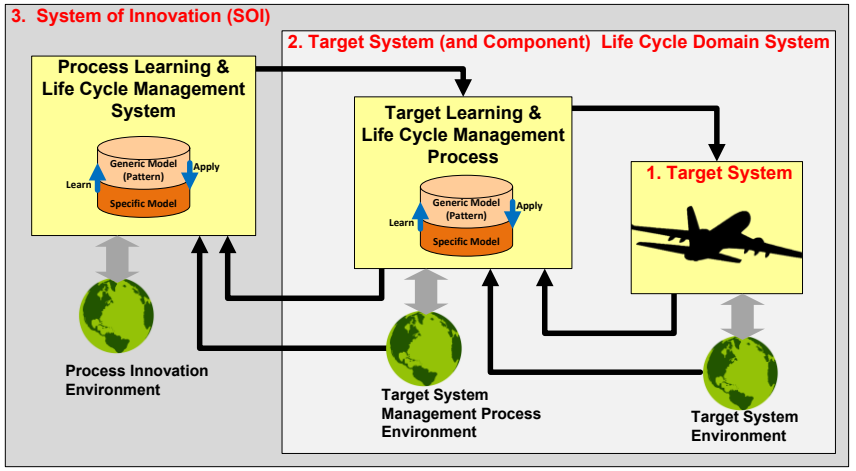
ICTT System Sciences®
Understand your systems.

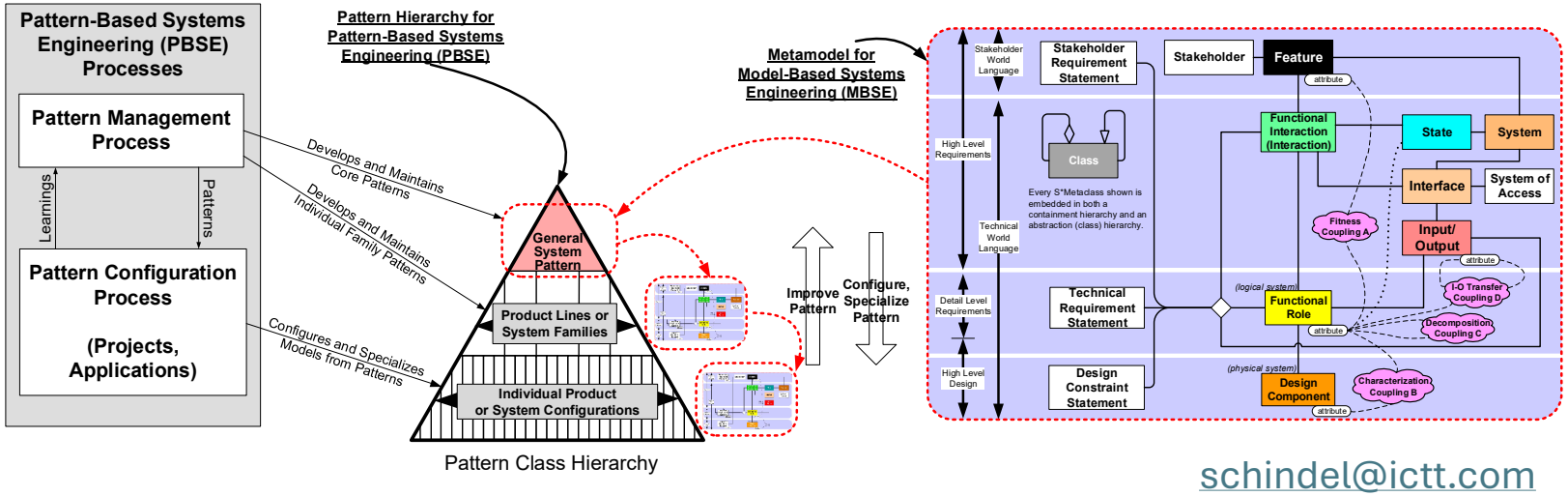
Phase I Session 6

<<Company/Org. Logo>>

Introduction to S*Models, S*Metamodel, S*Patterns, Mappings to Languages & Tools, S*Configuration Wizard, ASELCM S*Pattern



Level 0 View, INCOSE ASELCM Pattern



Pattern Class Hierarchy

schindel@ictt.com
lewis@ictt.com

V1.2.19

Copyright 2025, System Sciences, LLC

Contents

- Recap of Session 5:
  - Session 5 Take-aways, discussion, questions
  - Session 5 Homework
- Phase I Session 6 Class Content
- References

Recap of Session 5:
Take-aways impacting actual practices

What consistent practices seem possible *across this team*, *across different systems*, *customers*, and *projects*?

1. Describing variability of configurable systems, no matter when in life cycle they are configured, **using selectable Stakeholder Features and their attributes**?
2. **Driving configuration of objective Functional Interactions, Functional Roles, Requirements Statements, Design Components, and their Attributes**, etc., from Stakeholder Feature configuration, pattern configuration rules, and their consequences?
3. Curating updates to configurable S*Patterns as future project configurable starting point **repository of learnings about systems**—whether they are engineered systems or systems of engineering and life cycle management?

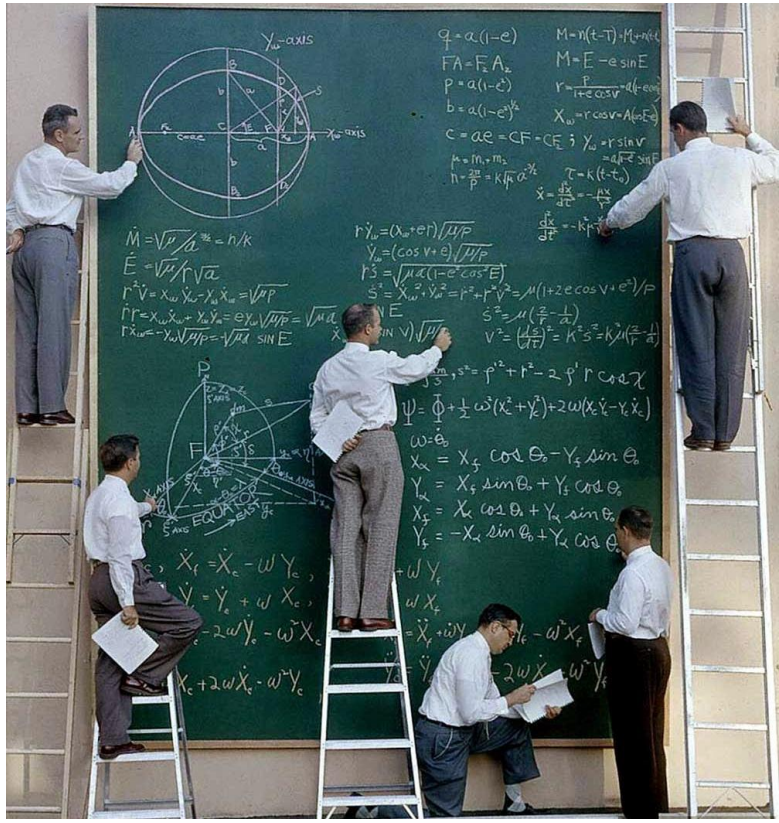


- Can a creative, talented team *align* its work?
- What should be *standard* about this, in your customer work?
- What should be *configurable* about this, in your customer work?

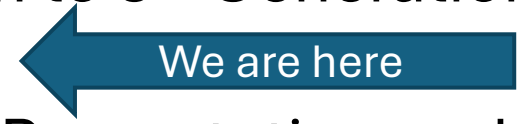


Session 5 Homework Discussion

- Use the example teaching Pattern (International Power Converter) or other pattern from prior homework assignments.
- Edit the Configuration Rules in the Pattern Functional Role Allocation and Pattern Requirements tables.



Phase I Class Sessions: Microstructure of Models and Patterns (6 Sessions, 3 hours each)

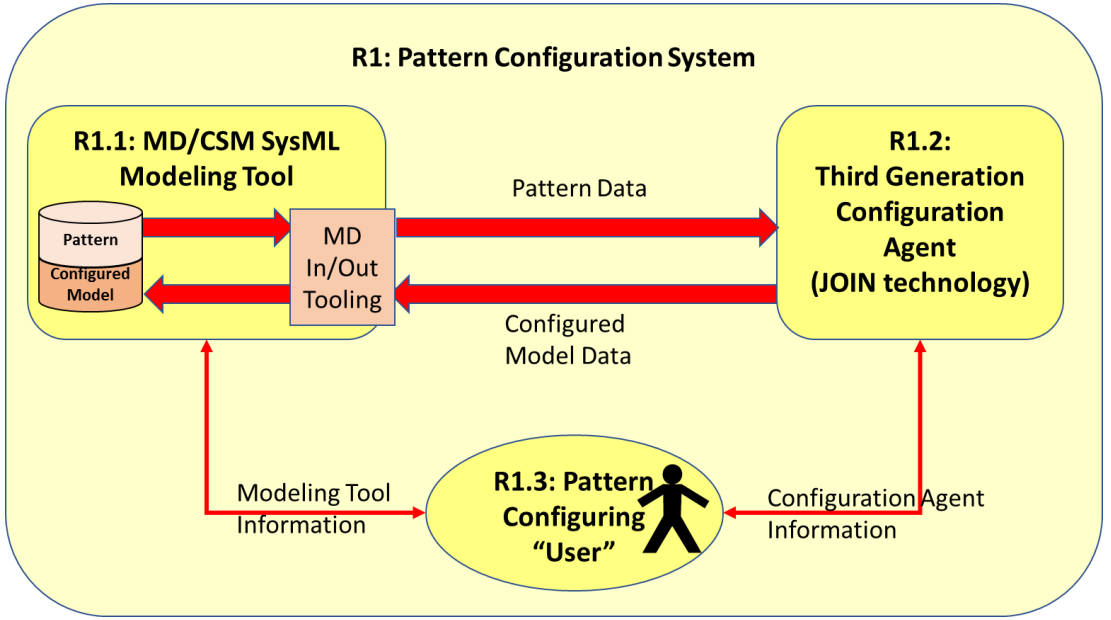
- S*Models, S*Metamodel:
  - Sessions 1-3
  - In Example Model Context
- Formal mapping of S*Metamodel into Tooling
  - Session 4
- S*Patterns, Configuration: In Example Model Context
  - Session 5
- Introduction to 3rd Generation Pattern Configurator Technology
  - Session 6 

- Homework Presentation and Feedback
  - Distributed over above sessions

Introduction to 3rd Generation Pattern Configurator Technology

Phase I Session 6 Class Content

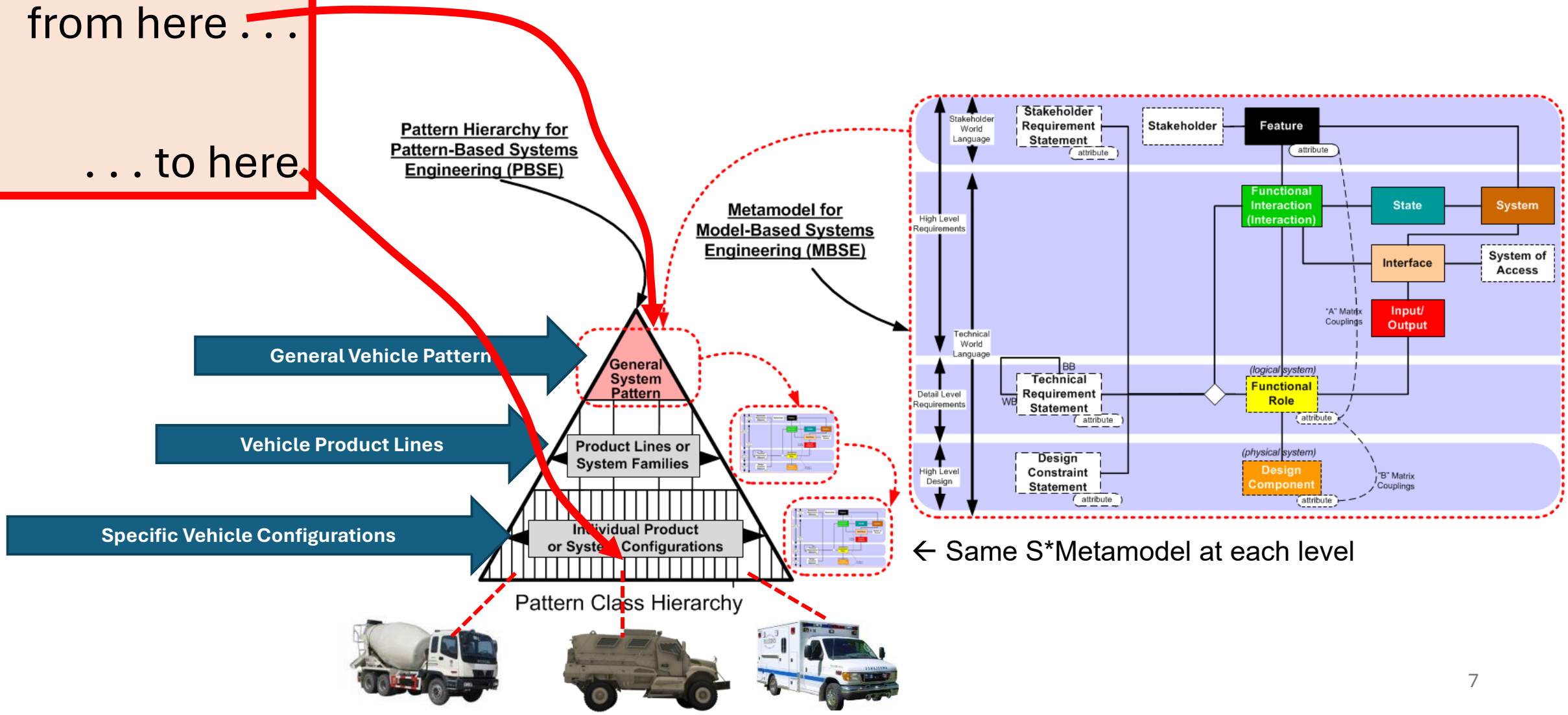
- Transforming S*Patterns to configured S*Models
- S*Pattern Configuration Algorithm
- Automating the Algorithm: Configuration Agents
- Three generations of S*Configuration Agents
- SysML and Cameo Modeler integration and interface specification
- Configure an S*Pattern Using the Configuration Wizard
- Configuration Agent Queries Overview
- Configuration Agent Controls Overview
- Session 6 Homework Assignment



Auto-transforming S*Patterns to configured S*Models (after an S*Pattern exists)

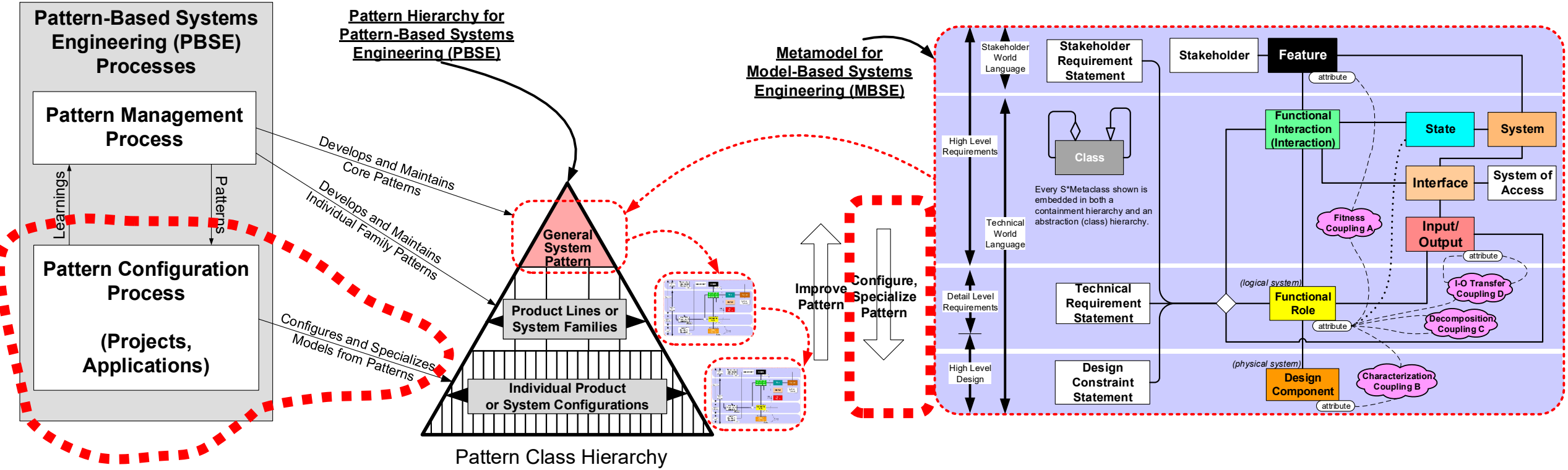
How to get from here . . .

. . . to here



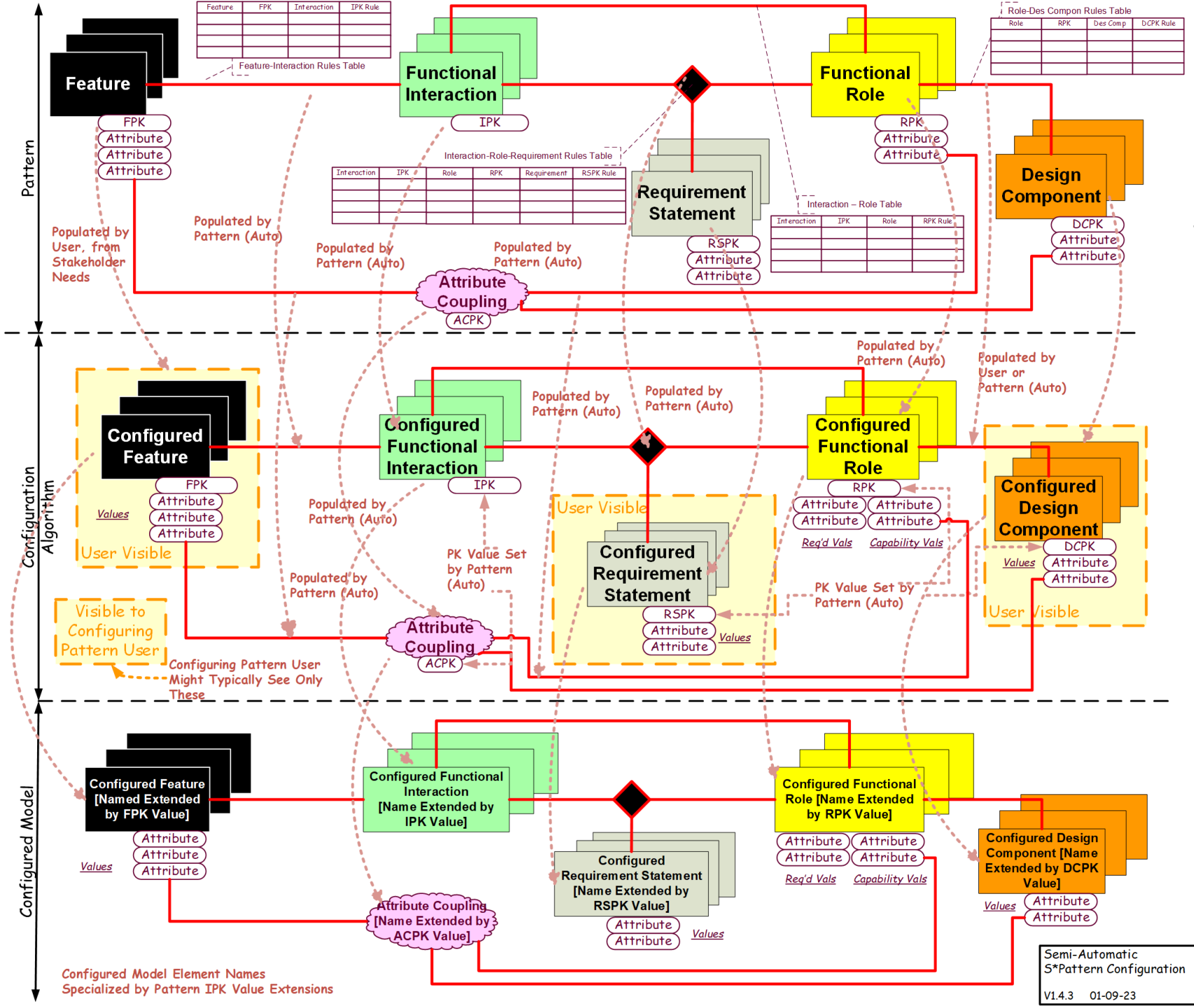
← Same S*Metamodel at each level

Transforming S*Patterns to configured S*Models



- **The Pattern Management Process:** Performed "across" projects, as a shared enterprise asset management process, capturing learning for effective future use. (Part of next (Phase II) ASELCM Ecosystem classes.)

