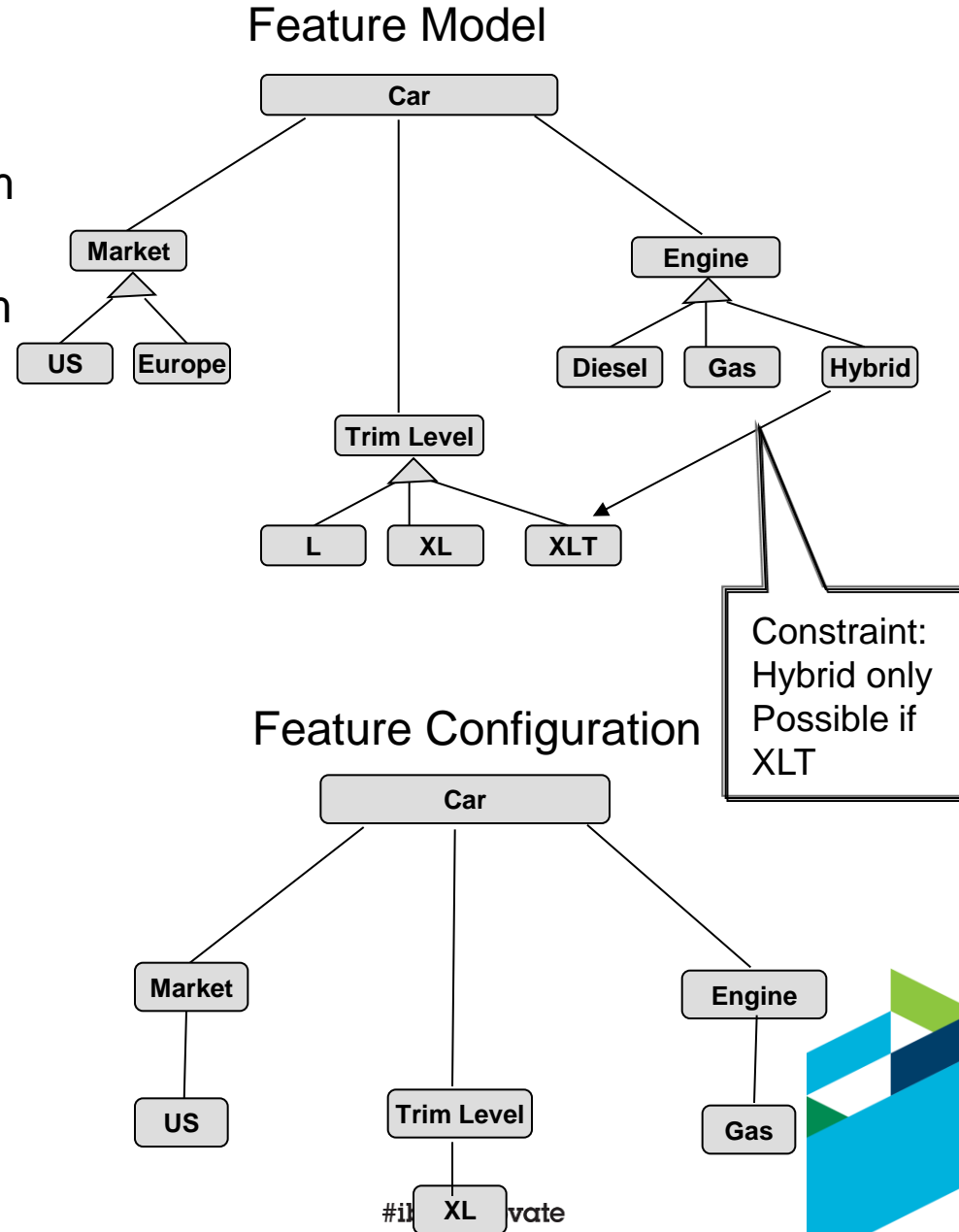
# Parametric variant management and product streams

- The base configuration introduce a document with a set of properties serve as the variability parameters (variability model)
- variant configuration assign different parameter values to those defined in the base
- Automation scripts modify the content of artifacts in variant configurations according to the assigned values (\*)
- In new platform baselines artifacts are pushed to variant configurations and variability update is calculated again
- Constraints can be checked using filters (query based) or scripts that check the constraints (\*)



# Feature Models

- Feature models represent the “problem domain” abstraction of the product line
- Capture a functional view of the system components and their variabilities from a product line management standpoint
- Feature models have configurations called *feature profiles*, which drive the variability parameters of the solution
  - In our case they can be mapped to dimension and dimension values
- We plan to enable 3<sup>rd</sup> party feature modeling tools to integrate with the platform PLE services
  - E.g. BigLever or PureVariants

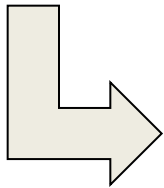
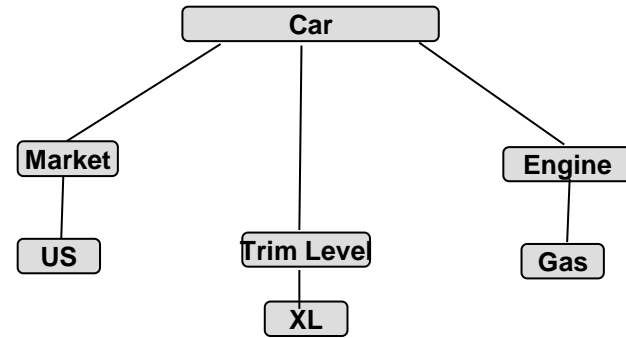


# Adding feature management to drive product configurations

## Feature selection

Variant/ Feature	Market	Trim	Engine	...	...
Variant 1	EU	L	Diesel		
Variant 2	US	XL	GAS		
Variant 3	US	XLT	GAS		

OR

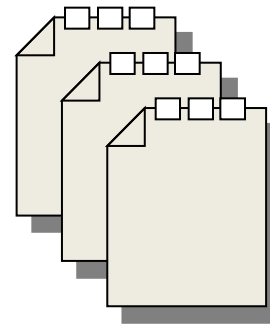


## Parameters configuration

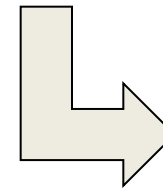
Engine = Gas  
Trim = XL  
Geo = US



## Parameterized Artifacts



## Product Artifacts





# Summary

- Configuration Management benefits
  - Isolated changes with controlled propagation
  - Reuse without copying
- Streams (workspaces) can express product versions and variants
- Propagate changes
  - In the common requirements by delivering them to the variant configurations
  - In a variant and deliver to the common stream
- Consider parametric and feature-driven approaches where you have a large variability space
- Start exploring now
  - Download the DOORS NG with CM open beta from [jazz.net](http://jazz.net)



#ibminnovate

# Thank You

**Innovate**2014

<Innovate@**SPEED**>



# Acknowledgements and Disclaimers

**Availability.** References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

© **Copyright IBM Corporation 2014. All rights reserved.**

- **U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.**

IBM, the IBM logo, ibm.com, are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)

Other company, product, or service names may be trademarks or service marks of others.