- **The Pattern Configuration Process:** Performed once per "system project", in early stage, as starting point that incorporates what has been learned in the past into the start of a new project. Could cover much of the



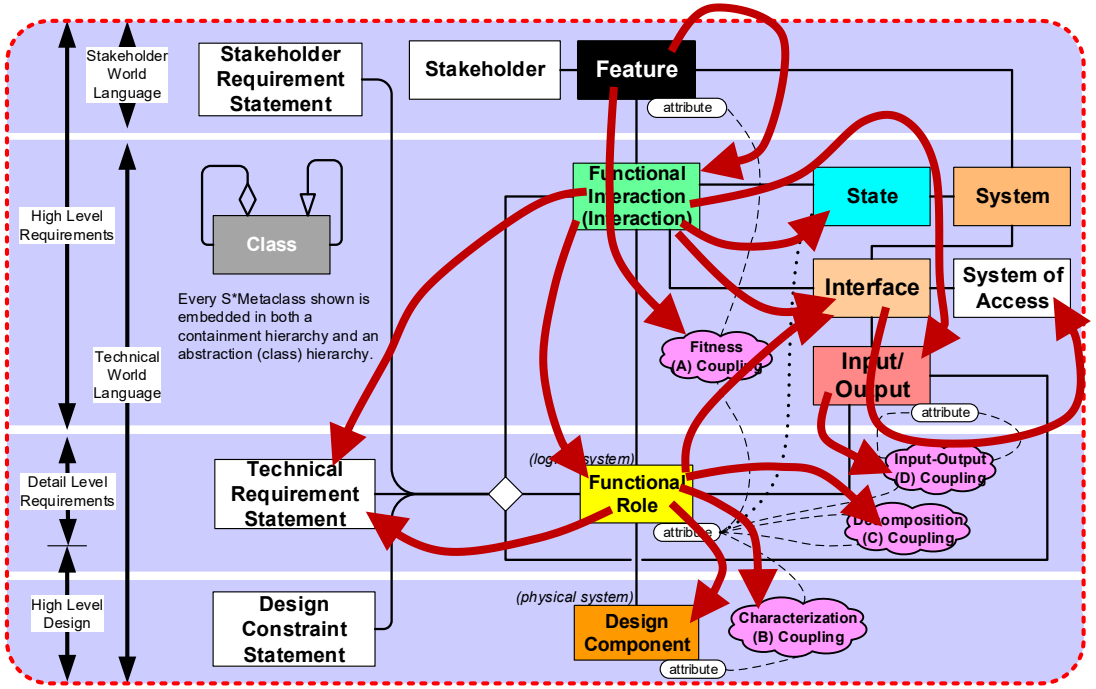
S*Pattern Configuration Algorithm: Conceptual Overview

From Session 5 class

Propagation of configuration population is inherent to the nature of all engineered systems

- S*Feature Space drives configuration from a smaller set of (stakeholder based) degrees of freedom / points of variation.
- Simplifies Product Line Engineering (PLE) model configuration rule-making and integrates PLE.

Inherent PLE degrees of freedom configuration propagation:



TRIGGERING METACLASSES ("IF")	POPULATED METACLASSES ("THEN")																								
	Feature	Interaction	Role	Design Component	Requirement Statement	State	Event	Transition	Interface	Architectural Relationship	Input/Output	Port	System of Access	Failure Impact	Counter Requirement Statement	Failure Mode	Feature Attribute	Role Attribute	Design Component Attribute	Input/Output Attribute	Fitness Attribute Coupling	Decomposition Attribute Coupling	Characterization Attribute Coupling	IO Attribute Coupling
Stakeholder Input																									
Feature																									
Interaction																									
Role																									
Design Component																									
Requirement Statement																									
State																									
Event																									
Transition																									
Interface																									
Architectural Relationship																									
Input/Output																									
Port																									
System of Access																									
Failure Impact																									
Counter Requirement Statement																									
Failure Mode																									
Feature Attribute																									
Role Attribute																									
Design Component Attribute																									
Input/Output Attribute																									
Fitness Attribute Coupling																									
Decomposition Attribute Coupling																									
Characterization Attribute Coupling																									
IO Attribute Coupling																									

Configuration Rules:
Part of an S*Pattern

Feature	FPK	Interaction	IPK Rule

Feature-Interaction Rules Table

Role	RPK	Des Comp	DCPK Rule

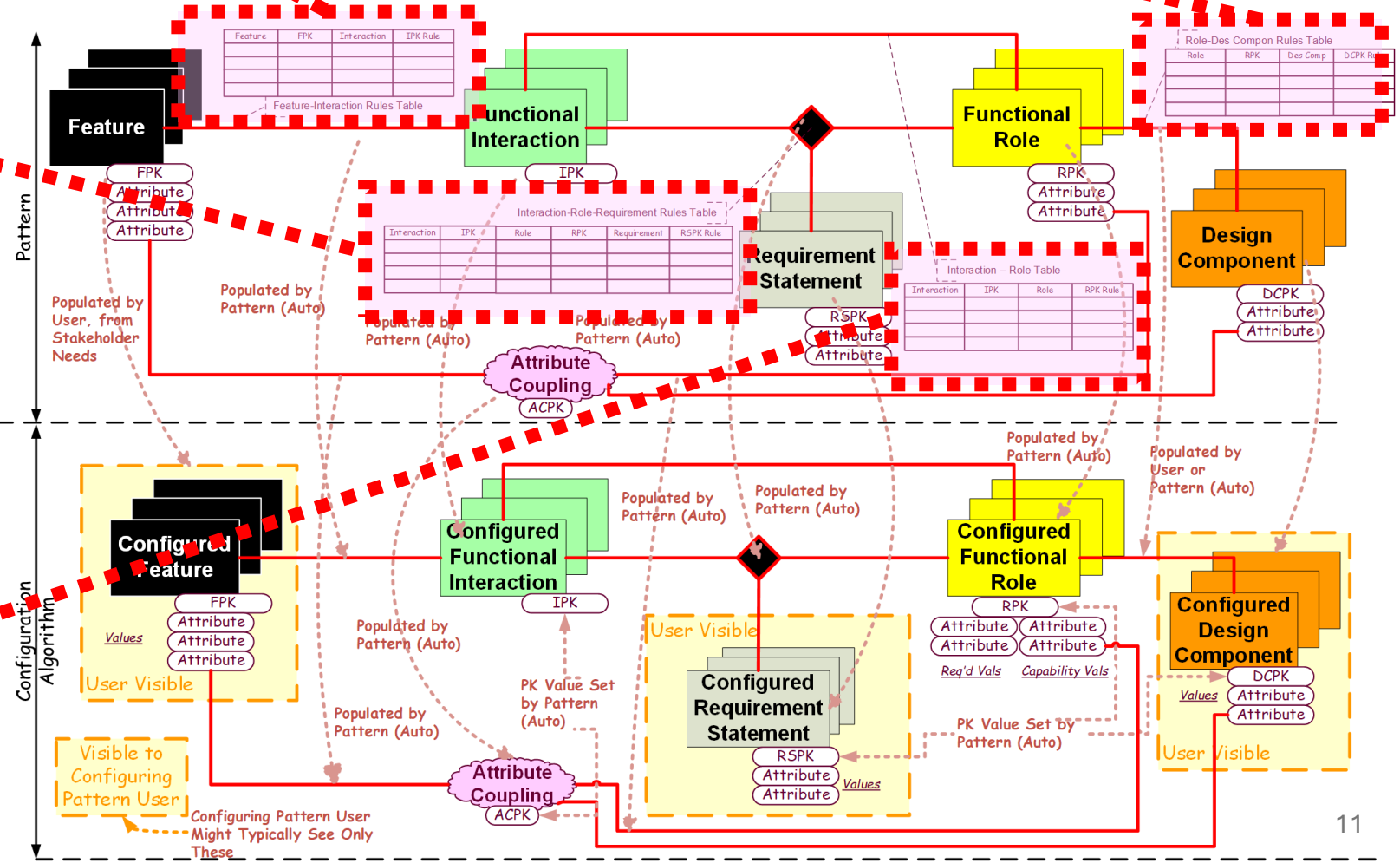
Role-Des Compon Rules Table

Interaction	IPK	Role	RPK	Requirement	RSPK Rule

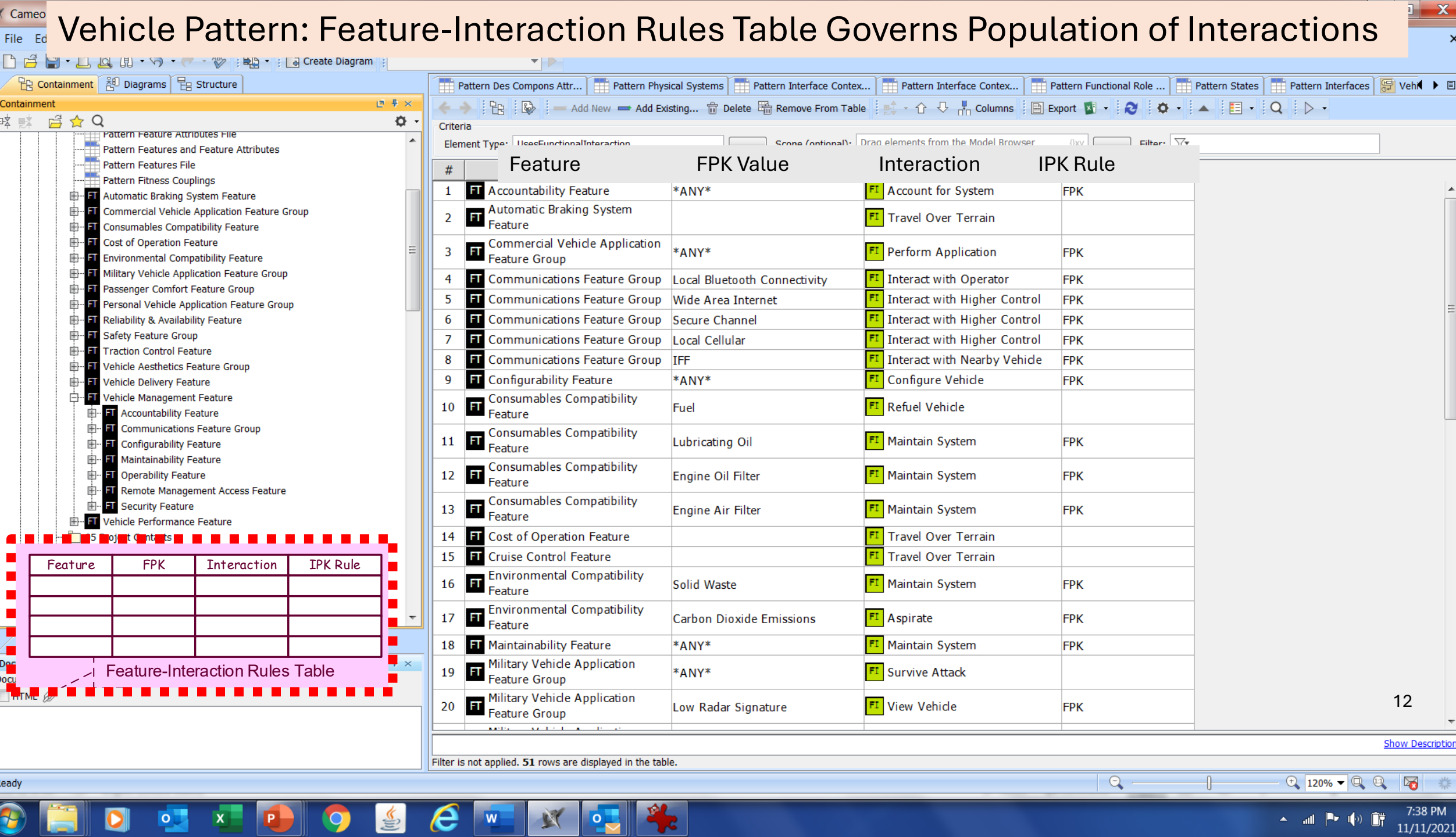
Interaction-Role-Requirement Rules Table

Interaction	IPK	Role	RPK Rule

Interaction – Role Table

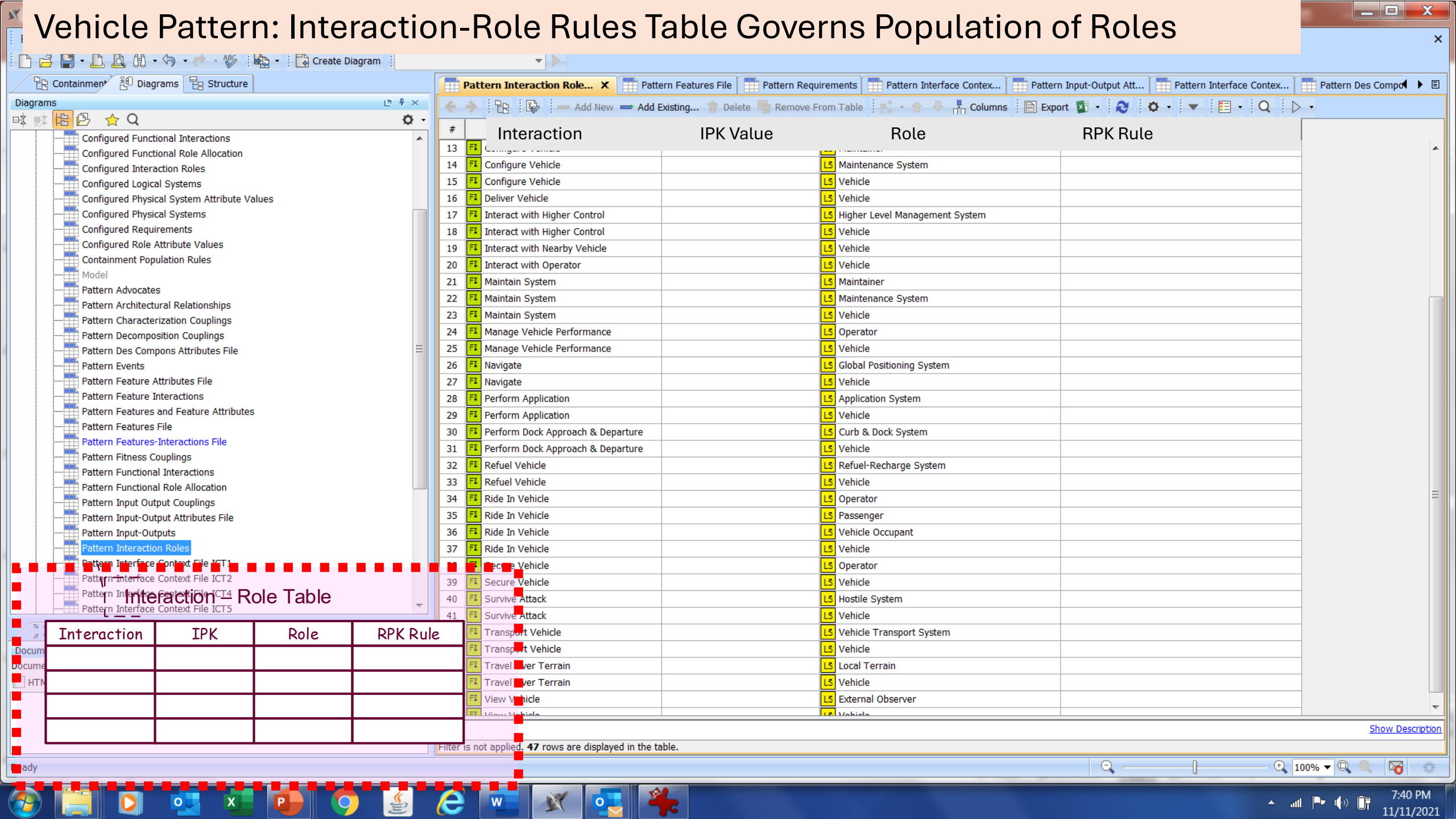


Vehicle Pattern: Feature-Interaction Rules Table Governs Population of Interactions

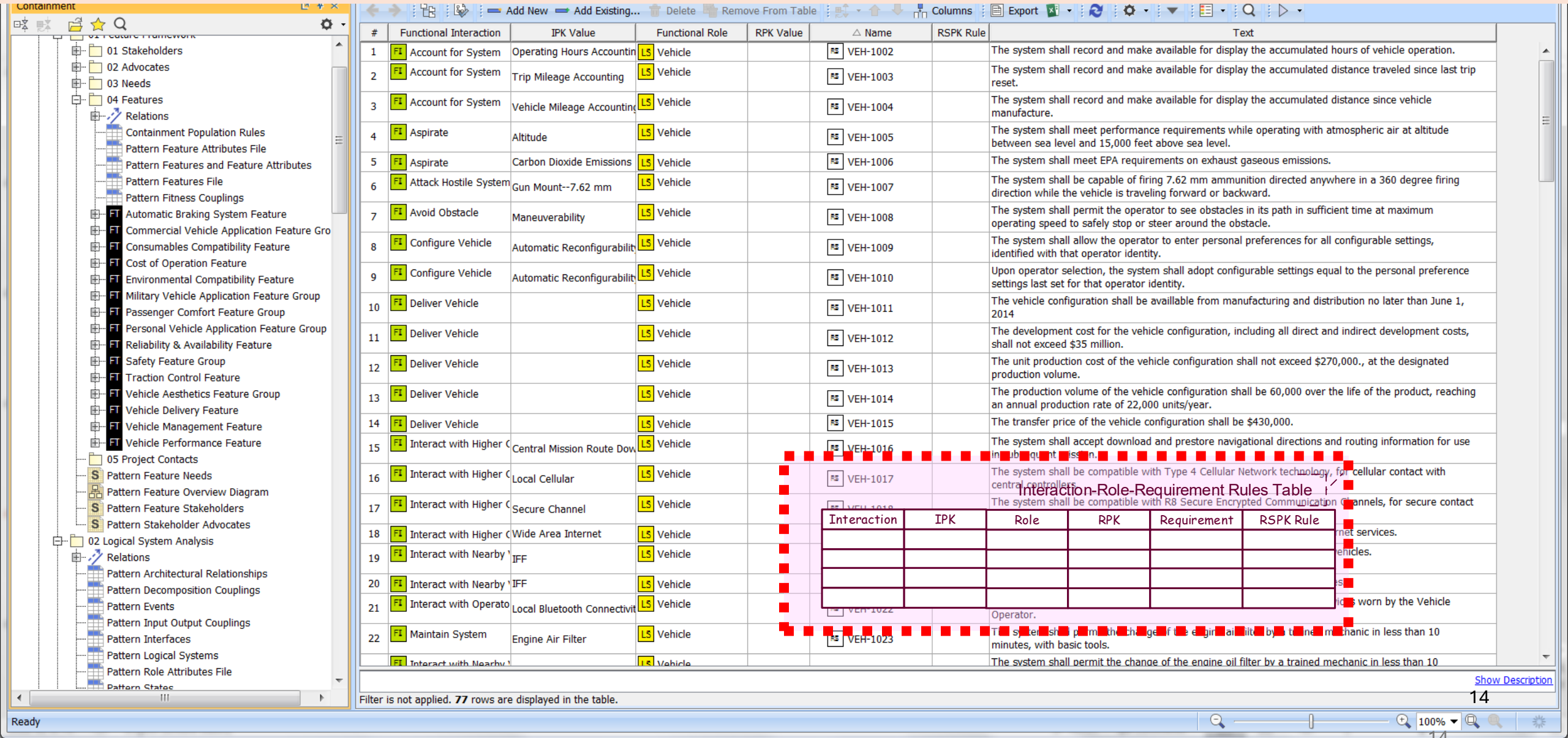


12

Vehicle Pattern: Interaction-Role Rules Table Governs Population of Roles

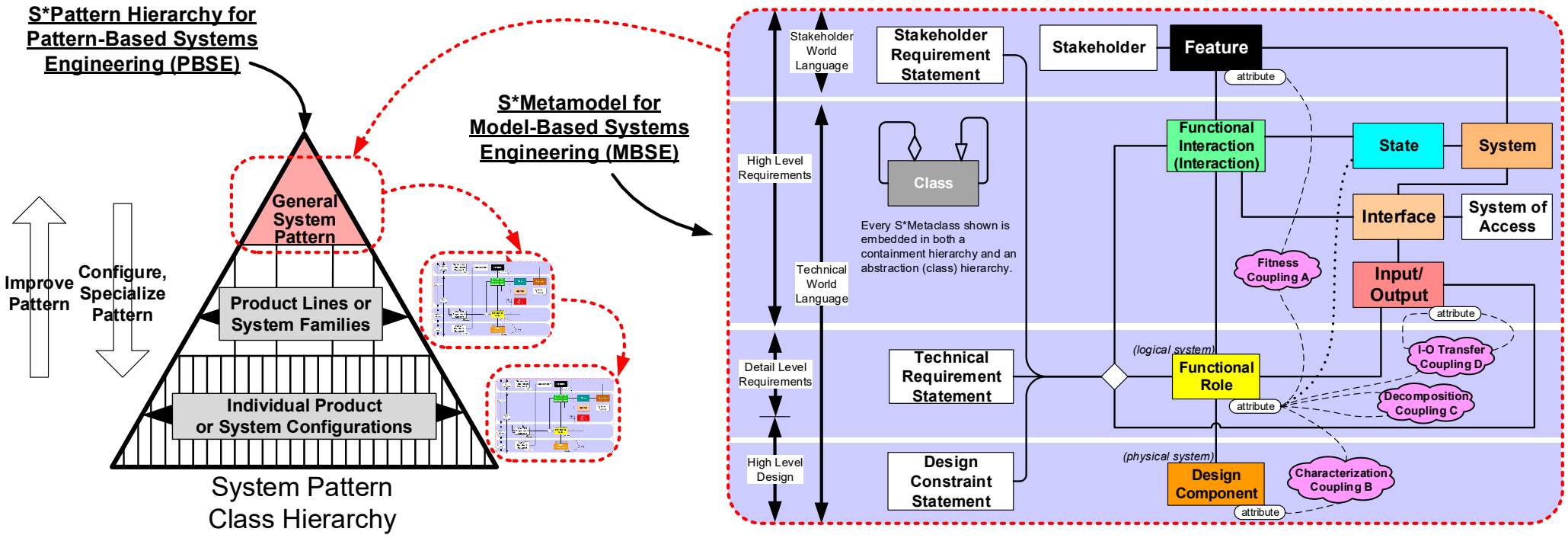


Vehicle Pattern: Interaction-Role-Requirements Table Governs Population of Requirements, based on combination "IF" of Interaction-Role pairs.



S*Pattern Configuration Algorithm

- Remember that:
  - The scope of an S*Pattern to be configured <u>may</u> cover the whole S*Metamodel: Requirements, Design, Failure Modes, etc.—or just some of those, depending on need.
    - We started with configuring just Requirements, a very useful capability!



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

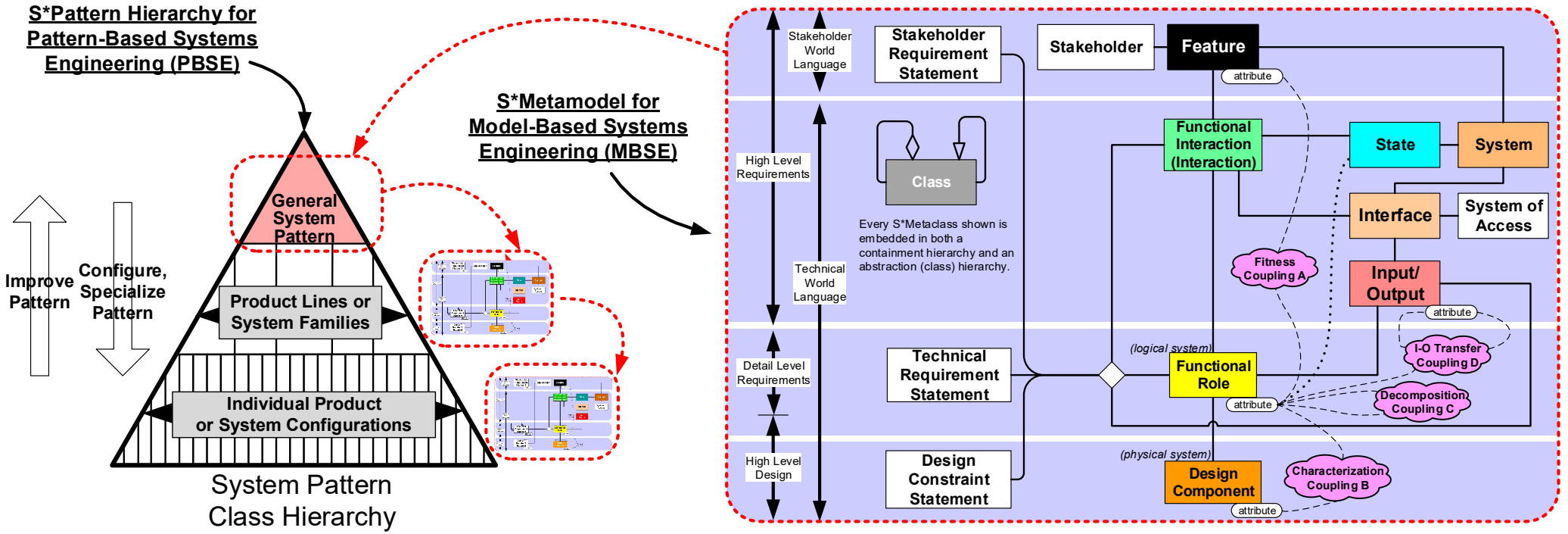
S*Pattern Configuration Algorithm

"Configure" a pattern means:

A. Populating (often multiple Model instances from one Pattern instance) and Depopulating model elements from the pattern: Classes, Relationships, Attributes, per the pattern's configuration rules, and . . .

B. Setting the Values of Attributes, per the configuration rules.

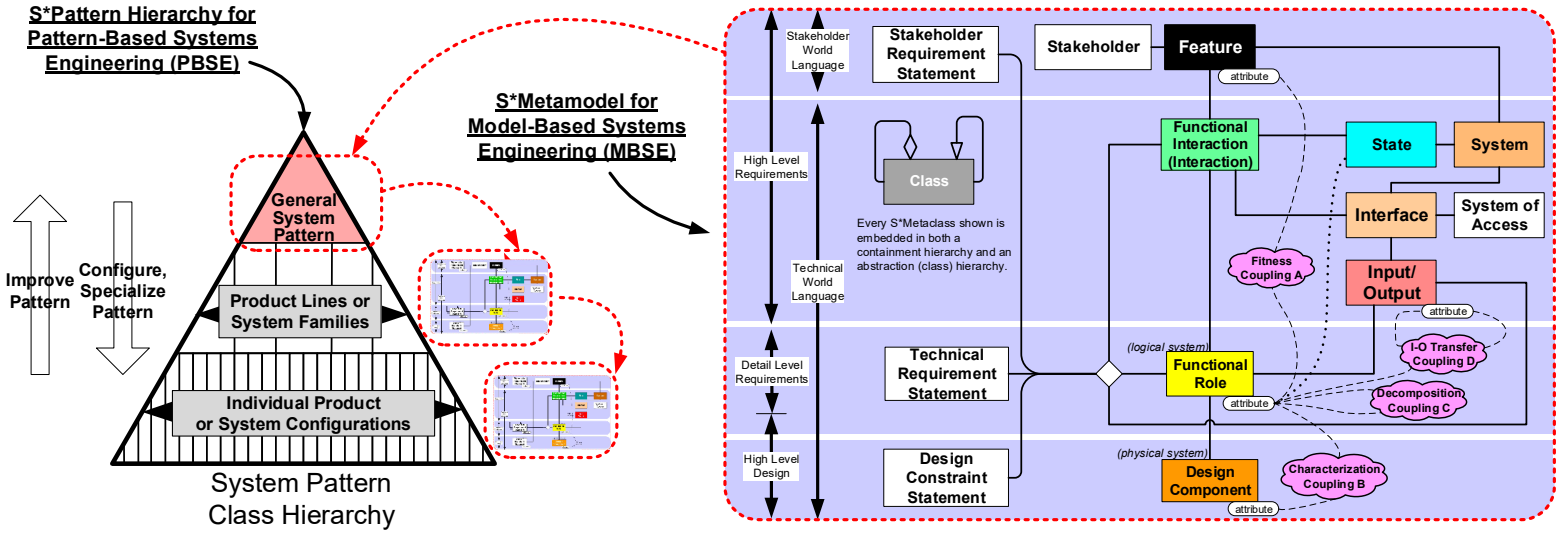
(A) above typically makes configured models much "bigger" than the compressed patterns they come from!



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

S*Pattern Configuration Algorithm

- In this example we will be:
  - Using an S*Pattern and resulting configured S*Model in SysML that has been mapped to S*Metamodel—but it could have been in other modeling languages or tools, by similar methods.
  - Configuring an engineered product (referred to as System 1 in the Ecosystem reference framework), but it could instead have been configuring an engineering or other life cycle management process (called System 2 in that same framework).
- They are all S*Model and S*Patterns, but describe different kinds of systems.



S*Metamodel informal summary pedagogical diagram
(formal S*Metamodel includes additional details.)

Automating the Algorithm: Configuration Agents

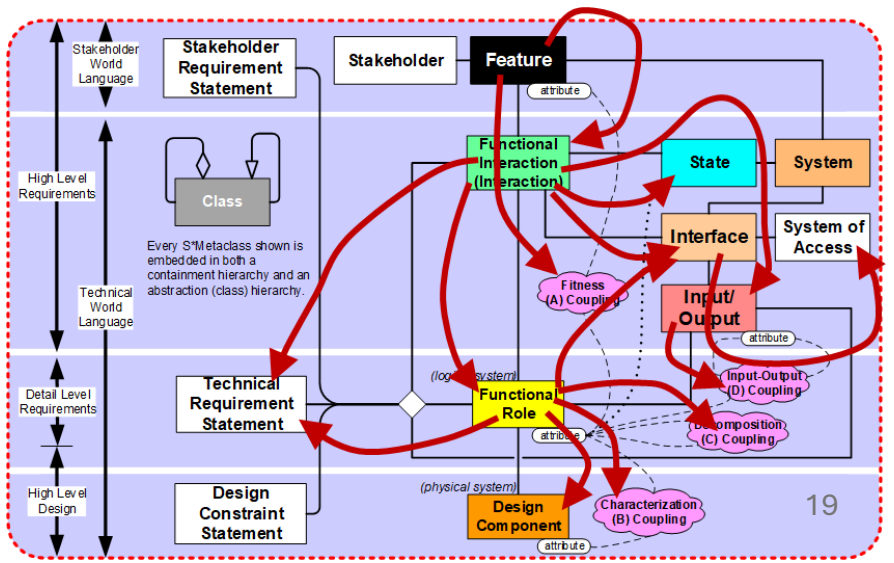
- Early emphasis was on Configurable S*Requirements Patterns, for rapid generation of high quality system Requirements—applied to Advanced Manufacturing Systems, Medical Devices, Consumer Products, etc.
- Later expanded to configuring High Level Designs, including FMEA risk analysis patterns.
- Most recently (via the INCOSE Interface Patterns Project and its follow-on Semantic Technologies for Systems Engineering (ST4SE) Project), expanding configuration rules scope to details of Interfaces, and the Interface Context.
- This led to updating the S*Metamodel's representation of Interface Context.
- Currently updating the configuration rules capabilities for scalability of consistency management, including bulk loading of instances describing high level system design.

Automating the Algorithm: Configuration Agents

- Over ten years of different automated agents, executing the underlying configuration algorithm, using various technologies and with third party COTS modeling tools:
  - S*Patterns generally portable across the agent

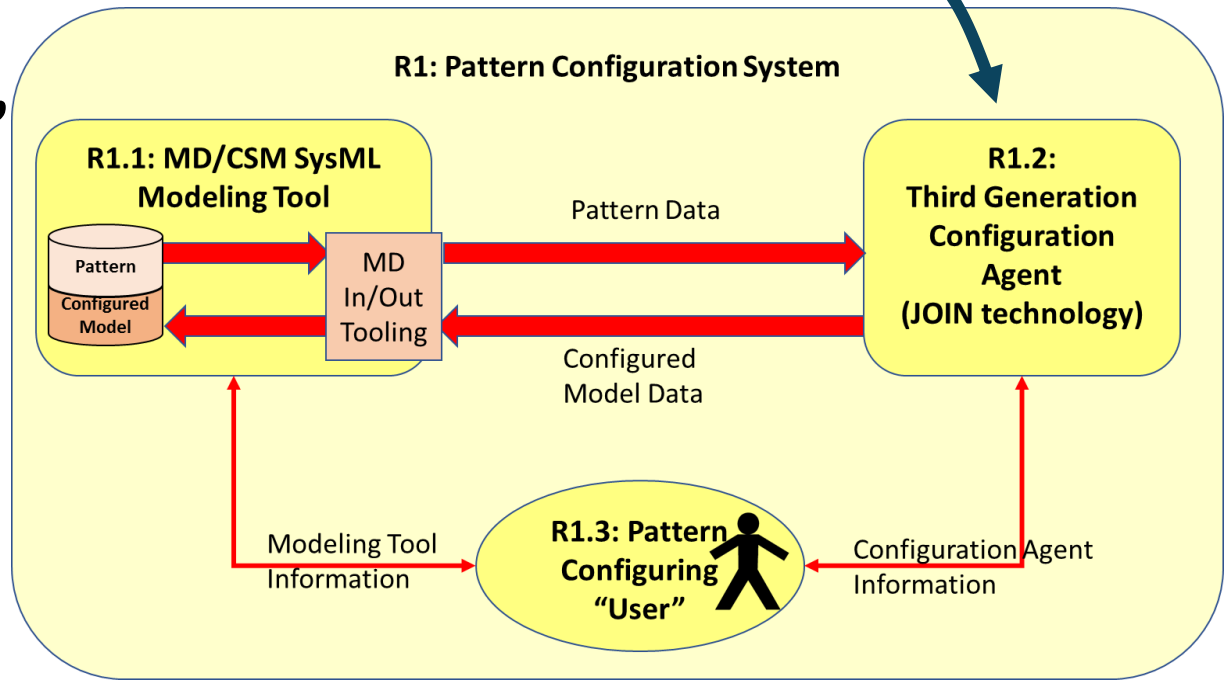
When	Configuration Agent Generation	Compatible Modeling Tools	Scalability
First generation, 2005	SE Pattern Workbook: Based in desktop spreadsheet automation for self-contained S*Patterns and S*Models.	Self-contained	Small and medium-scale models and patterns.
Second generation, 2013	SE Pattern Agent: Desktop agent software integrated as add-on to commercial third party MBSE tooling across multiple COTS tools.	Models and patterns in DOORS®, Requisite Pro®, Enterprise Architect SysML®, Magic Draw / Cameo Systems Modeler SysML®	Small and medium-scale models and patterns.
Third generation, 2021	Configuration Wizard: Based in desktop or server bulk JOIN software integrated as add-on to commercial third party MBSE tooling across multiple toolsets.	Generic OMG SysML®, Dassault Cameo Systems Modeler® SysML, Sparx Enterprise Architect® SysML.	Medium and large-scale models and patterns.

TRIGGERING METACLASSES ("IF") \ POPULATED METACLASSES ("THEN")	Feature	Interaction	Role	Design Component	Requirement Statement	State	Event	Transition	Interface	Architectural Relationship	Input/Output	Port	System of Access	Failure Impact	Counter Requirement Statement	Failure Mode	Feature Attribute	Role Attribute	Design Component Attribute	Input/Output Attribute	Fitness Attribute Coupling	Decomposition Attribute Coupling	Characterization Attribute Coupling	IO Attribute Coupling
Stakeholder Input	■																							
Feature	■	■			■	■	■		■	■	■	■	■	■			■							
Interaction		■	■		■	■	■		■	■	■	■	■											
Role			■		■	■			■	■	■	■	■					■						
Design Component				■													■		■					
Requirement Statement					■										■									
State						■	■																	
Event							■																	
Transition								■																
Interface									■															
Architectural Relationship										■														
Input/Output											■									■				
Port												■												
System of Access													■											
Failure Impact														■										
Counter Requirement Statement															■									
Failure Mode																■								
Feature Attribute																	■				■			
Role Attribute																		■			■	■		
Design Component Attribute																			■					
Input/Output Attribute																				■				■
Fitness Attribute Coupling																					■			
Decomposition Attribute Coupling																						■		
Characterization Attribute Coupling																							■	
IO Attribute Coupling																								■



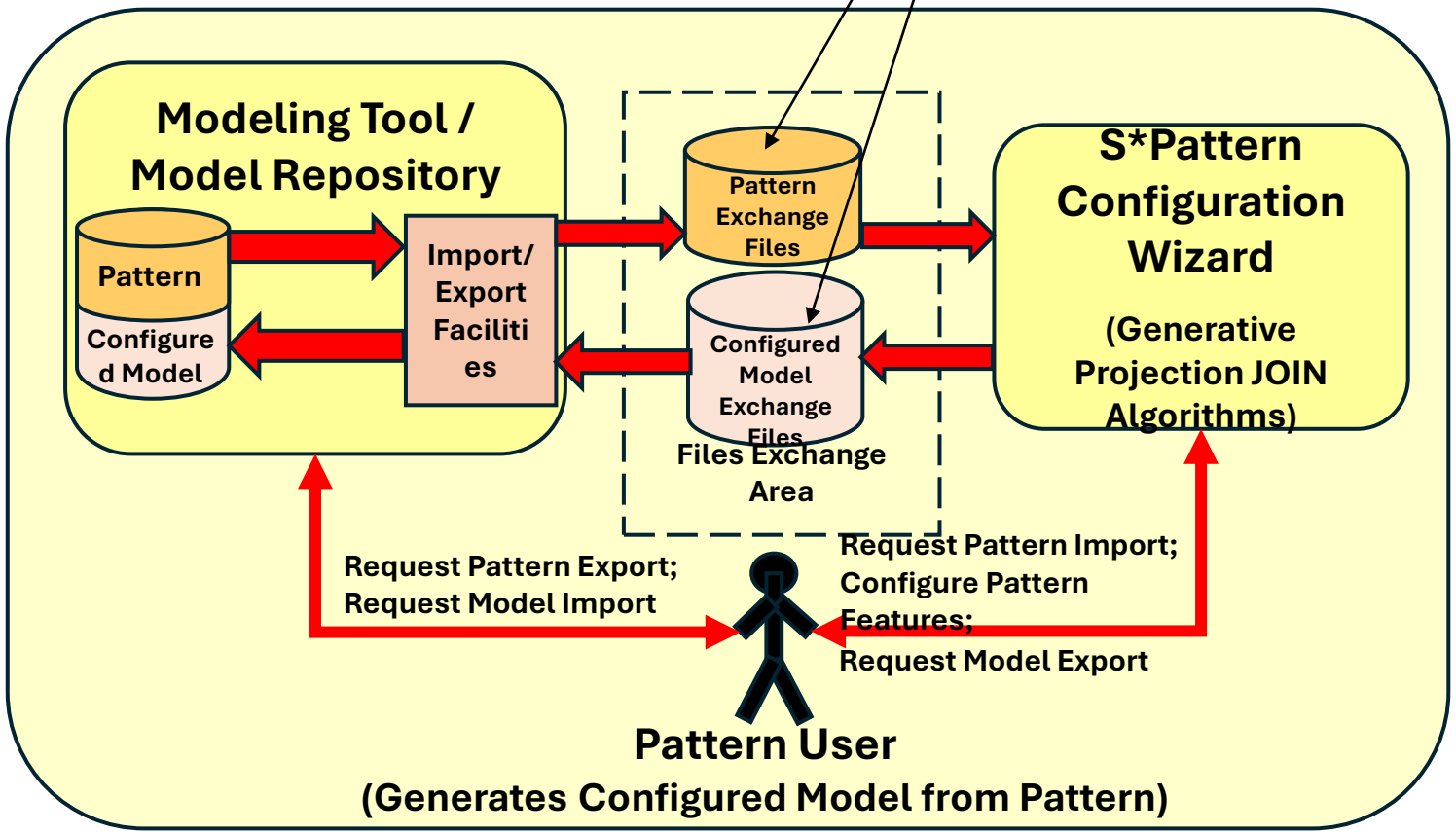
Automating the Algorithm: Configuration Agents

- The latest (third generation) automation uses JOIN technology, because of the inherent JOIN/Projection nature of what it means to generate a high dimension configured model from a low dimension compressed pattern.

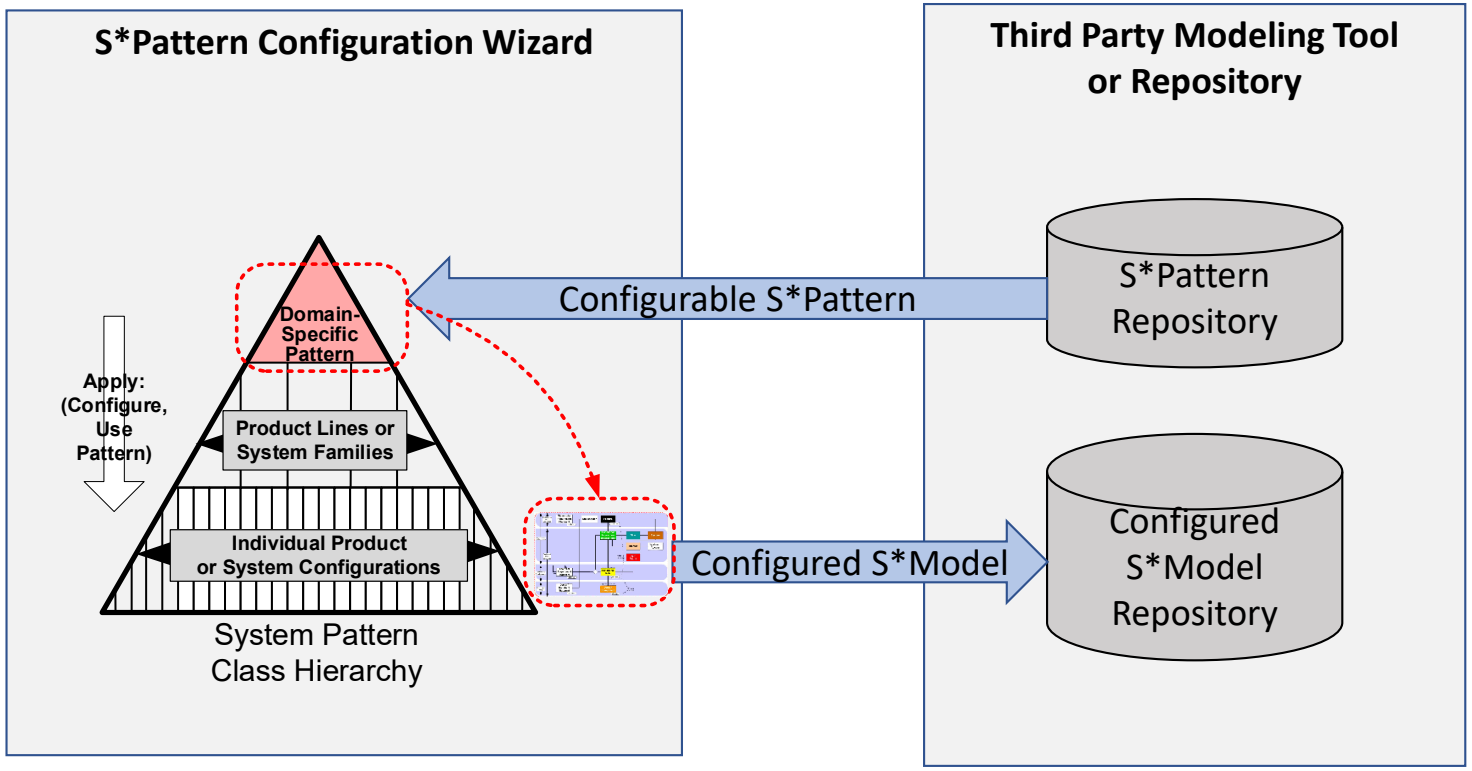


Pattern and Configured Model Segments:

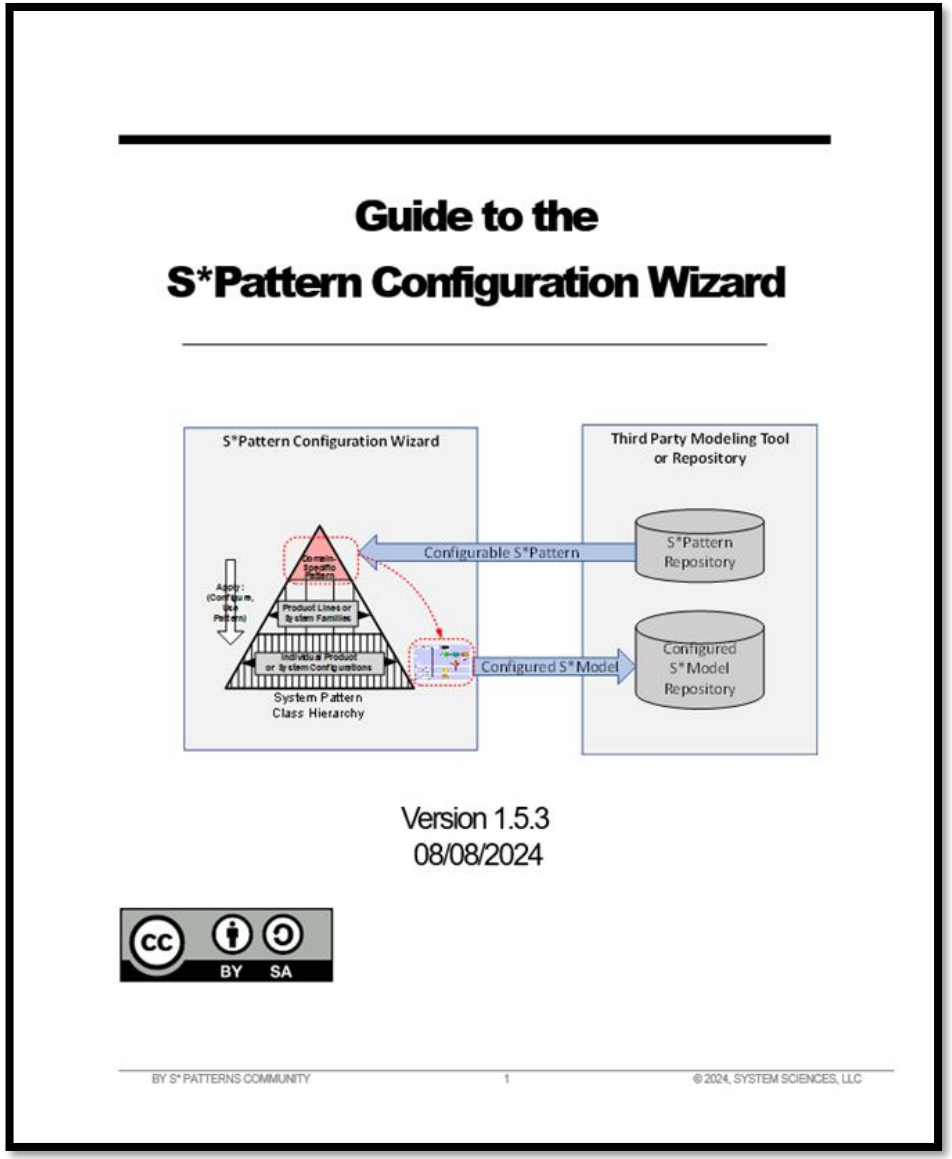
- HLR/DLR/HLD (mandatory base)
- Interfaces (optional)
- States/Modes (optional)
- Parametric Attribute Couplings (optional)
- Risk Analysis / FMEA (optional)
- Bulk Consistency Management (optional)



Third Generation Configuration Agent, Using JOIN Technology: The Configuration Wizard



IMPORTANT REFERENCE TO READ



Guide to the

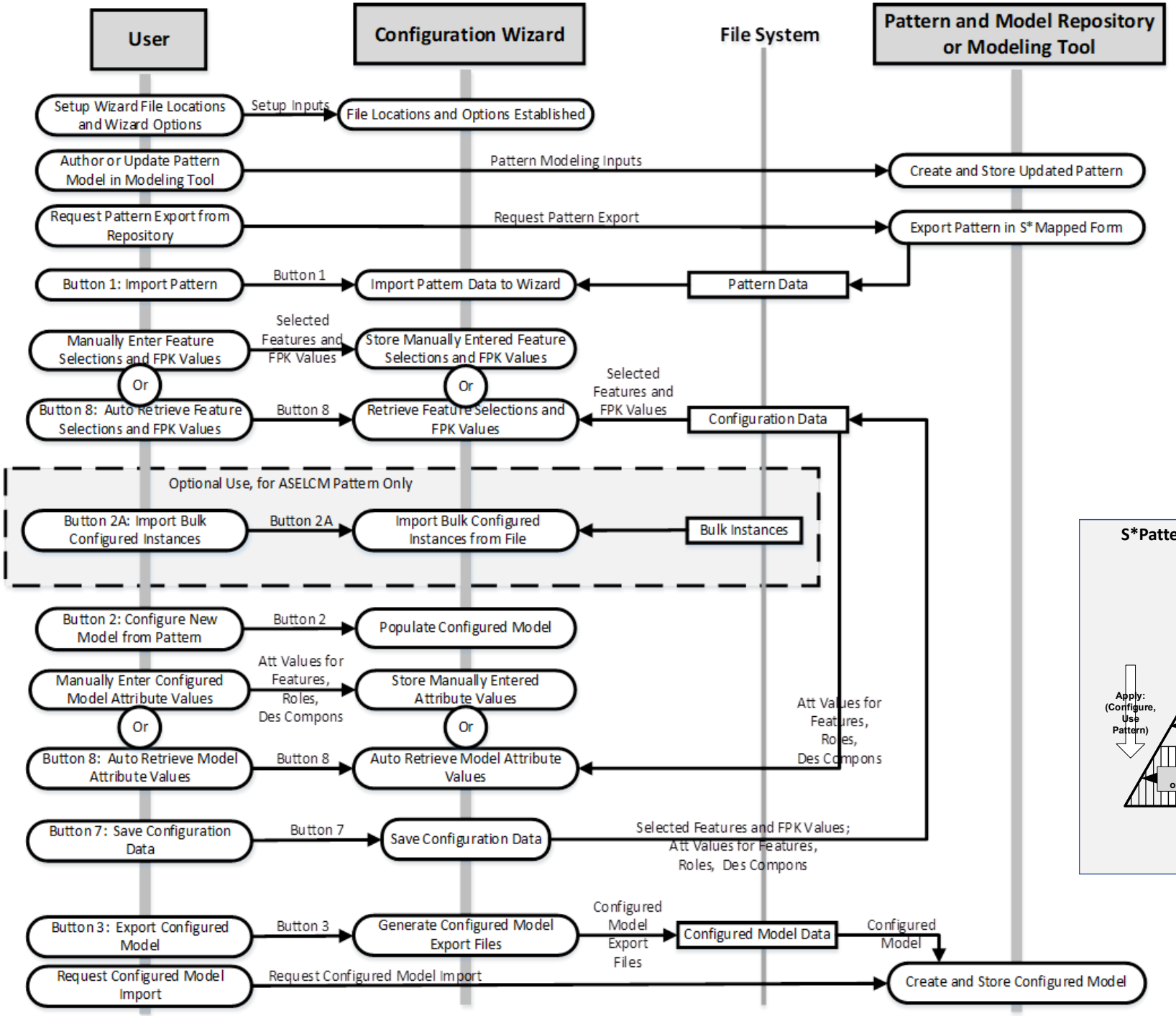
S*Pattern Configuration Wizard

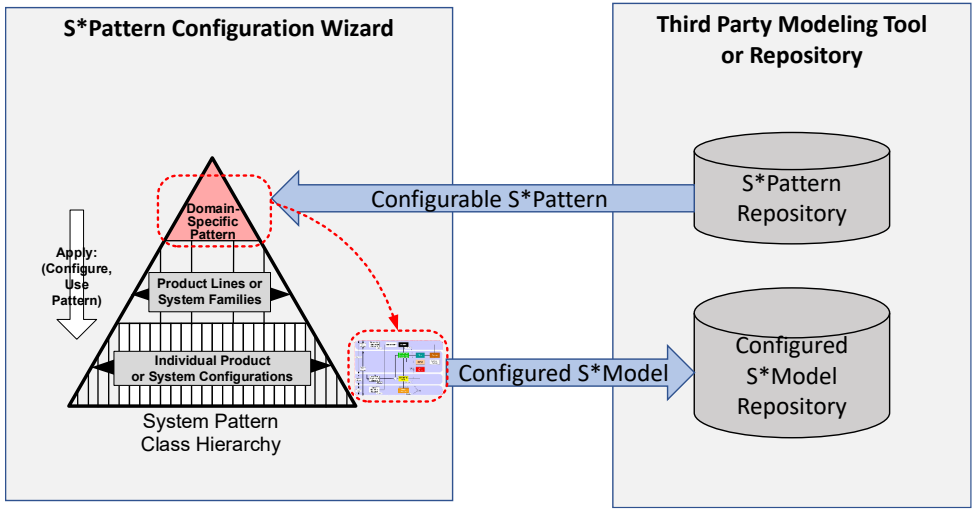
Version 1.5.3
08/08/2024

CC BY SA

https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:guide_to_the_pattern_configuration_wizard_v1.5.4.pdf

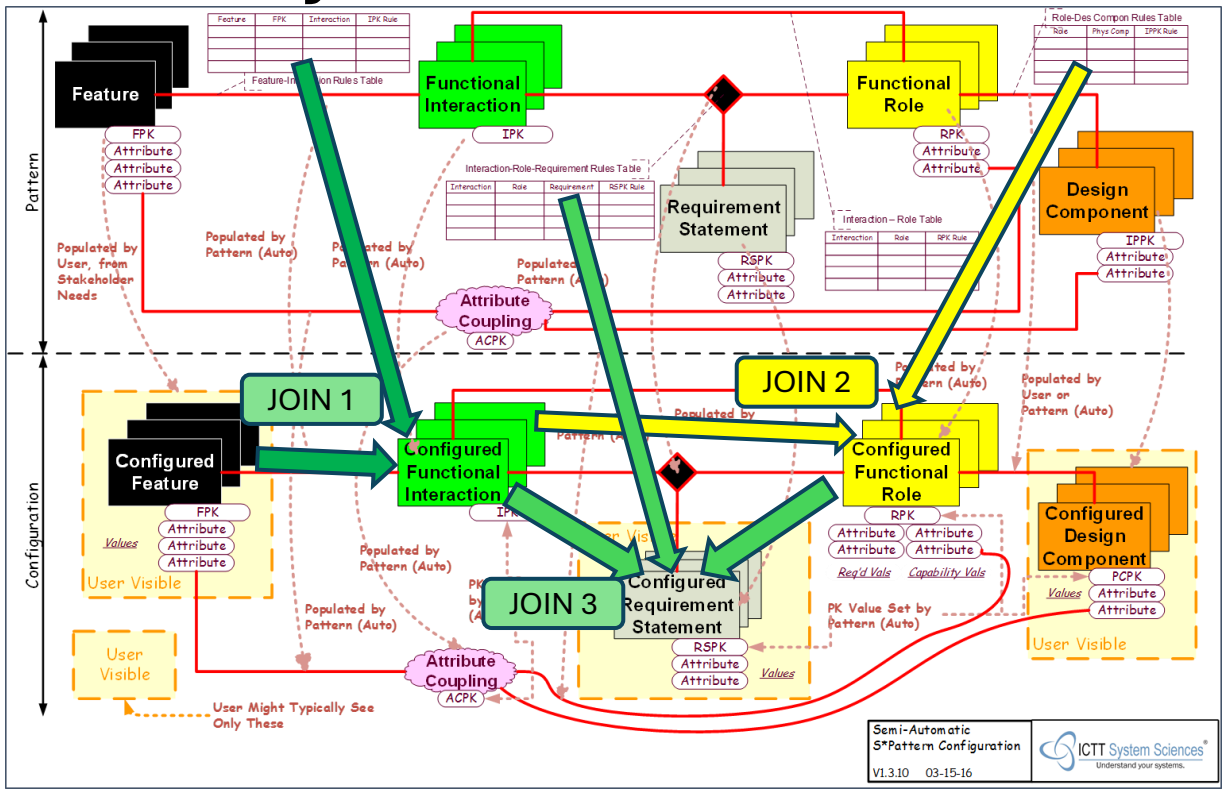
Overview of Typical Use of Configuration Wizard





Why “JOIN” technology? What is it? Why?

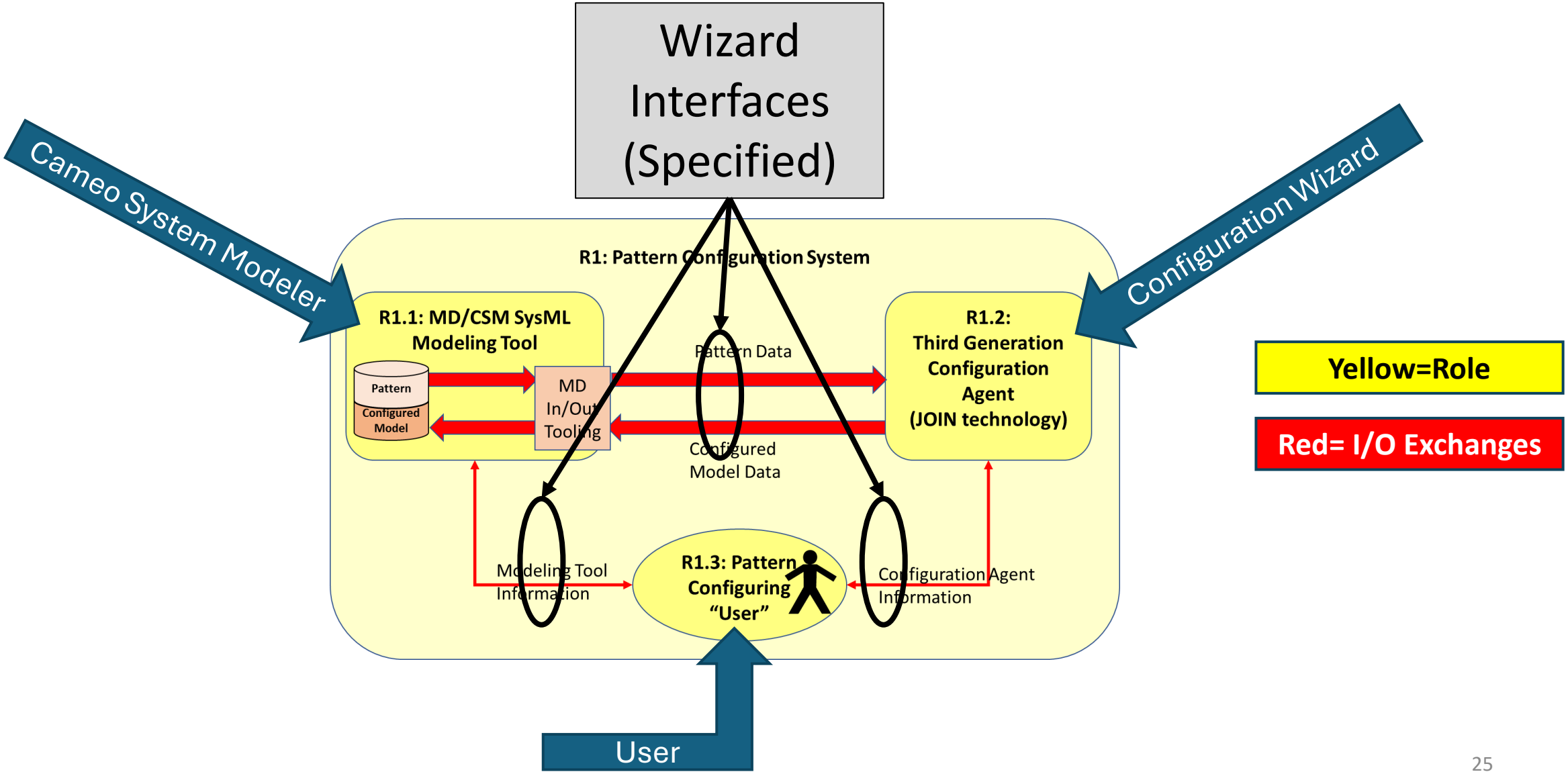
- Originating in RDBMS SQL technology, but also provided for in Semantic Technologies (e.g., SPARQL).
- A simple way to think of a JOIN is creating a new set of data by pair-wise combinations of two sets of data, based on the similarity of some of their properties
- Implements learned pattern-based generative inference.



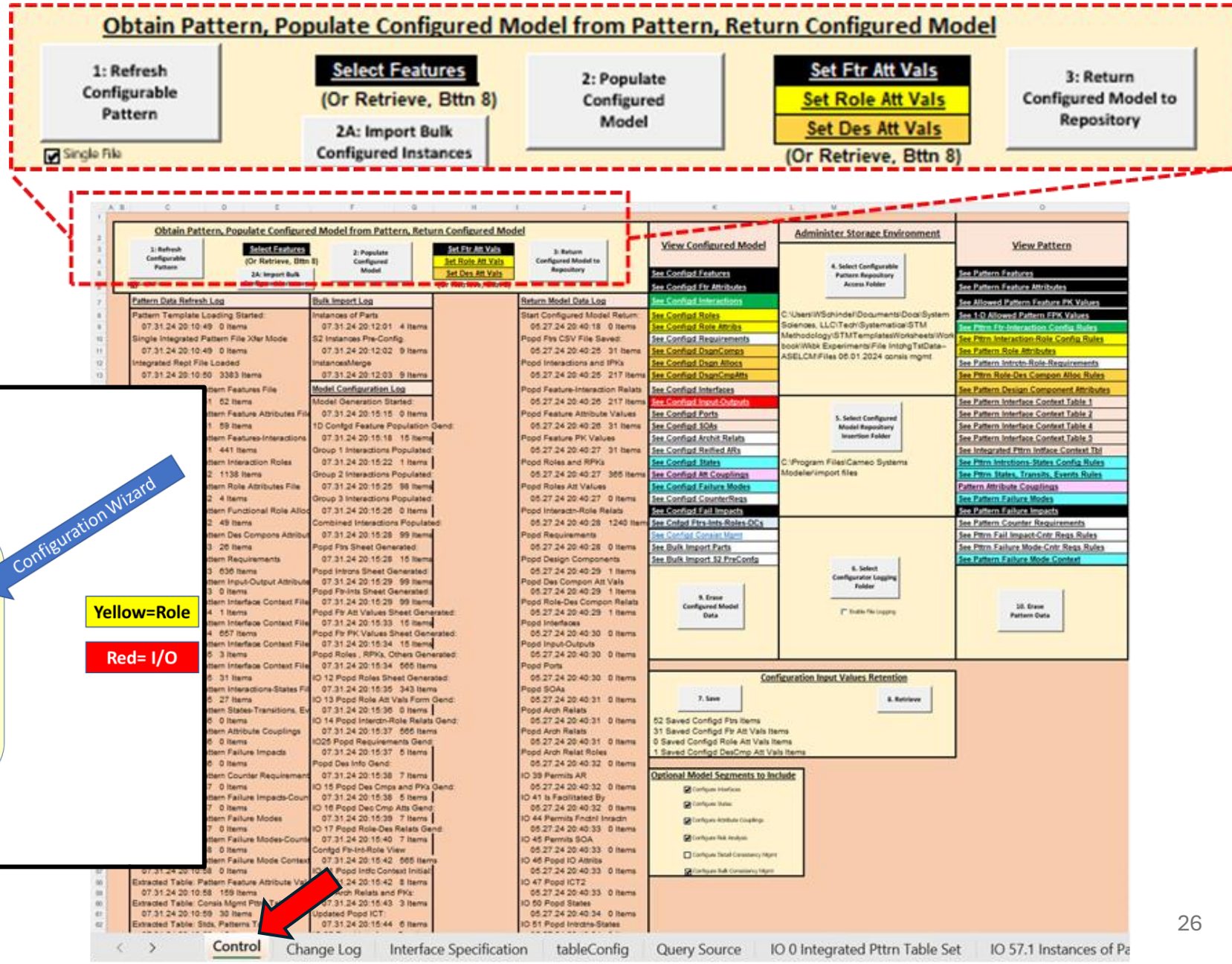
- In expanding a pattern into a populated model, the repeating JOIN paradigm is that we are comparing [a part of the pattern] with [some already populated model data], to generate some [additional populated model data]:
  - [Pattern Feature Rules] JOIN with [Stakeholder Selections] → [Populated Features]
  - [Pattern Feature-Interaction Rules] JOIN with [Populated Features] → [Populated Interactions]
  - [Pattern Interaction-Role Rules] JOIN with [Populated Interactions] → [Populated Roles]
  - [Pattern Interaction-Role-Requirement Rules] JOIN with [Populated Interactions & Roles] → [Populated Requirements]

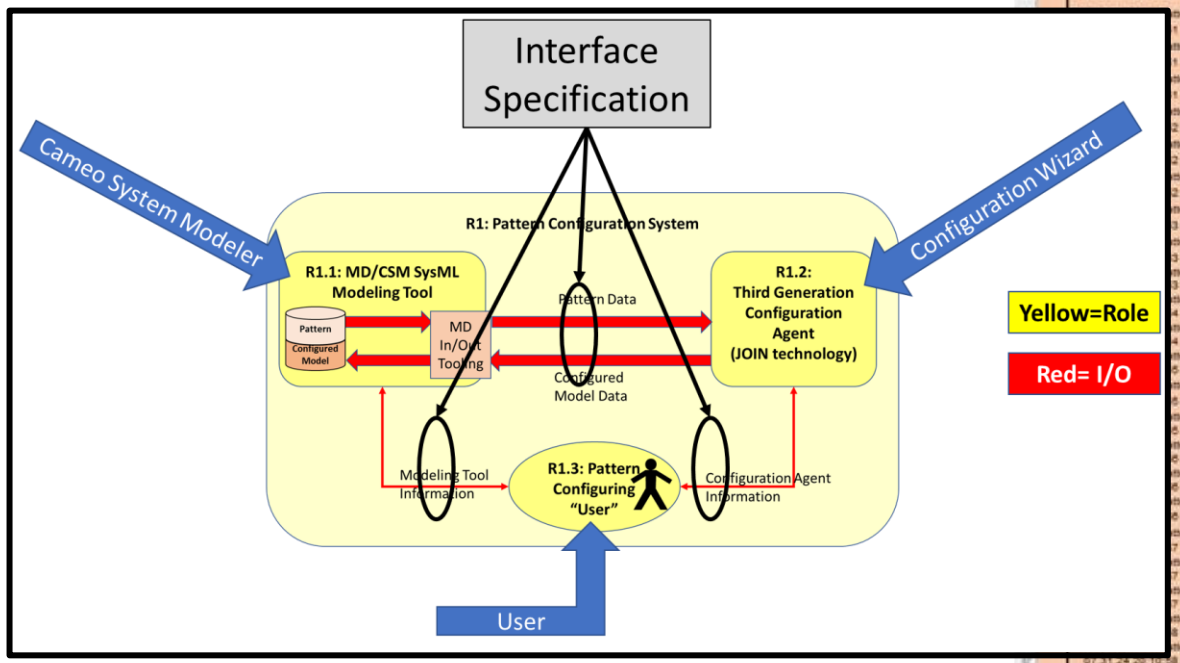
This can populate a much larger model (more instances) than the parent pattern!

Three Interacting Entities: Modeling Tool-Configuration Agent-User



Wizard User's Control Interface and Logging Tab

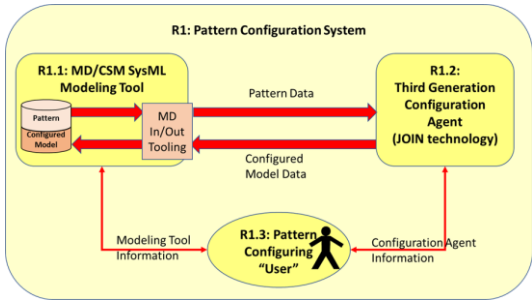


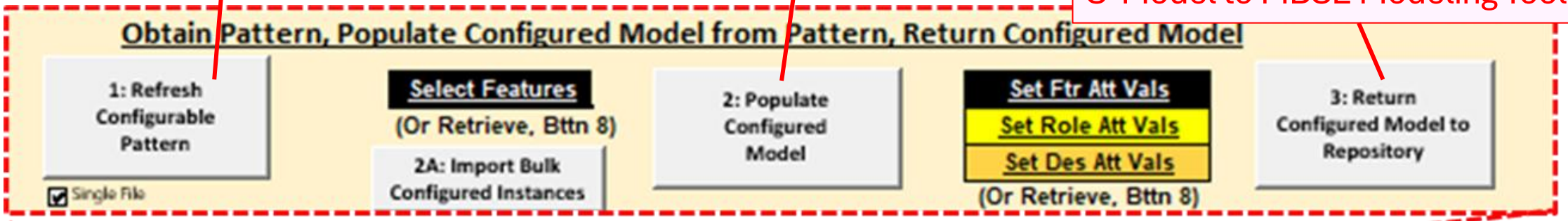


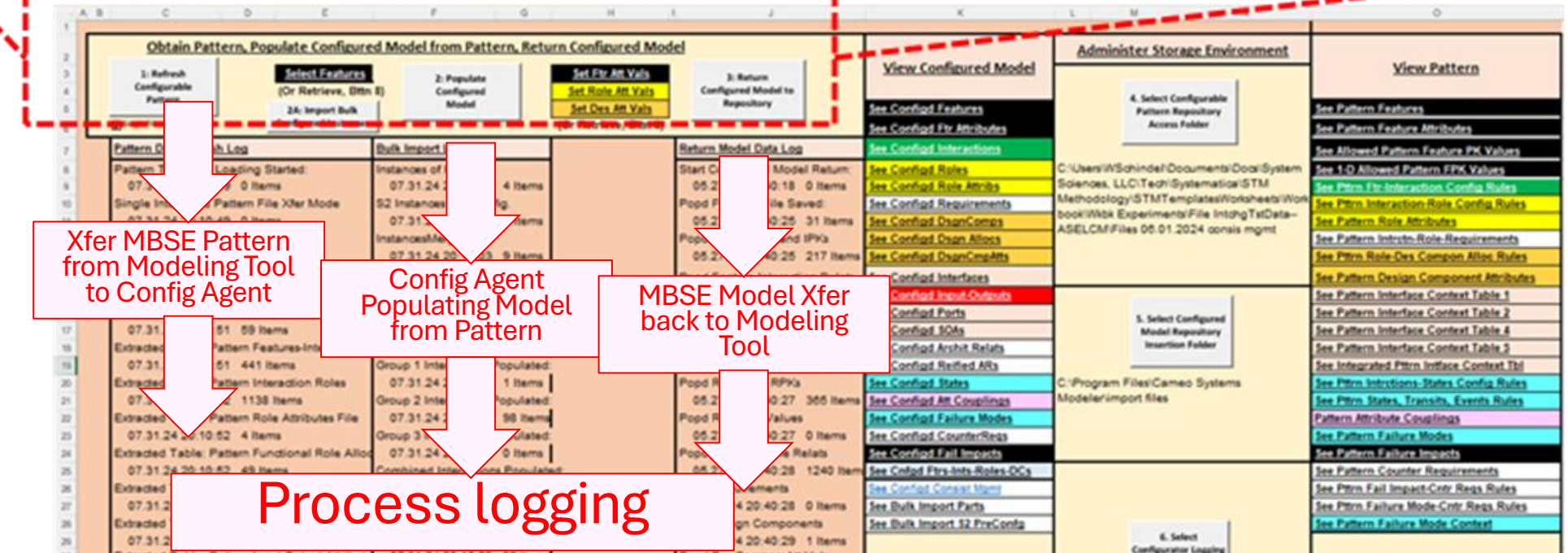
Button 1: Retrieve S*Pattern from MBSE Modeling Tool

Button 2: After user enters Feature configuration data, generate Configured S*Model

Button 3: Return Configured S*Model to MBSE Modeling Tool







Xfer MBSE Pattern from Modeling Tool to Config Agent

Config Agent Populating Model from Pattern

MBSE Model Xfer back to Modeling Tool

Process logging

Priming the Pump:
Wizard User's Feature Selection Interface (Including Feature Primary Key Value Population)

	Feature Configuratic	Feature Name	Feature Attribute	Populate? Yes/No	Selection 1	Selection 2	Selection 3	Selection 4	Selection 5	Selection 6
1										
9	Mandatory	Cruise Control Feature								
10	Optional	Environmental Compatibility	Environmental Issue							
11	Mandatory	Maintainability Feature	Maintenance Capability							
12	Optional	Military Vehicle Application Feature Group	Military Application Type							
13	Optional	Navigation Feature	Navigation Capability							
14	Mandatory	Operability Feature	Operations Capability	Yes	Automatic Performance Data Measurement and Display	Automatic Performance Threshold Detection and Reporting	Maneuverability			
15	Optional	Passenger Comfort Feature Group	Comfort Issue				Automatic Performan			
16	Optional	Personal Vehicle Application Feature Group	Personal Application Type				Maneuverability			
17	Mandatory	Reliability & Availability Feature					Manual Performance			
18	Optional	Remote Management Access	Remote Access Capability				Manual Performance			
19	Optional	Remote-Autonomous Operation		No			Operations Procedure			
20	Mandatory	Safety Feature Group					Visibility			
21	Optional	Security Feature	Security Management Capability	No	Automatic Operational Privileges Authorization	Identification and Authentication	Physical Access Locks	Security Data Management		
22	Optional	Traction Control Feature		No						
23	Optional	Vehicle Aesthetics Feature Group	Aesthetics Issue	No	Exterior Body Style	Exterior Color Galeon Blue	Exterior Color Handon Green	Interior Color Rich Brown	Interior Color Sand Dune	Overall Passenger
24	Mandatory	Vehicle Delivery Feature		No						
25		Vehicle Management Feature		No						
26	Mandatory	Vehicle Performance Feature		No						

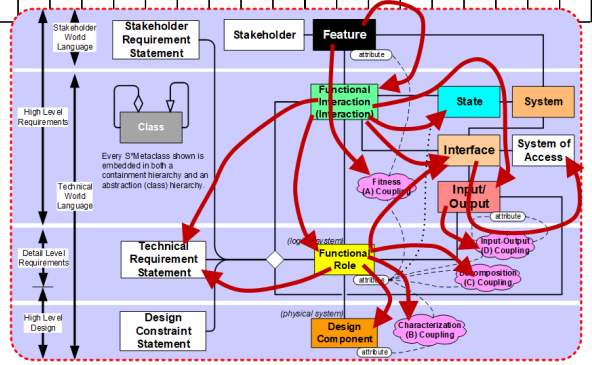
Fixed Pattern Menu of Features and FPK Attributes

Variable User Selections of Features and Populated Feature PK Values

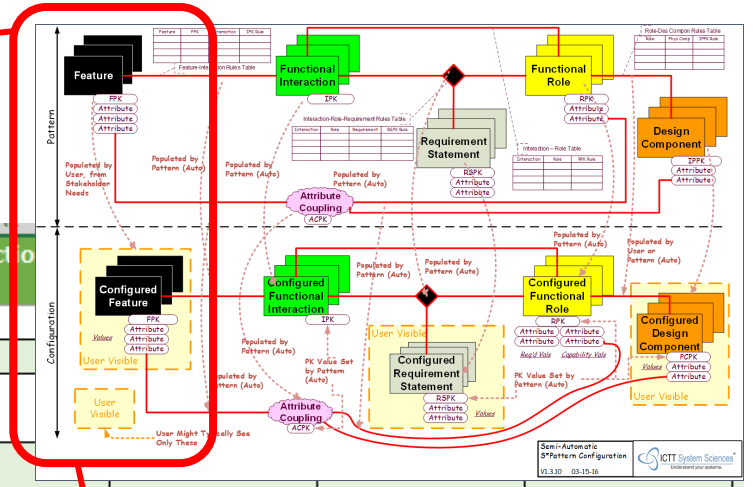
POPULATED METACLASSES ("THEN")

TRIGGERING METACLASSES ("IF")	Feature	Interaction	Role	Design Component	Requirement Statement	State	Event	Transition	Interface	Architectural Relationship	Input/Output	Port	System of Access	Failure Impact	Counter Requirement Statement	Failure Mode	Feature Attribute	Role Attribute	Design Component Attribute	Input/Output Attribute	Fitness Attribute Coupling	Decomposition Attribute Coupling	Characterization Attribute Coupling	IO Attribute Coupling
Stakeholder Input																								
Feature																								
Interaction																								
Role																								
Design Component																								
Requirement Statement																								
State																								
Event																								
Transition																								
Interface																								
Architectural Relationship																								
Input/Output																								
Port																								
System of Access																								
Failure Impact																								
Counter Requirement Statement																								
Failure Mode																								
Feature Attribute																								
Role Attribute																								
Design Component Attribute																								
Input/Output Attribute																								
Fitness Attribute Coupling																								
Decomposition Attribute Coupling																								
Characterization Attribute Coupling																								
IO Attribute Coupling																								

Auto Propagation



Wizard User's Feature Selection Interface (Including Feature Primary Key Value Population)



	Feature Configuratic	Feature Name	Feature Attribute	Populate? Yes/No	Selection 1	Selection 2	Selec			
9	Mandatory	Cruise Control Feature								
10	Optional	Environmental Compatibility	Environmental Issue							
11	Mandatory	Maintainability Feature	Maintenance Capability							
12	Optional	Military Vehicle Application Feature Group	Military Application Type							
13	Optional	Navigation Feature	Navigation Capability							
14	Mandatory	Operability Feature	Operations Capability	Yes	Automatic Performance Data Measurement and Display	Automatic Performance Threshold Detection and Reporting	Maneuverability			
15	Optional	Passenger Comfort Feature Group	Comfort Issue							
16	Optional	Personal Vehicle Application Feature Group	Personal Application Type							
17	Mandatory	Reliability & Availability Feature								
18	Optional	Remote Management Access	Remote Access Capability							
19	Optional	Remote-Autonomous Operation		No						
20	Mandatory	Safety Feature Group								
21	Optional	Security Feature	Security Management Capability	No	Automatic Operational Privileges Authorization	Identification and Authentication	Physical Access Locks	Security Data Management		
22	Optional	Traction Control Feature		No						
23	Optional	Vehicle Aesthetics Feature Group	Aesthetics Issue	No	Exterior Body Style	Exterior Color Galeon Blue	Exterior Color Handon Green	Interior Color Rich Brown	Interior Color Sand Dune	Overall Passenger
24	Mandatory	Vehicle Delivery Feature		N						
25		Vehicle Management Feature								
26	Mandatory	Vehicle Performance Feature								

Automatic Performan
Automatic Performan
Maneuverability
Manual Performance
Manual Performance
Operations Procedure
Visibility

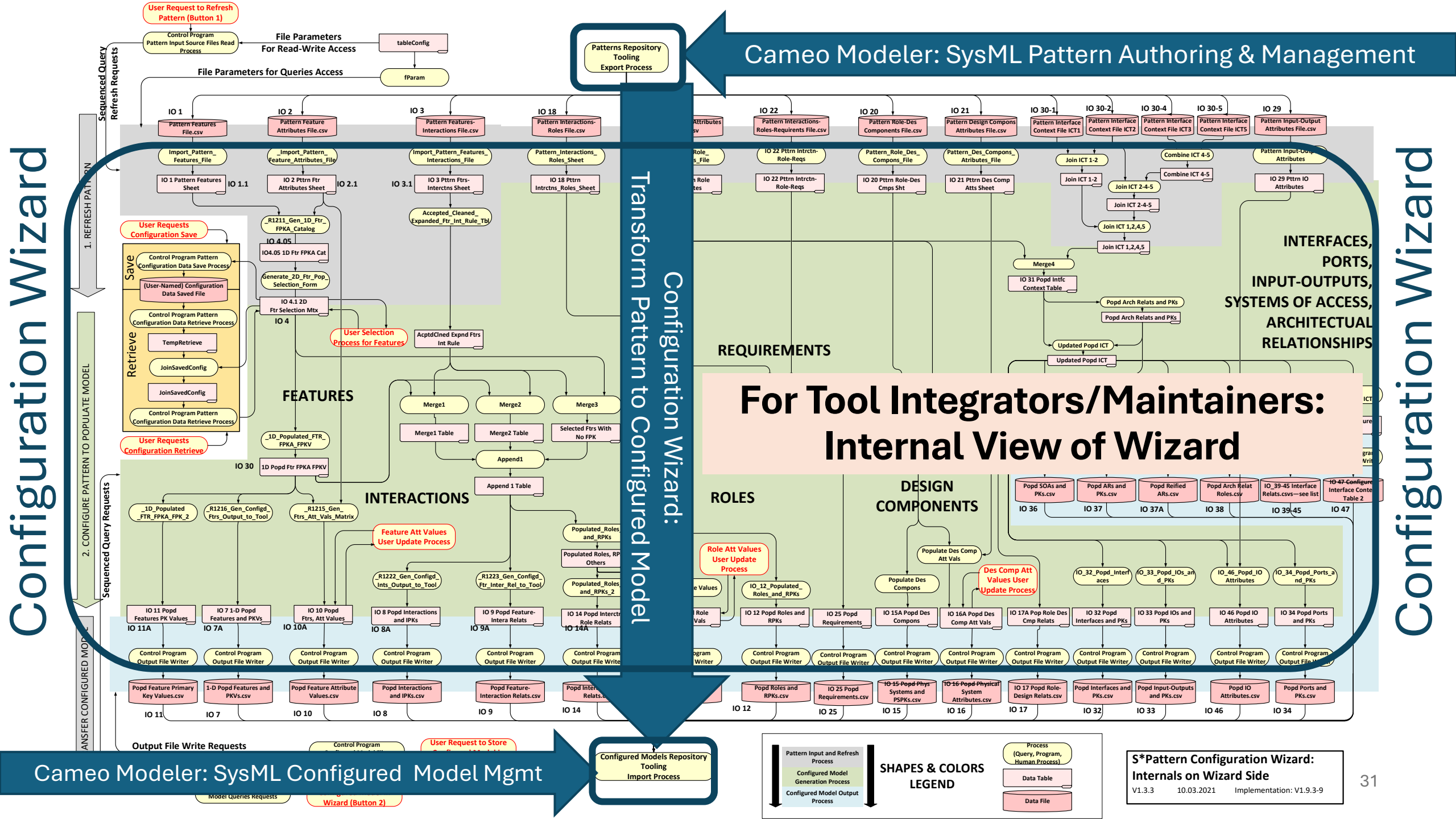
User Selection of Feature Primary Key Values

Fixed Pattern Menu of Features

Variable User Selections of Features and Populated Feature PK Values

For Tool Integrators/Maintainers: Built-In Tool-Agent-User Interface Specification

	A	B	C	D	E	F	G	H	I	J	K	L
1	Use Case	Tool to Agent	Agent to Tool	Agent to User	User to Agent	IO #	IO Name (File Name for Files)	Table Name in Integrated Report	IO Type	Col #	Column Name	Meaning and Notes
2	Configure Pattern	X				IO 0	Export Tables	N/A	XLS File	1		Integrated rept of Pattern Tables from repository
3	Configure Pattern	X				IO 0	Export Tables	N/A	XLS File	2-N		Tbl Col Hdgs, or Tbl Col Data; Contains IO 1,2,3,18
4	Configure Pattern	X				IO 1	Pattern Features File	Pattern Features File	CSV File	1	Name	
5	Configure Pattern	X				IO 1	Pattern Features File	Pattern Features File	CSV File	2	Configuration Rule	
6	Configure Pattern	X				IO 2	Pattern Feature Attributes File	Pattern Feature Attributes File	CSV File	1	Owner	Feature Name
7	Configure Pattern	X				IO 2	Pattern Feature Attributes File	Pattern Feature Attributes File	CSV File	2	Name	Feature Attribute
8	Configure Pattern	X				IO 2	Pattern Feature Attributes File	Pattern Feature Attributes File	CSV File	3	Applied Stereotype	Feature PK Flag ("Feature Attribute" or "Feature P
9	Configure Pattern	X				IO 2	Pattern Feature Attributes File	Pattern Feature Attributes File	CSV File	4	Possible Values	Possible values, using colon (:) delimiters within si
10	Configure Pattern	X				IO 3	Pattern Features-Interactions File	Pattern Features-Interactions File	CSV File	1	Type (Role B)	Feature Name
11	Configure Pattern	X				IO 3	Pattern Features-Interactions File	Pattern Features-Interactions File	CSV File	2	FPK Value	FPK Value
12	Configure Pattern	X				IO 3	Pattern Features-Interactions File	Pattern Features-Interactions File	CSV File	3	Type (Role A)	Interaction Name
13	Configure Pattern	X				IO 3	Pattern Features-Interactions File	Pattern Features-Interactions File	CSV File	4	IPK Rule	Interaction PK Rule
14	Configure Pattern	X				IO 18	Pattern Interactions-Roles File	Pattern Interaction Roles	CSV File	1	Type (Role B)	Interaction Name
15	Configure Pattern	X				IO 18	Pattern Interactions-Roles File	Pattern Interaction Roles	CSV File	2	Type (Role A)	Role Name
16	Configure Pattern	X				IO 18	Pattern Interactions-Roles File	Pattern Interaction Roles	CSV File	3	<<HasRole>> IPK Value	RPK Rule; guillemets portion is optional
17	Configure Pattern	X				IO 18	Pattern Interactions-Roles File	Pattern Interaction Roles	CSV File	4	<<HasRole>> RPK Rule	IPK Value; guillemets portion is optional
18	Configure Pattern	X				IO 19	Pattern Role Attributes File	Pattern Role Attributes File	CSV File	1	Owner	Logical System Name
19	Configure Pattern	X				IO 19	Pattern Role Attributes File	Pattern Role Attributes File	CSV File	2	Name	Attribute Name
20	Configure Pattern	X				IO 20	Pattern Role-Des Compons File	Pattern Functional Role Allocation	CSV File	1	Type (Role B)	Des Compon Name
21	Configure Pattern	X				IO 20	Pattern Role-Des Compons File	Pattern Functional Role Allocation	CSV File	2	Type (Role A)	Role Name
22	Configure Pattern	X				IO 20	Pattern Role-Des Compons File	Pattern Functional Role Allocation	CSV File	3	<<FunctionalRoleAlloca	PSPK Rule; guillemets portion is optional
23	Configure Pattern	X				IO 20	Pattern Role-Des Compons File	Pattern Functional Role Allocation	CSV File	4	<<FunctionalRoleAlloca	RPK Value; guillemets portion is optional
24	Configure Pattern	X				IO 21	Pattern Des Compons Attributes Fi	Pattern Des Compons Attributes File	CSV File	1	Owner	Des Compon Name
25	Configure Pattern	X				IO 21	Pattern Des Compons Attributes Fi	Pattern Des Compons Attributes File	CSV File	2	Name	Des Compon Attribute
26	Configure Pattern	X				IO 22	Pattern Interaction-Role-Reqs File	Pattern Requirements	CSV File	1	Owner	RTF Name
27	Configure Pattern	X				IO 22	Pattern Interaction-Role-Reqs File	Pattern Requirements	CSV File	2	Functional Interaction	
28	Configure Pattern	X				IO 22	Pattern Interaction-Role-Reqs File	Pattern Requirements	CSV File	3	IPK Value	


Cameo Modeler: SysML Pattern Authoring & Management
Configuration Wizard
Configuration Wizard
Configuration Wizard: Transform Pattern to Configured Model
For Tool Integrators/Maintainers: Internal View of Wizard
Cameo Modeler: SysML Configured Model Mgmt
Patterns Repository Tooling Export Process
Configured Models Repository Tooling Import Process
1. REFRESH PATTERN
2. CONFIGURE PATTERN TO POPULATE MODEL
3. TRANSFER CONFIGURED MODEL
Sequenced Query Refresh Requests
Sequenced Query Requests
User Request to Refresh Pattern (Button 1)
Control Program Pattern Input Source Files Read Process
File Parameters For Read-Write Access
tableConfig
File Parameters for Queries Access
fParam
IO 1
Pattern Features File.csv
IO 2
Pattern Feature Attributes File.csv
IO 3
Pattern Features-Interactions File.csv
IO 18
Pattern Interactions-Roles File.csv
IO 22
Pattern Interactions-Roles-Requirents File.csv
IO 20
Pattern Role-Des Components File.csv
IO 21
Pattern Design Compons Attributes File.csv
IO 30-1
Pattern Interface Context File ICT1
IO 30-2
Pattern Interface Context File ICT2
IO 30-4
Pattern Interface Context File ICT3
IO 30-5
Pattern Interface Context File ICT5
IO 29
Pattern Input-Output Attributes File.csv
IO 1.1
IO 2.1
IO 3.1
IO 4.05
IO 4
IO 30
IO 11A
IO 7A
IO 10A
IO 8A
IO 9A
IO 14A
IO 12
IO 25
IO 15
IO 16
IO 17
IO 32
IO 33
IO 46
IO 34
IO 36
IO 37
IO 37A
IO 38
IO 39-45
IO 47
REQUIREMENTS
FEATURES
INTERACTIONS
ROLES
DESIGN COMPONENTS
INTERFACES, PORTS, INPUT-OUTPUTS, SYSTEMS OF ACCESS, ARCHITECTUAL RELATIONSHIPS
User Requests Configuration Save
Save
Control Program Pattern Configuration Data Save Process
(User-Named) Configuration Data Saved File
Retrieve
Control Program Pattern Configuration Data Retrieve Process
TempRetrieve
JoinSavedConfig
JoinSavedConfig
Control Program Pattern Configuration Data Retrieve Process
User Requests Configuration Retrieve
User Selection Process for Features
Feature Att Values User Update Process
Role Att Values User Update Process
Des Comp Att Values User Update Process
Merge1
Merge2
Merge3
Merge1 Table
Merge2 Table
Selected Ftrs With No FPK
Append1
Append 1 Table
Merge4
Updated Popd ICT
Output File Write Requests
Control Program Output File Writer
Popd Feature Primary Key Values.csv
1-D Popd Features and PKVs.csv
Popd Feature Attribute Values.csv
Popd Interactions and IPKs.csv
Popd Feature-Interaction Relats.csv
Popd Roles and RPKs.csv
IO 25 Popd Requirements.csv
IO 15 Popd Phys Systems and PSPKs.csv
IO 16 Popd Physical System Attributes.csv
IO 17 Popd Role-Design Relats.csv
Popd Interfaces and PKs.csv
Popd Input-Outputs and PKs.csv
Popd IO Attributes.csv
Popd Ports and PKs.csv
Popd SOAs and PKs.csv
Popd ARs and PKs.csv
Popd Reified ARs.csv
Popd Arch Relat Roles.csv
IO_39-45 Interface Relats.csvs—see list
Pattern Input and Refresh Process
Configured Model Generation Process
Configured Model Output Process
SHAPES & COLORS LEGEND
Process (Query, Program, Human Process)
Data Table
Data File
S*Pattern Configuration Wizard: Internals on Wizard Side
V1.3.3 10.03.2021 Implementation: V1.9.3-9

Shared patterns and confidential configurations

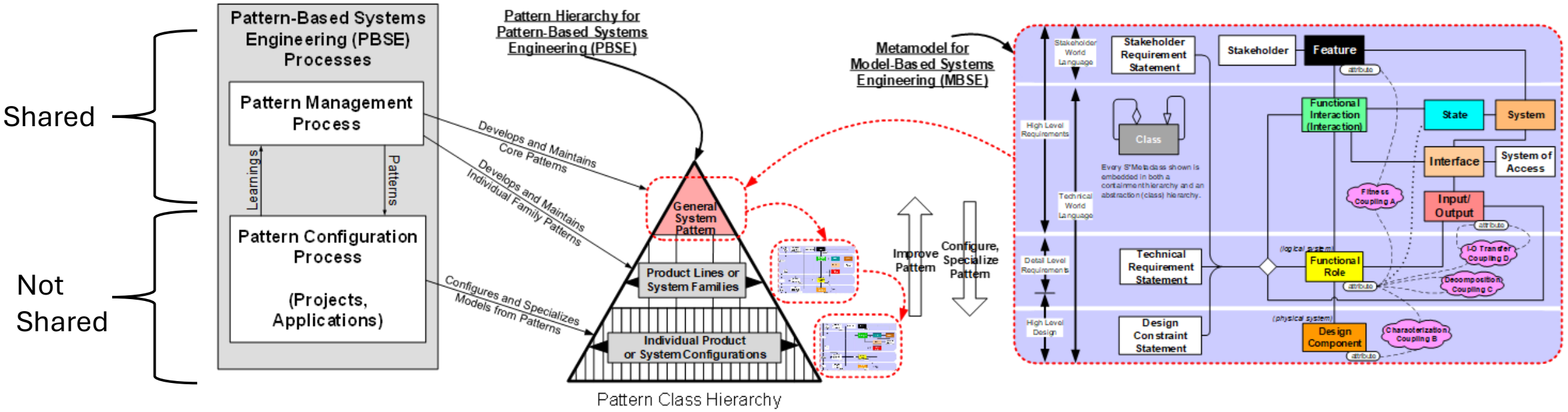
1. Pattern sharing across projects and enterprises encourages:
   - Economy and speed in acquisition
   - Compatibility of models
   - Interoperability across the modeled systems
2. But sharing of models has historically been viewed as contrary to protection of proprietary or classified information.

S*Patterns offer a new option for accomplishing (1) while protecting (2):

Put the sensitive information in the configuration part, not the pattern.

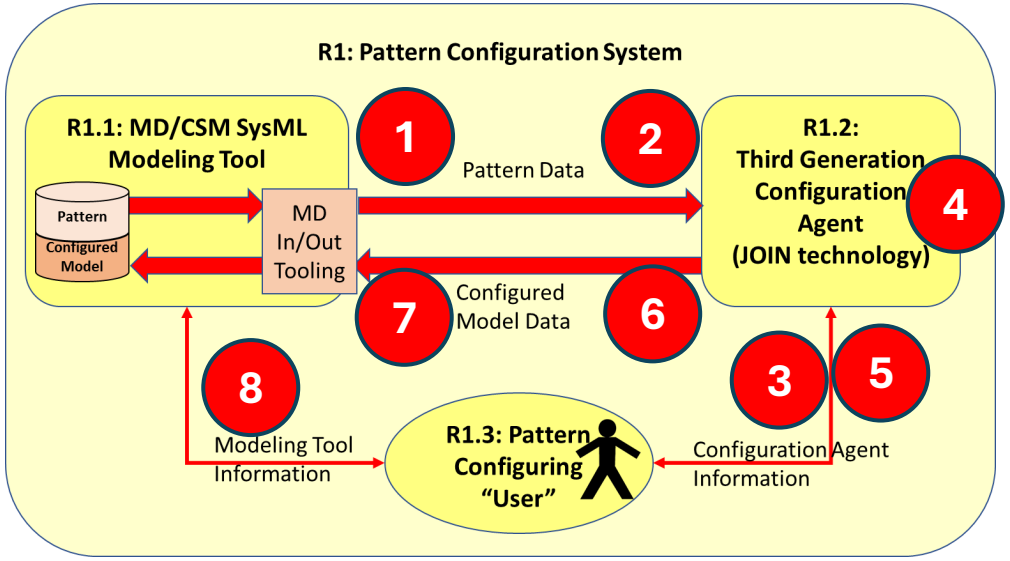
Shared patterns and confidential configurations

- Remember that the configurable parts of the pattern include:
  - The configured combinations of populated Features, Interactions, Roles, Design Components, Interfaces, States, Failure Modes, etc.
  - The values of attributes of Features, Roles, and Design Components.



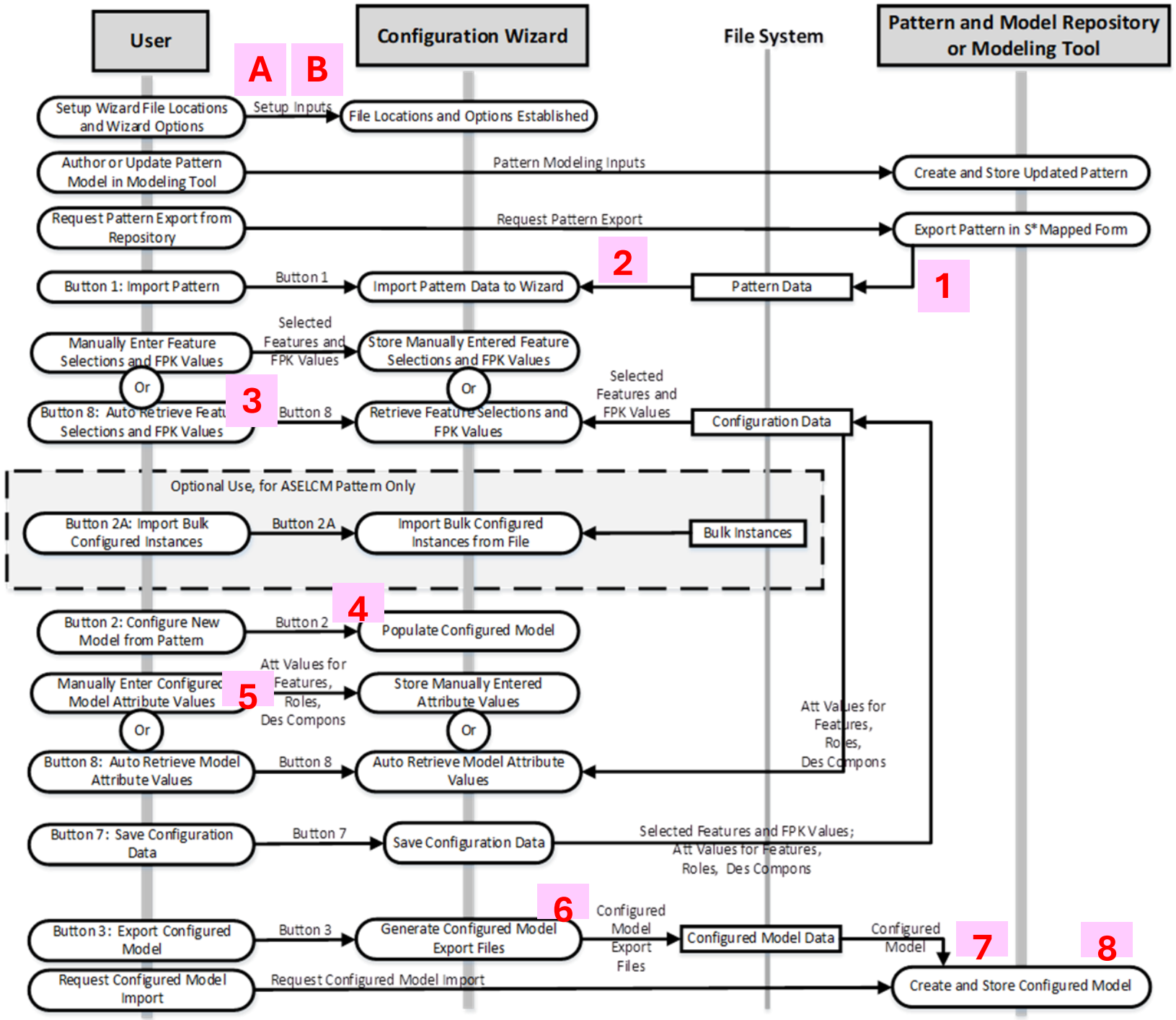
BREAK

Configure an S*Pattern Using the Configuration Wizard



A short demonstration of the configuration cycle:

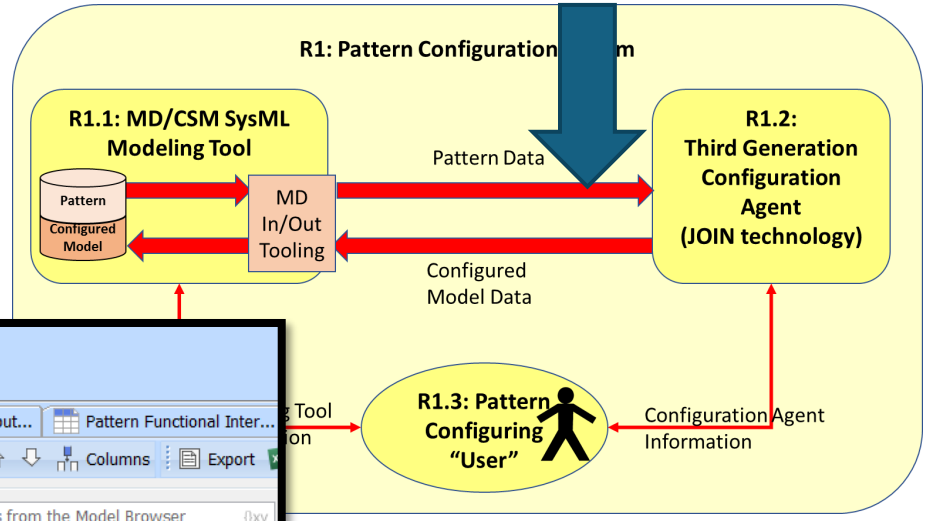
1. Exporting SysML S*Pattern from Cameo Systems Modeler.
2. Configuration Agent imports S*Pattern.
3. User selects Stakeholder Features as basis for a configured model.
4. Configuration Agent generates Configured Model from Pattern.
5. User manages configured values of Attributes of Features, Roles, Design Compons
6. Configured Model exported from Agent.
7. Configured Model imported into Cameo Systems Modeler.
8. User accesses SysML Configured Model in Cameo Systems Modeler.

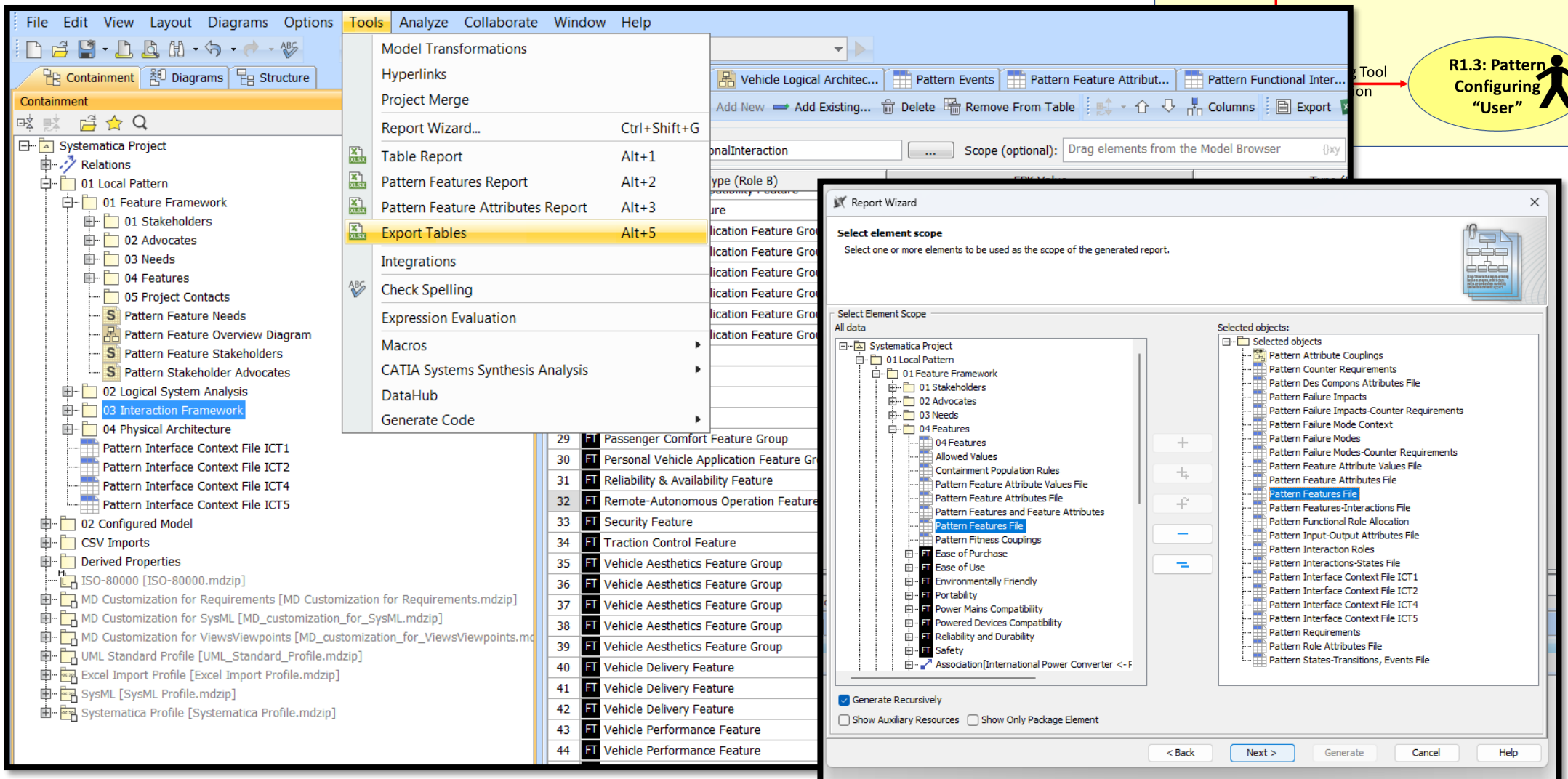


A. User selects location for Export Tables Pattern File.
B. User selects location for Configured Model Import Files.

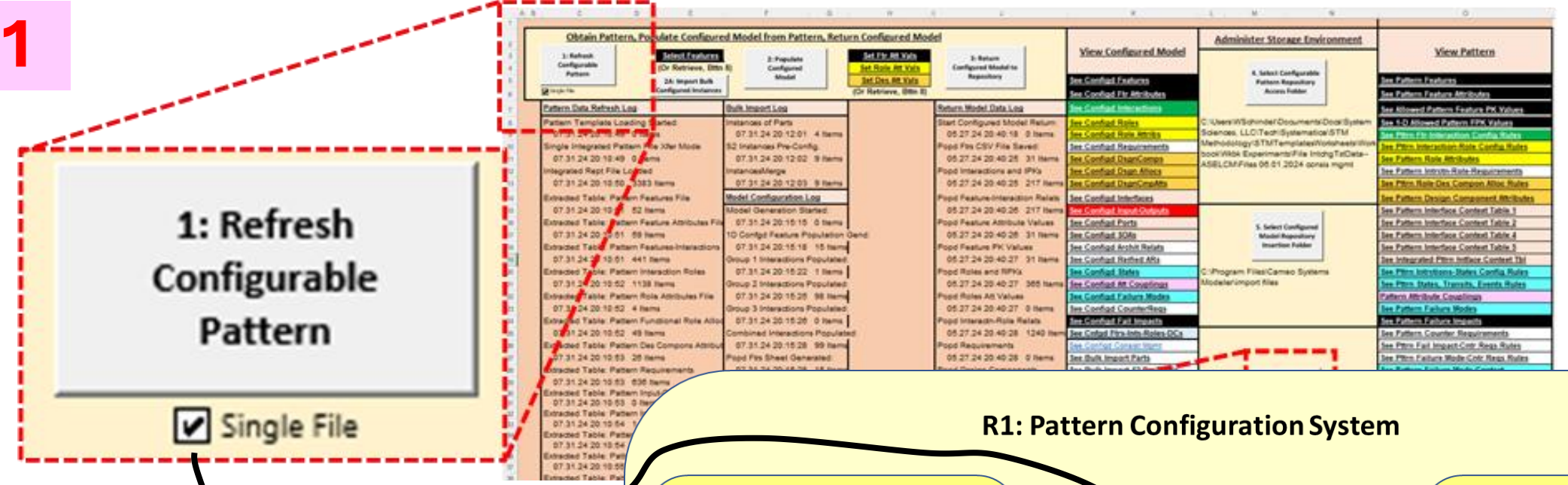
1. Exporting MBSE S*Pattern from MBSE Modeling Tool.
2. Configuration Wizard imports S*Pattern.
3. User selects Stakeholder Features as basis for a configured model.
4. Configuration Wizard generates Configured Model from Pattern.
5. User manages configured values of Attributes for Features, Roles, Design Components.
6. Configured Model exported from Wizard.
7. Configured Model imported into MBSE Modeling Tool.
8. User accesses Configured Model in MBSE Modeling Tool.

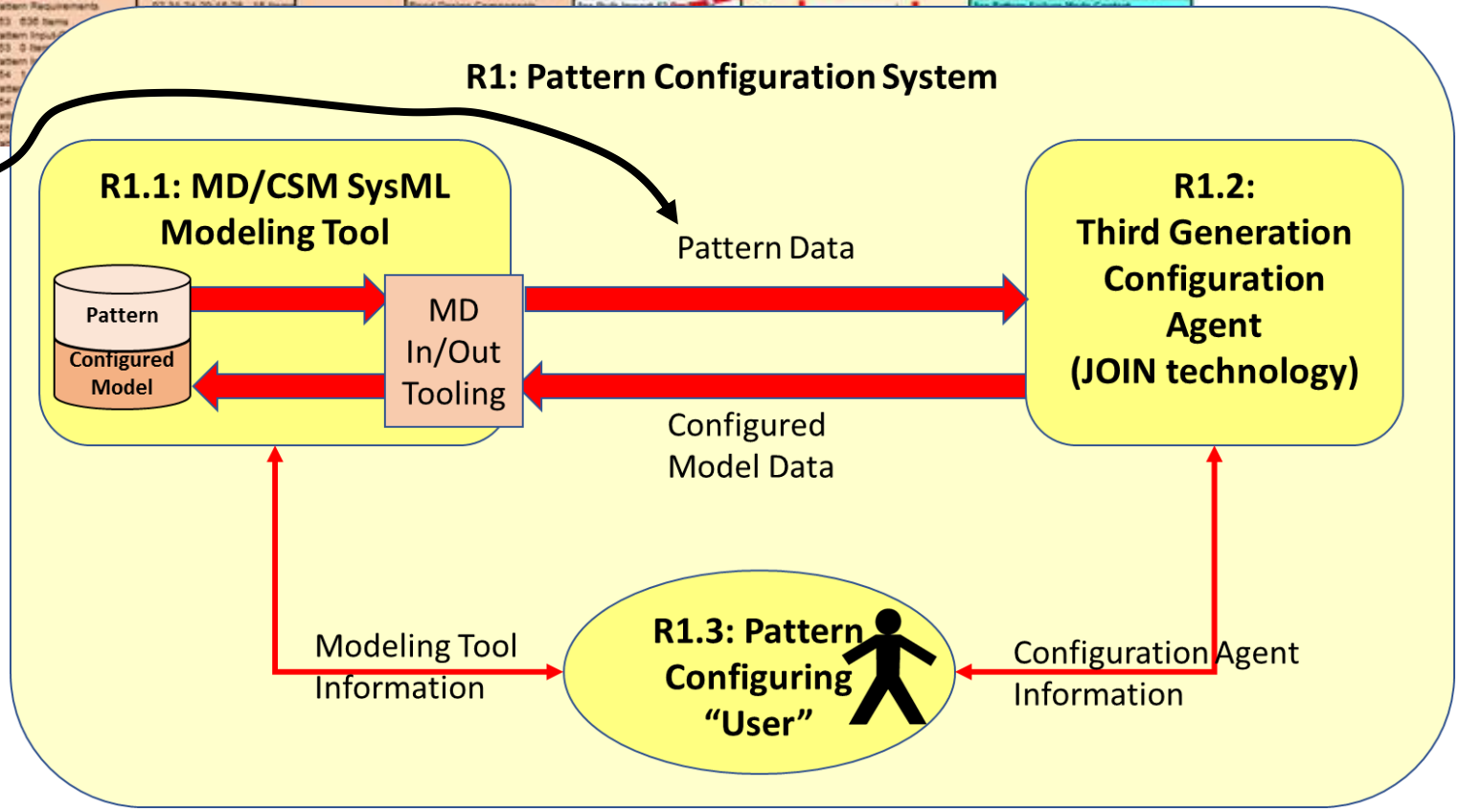
1 Export S* Pattern Tables using the Report Wizard





1

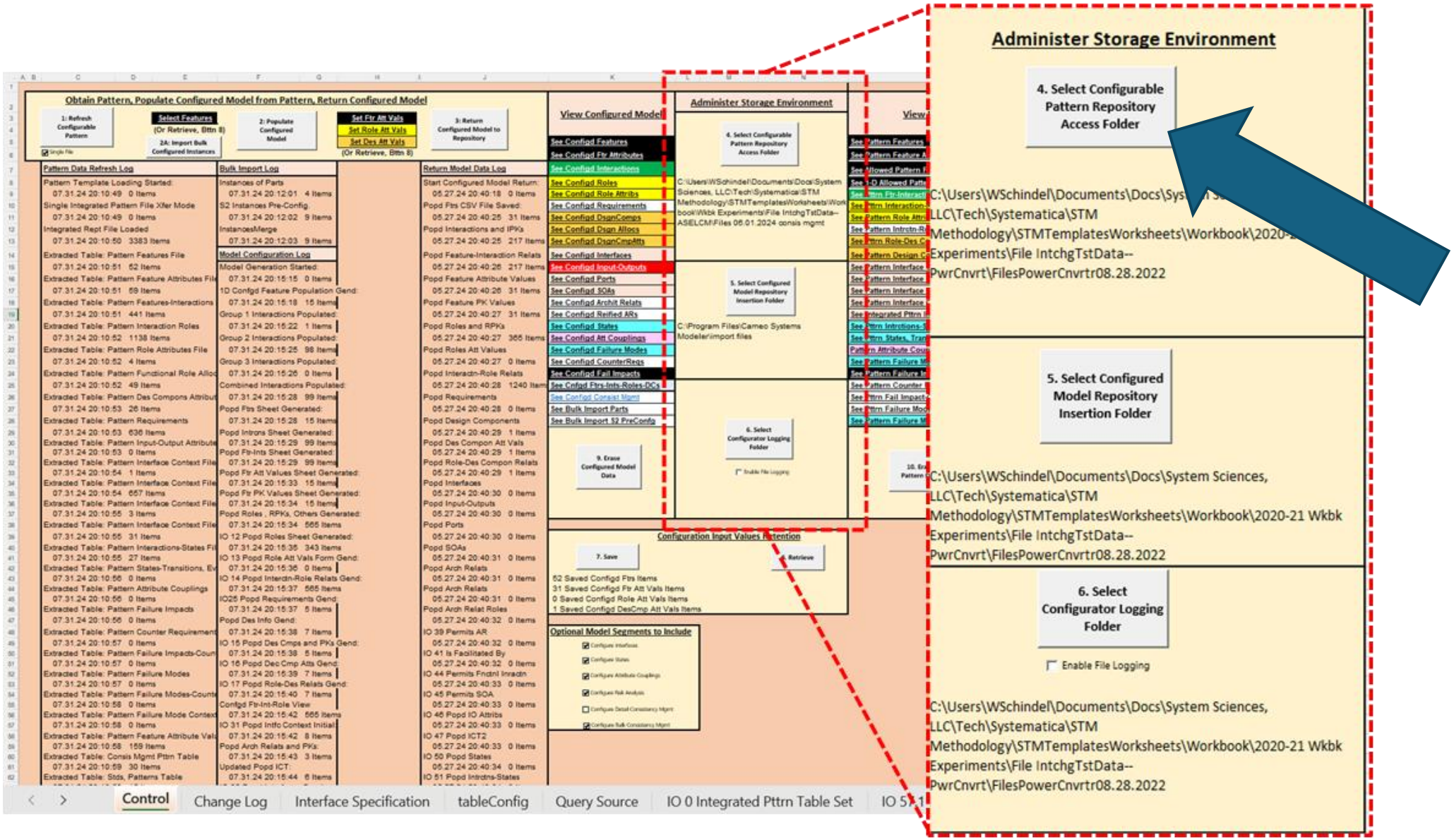




Option: Pattern data can be "refreshed" (transferred to Configuration Wizard) in a single file report of multiple tables, or in multiple files of one table per file.

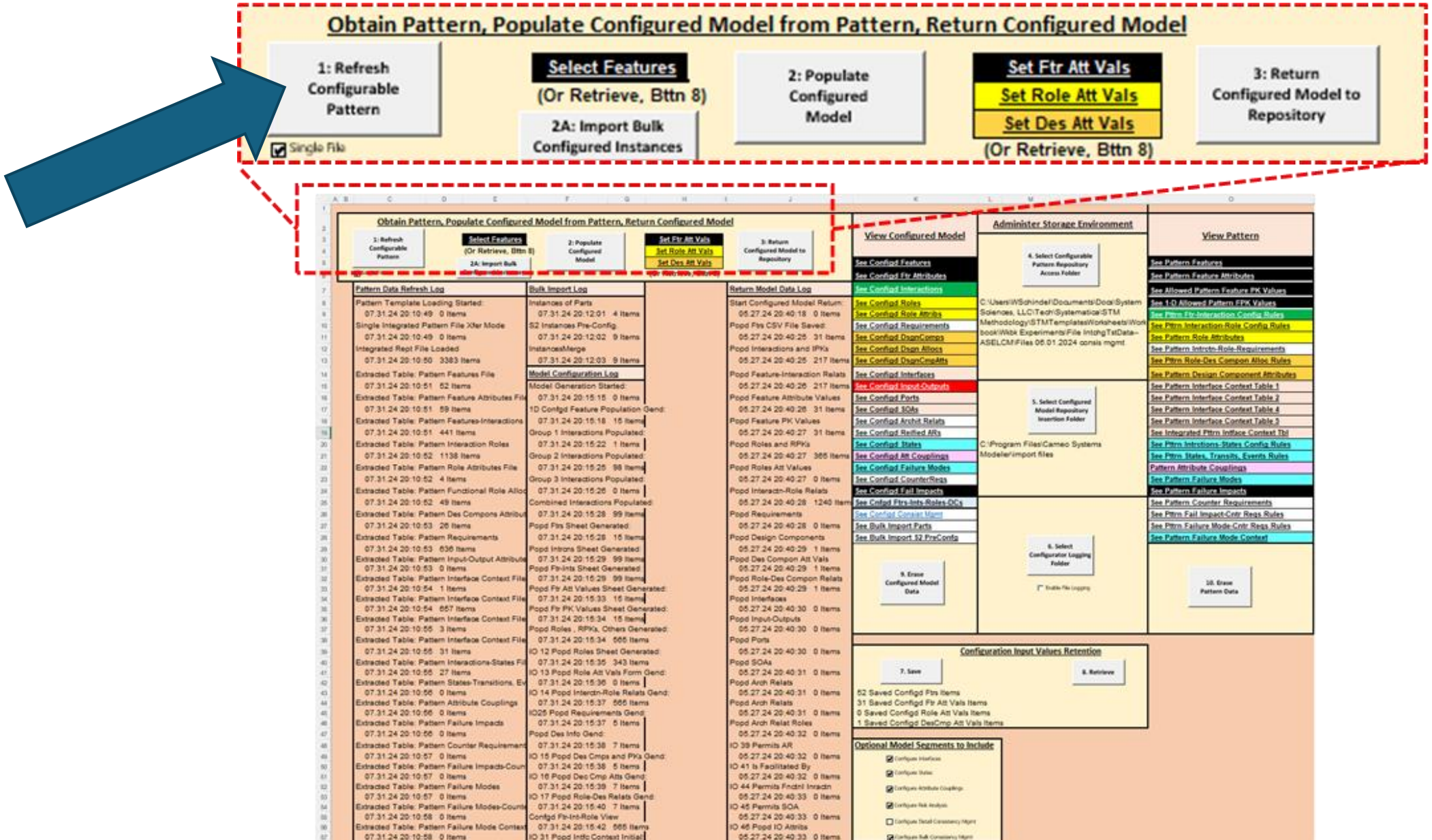
Pattern File Repository Location Selection

A

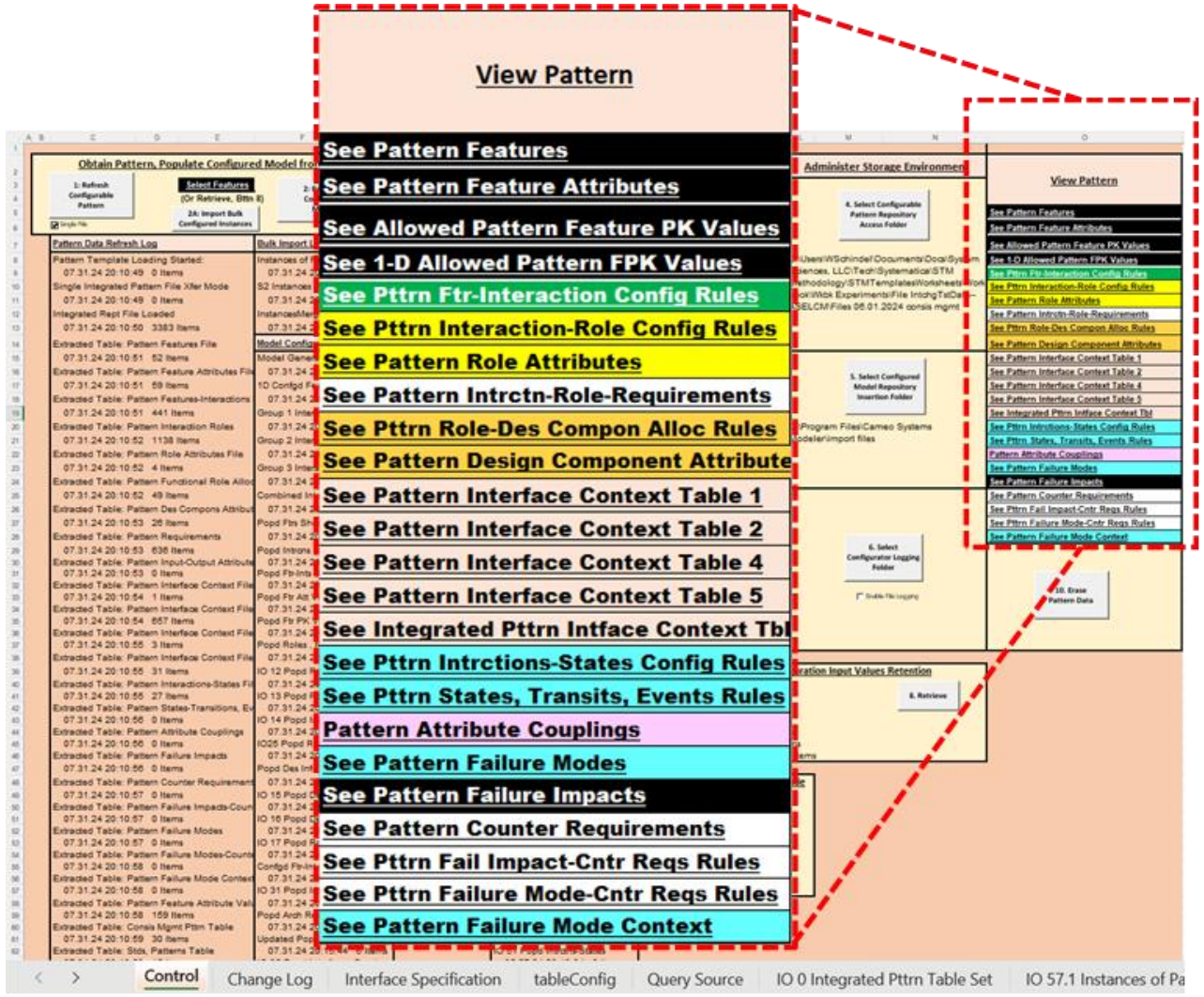


2

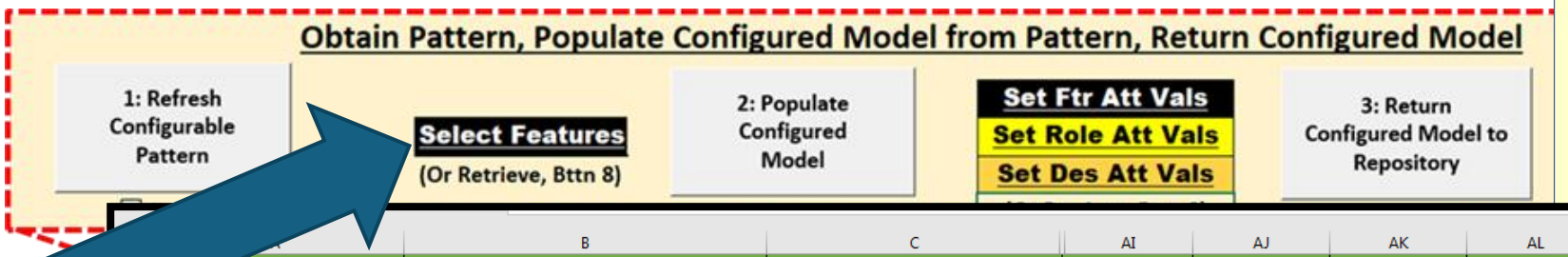
Pattern File Repository Access

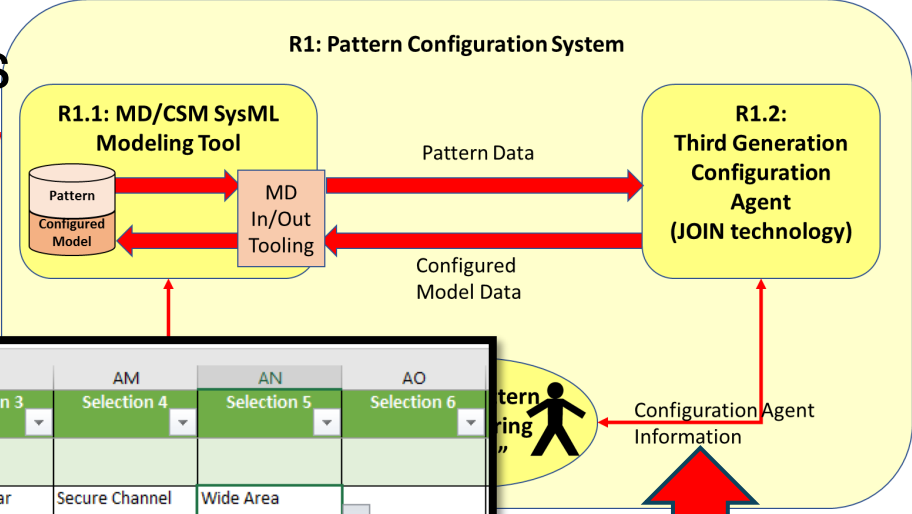


Links (buttons) for user viewing of resulting Pattern Data in the Wizard, after importing from the Export Tables:



3 Select Features & Feature Primary Key Values



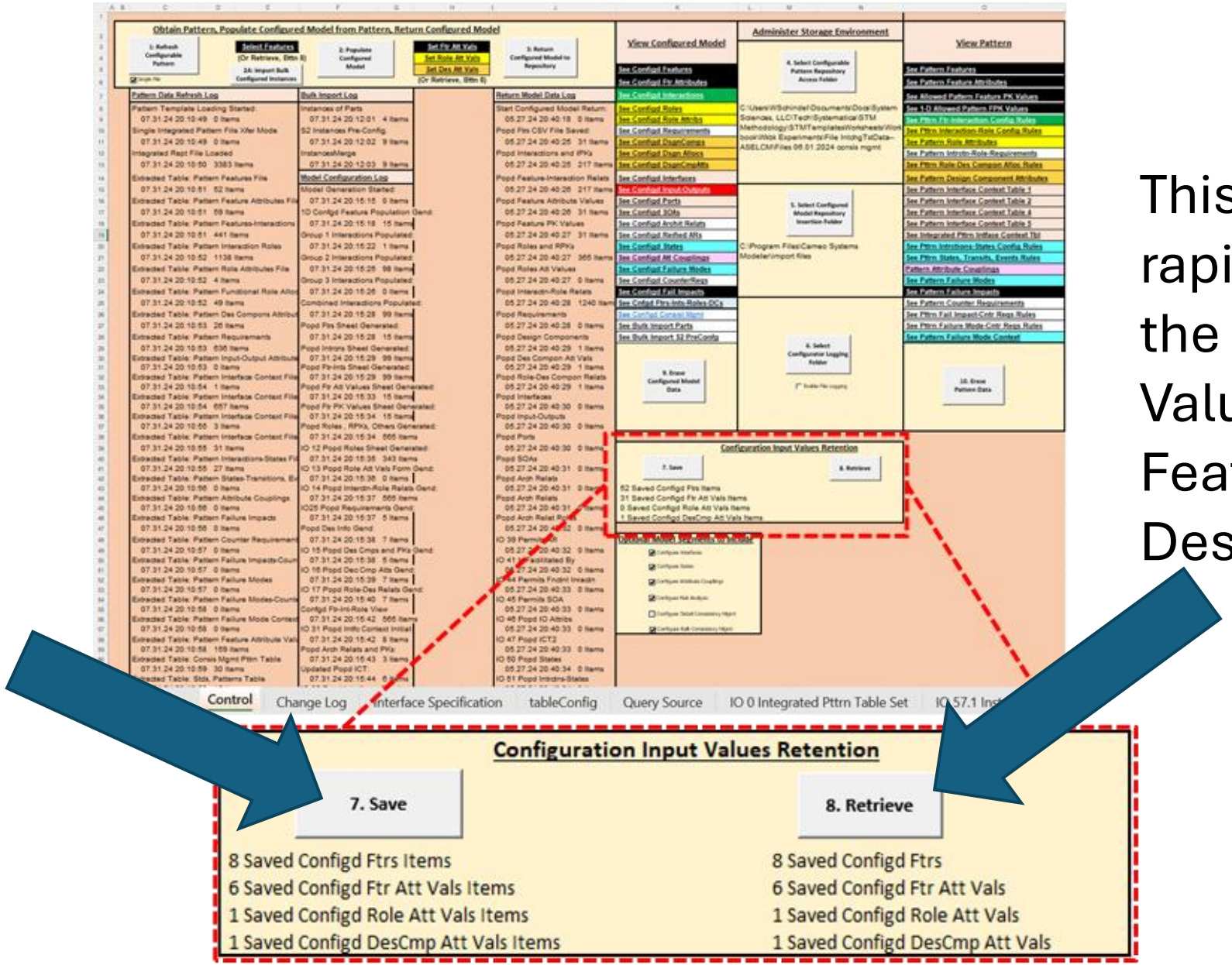


	B	C	AI	AJ	AK	AL	AM	AN	AO
	Configuration Rule	Feature Name	Feature Attribute	Populate? Yes/No	Selection 1	Selection 2	Selection 3	Selection 4	Selection 5	Selection 6
4	Optional	Commercial Vehicle Application Feature Group	Commercial Application Type	No	Hazardous Liquid Transport					
5	Optional	Communications Feature Group	Communication Capability	Yes	IFF	Local Bluetooth Connectivity	Local Cellular	Secure Channel	Wide Area Internet	
6	Optional	Configurability Feature	Configuration Management Capability	No						
7	Optional	Consumables Compatibility Feature	Consumable Type	No						
8	Mandatory	Cost of Operation Feature		No						
9	Mandatory	Cruise Control Feature		No						
10	Optional	Environmental Compatibility Feature	Environmental Issue	No						
11	Mandatory	Maintainability Feature	Maintenance Capability	No						
12	Optional	Military Vehicle Application Feature Group	Military Application Type	No						
13	Optional	Navigation Feature	Navigation Capability	No						
14	Mandatory	Operability Feature	Operations Capability	No						
15	Optional	Passenger Comfort Feature Group	Comfort Issue	No						
16	Optional	Personal Vehicle Application Feature Group	Personal Application Type	No						
17	Mandatory	Reliability & Availability Feature		No						
18	Optional	Remote Management Access Feature	Remote Access Capability	No						
19	Optional	Remote-Autonomous Operation Feature		No						
20	Mandatory	Safety Feature Group		No						
21	Optional	Security Feature	Security Management Capability	No						
22	Optional	Traction Control Feature		No						
23	Optional	Vehicle Aesthetics Feature Group	Aesthetics Issue	Yes	Exterior Body Style	Exterior Color Galeon Blue	Exterior Color Handon Green	Interior Color Rich Brown	Interior Color Sand Dune	Overall Passenger Experience
24	Mandatory	Vehicle Delivery Feature		No						
25		Vehicle Management Feature		No						
26	Mandatory	Vehicle Performance Feature		Yes						
27										
28										
29										
30										

IFF
Local Bluetooth Connectivi
Local Cellular
Secure Channel
Wide Area Internet

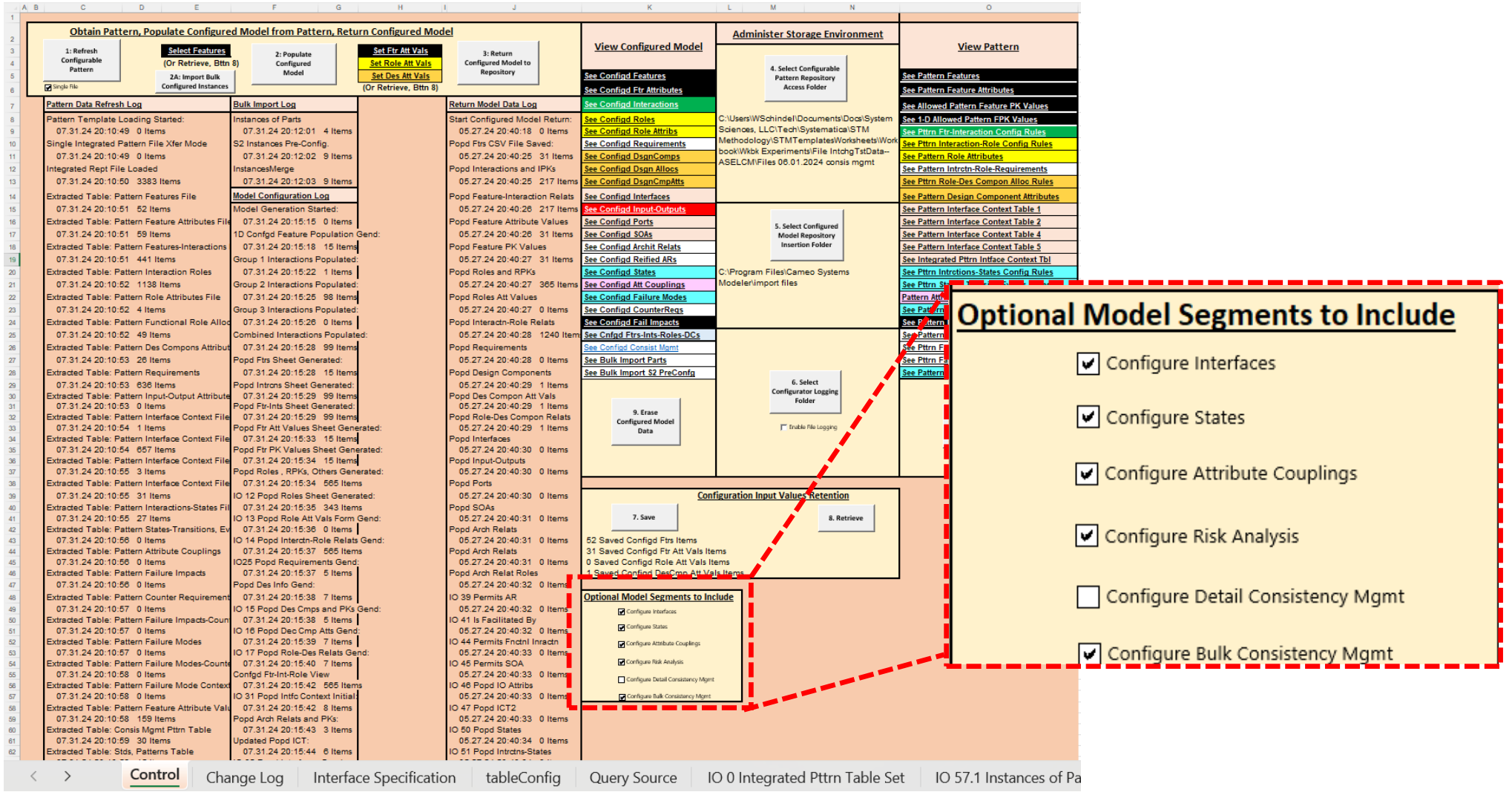
... | JoinICT1,2,4,5 | IO 4.05 1D Ftr FPKA Cat | AcptdCln ed Expnd Ftrs Ints Rule | Merge 1 Table | Merge 2 Table | Selected Ftrs with No FPK | Append 1 Table | IO 4.1 2D

3 Save and Retrieve Feature Configurations



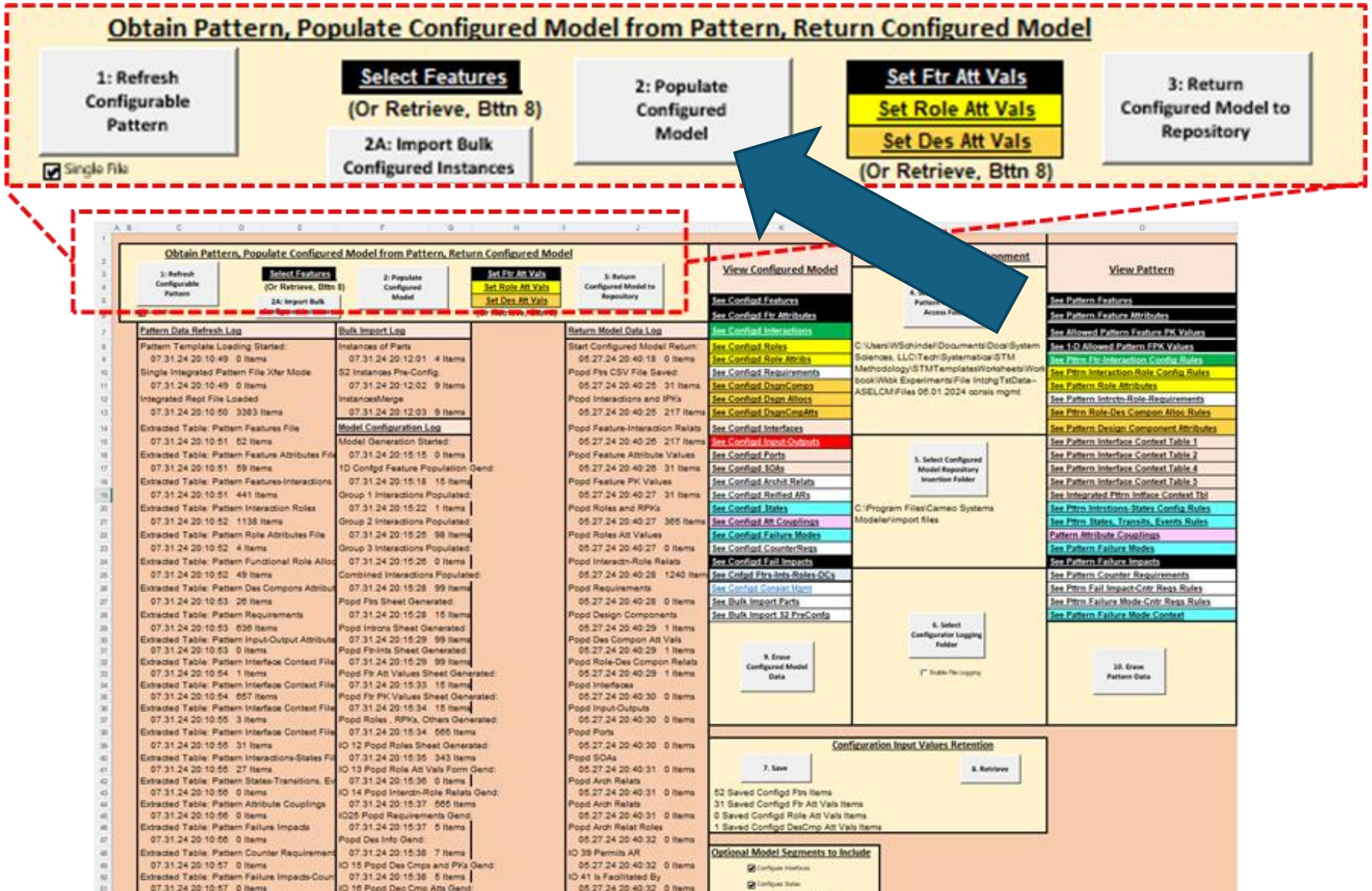
This also can be used to rapidly Save or Retrieve the entered Attribute Values for configured Features, Roles and Design Components.

Selecting Optional Segments of the Model to Configure

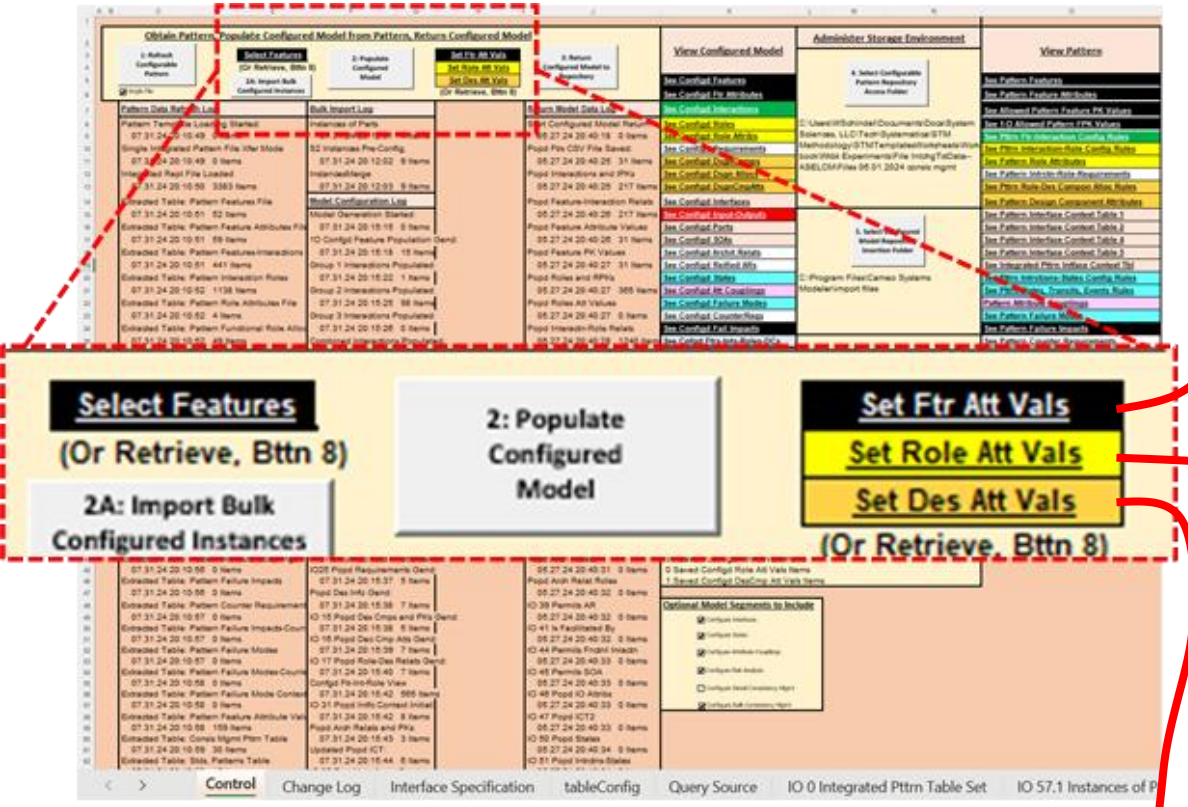


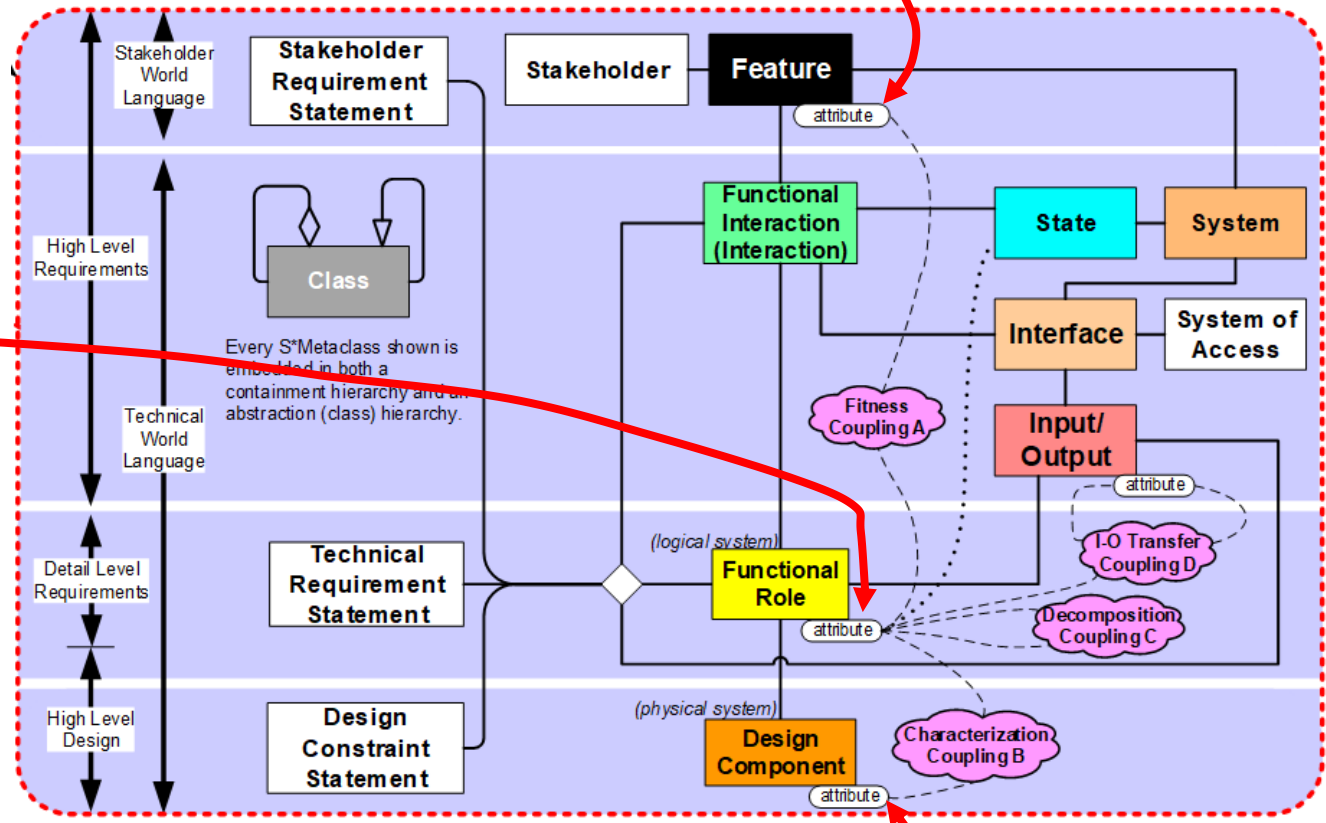
4

Populate the Configured Model



5 Links for user insertion (setting) of Attribute Values for configured Features, Roles, and Design Components:





5 User insertion (setting) of Configured Attribute Values for configured Features, Roles, and Design Components:

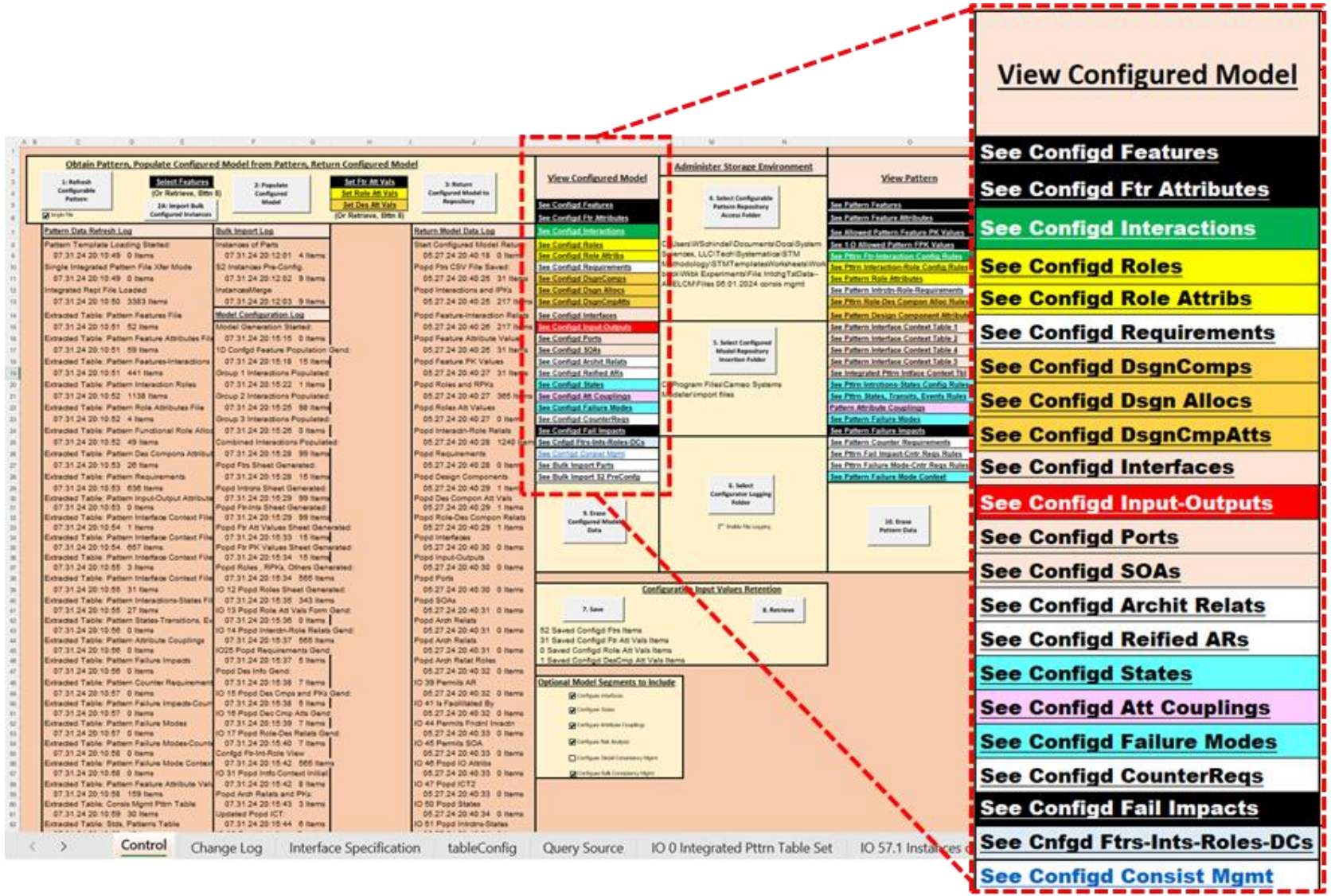
	A	B	C
1	Configured Feature Name	Configured Feature Attribute	Configured Attribute Value
2	Power Mains Compatibility	Max Drain on Mains	2A
3	Power Mains Compatibility	Power Mains Type	110 VAC
4	Powered Devices Compatibility [Power Output	Max Powered Device Load	0.5A
5	Powered Devices Compatibility [Power Output	Max Powered Device Load	0.75A
6	Powered Devices Compatibility [Power Output	Max Powered Device Load	0.75A
7	Reliability and Durability	Design Life	1 Year
8			

	A	B	C
1	Configured Role Name	Configured Role Attribute Name	Configured Role Attribute Value
2	International Power Converter	Max Power Drain	2A
3			
4			
5			
6			

	A	B	C
1	Configured Physical System Name	Configured Physical System Attribute	Configured Physical System Attribute Value
2	Power Converter Assembly	Part Number	ASSY 560
3			
4			
5			

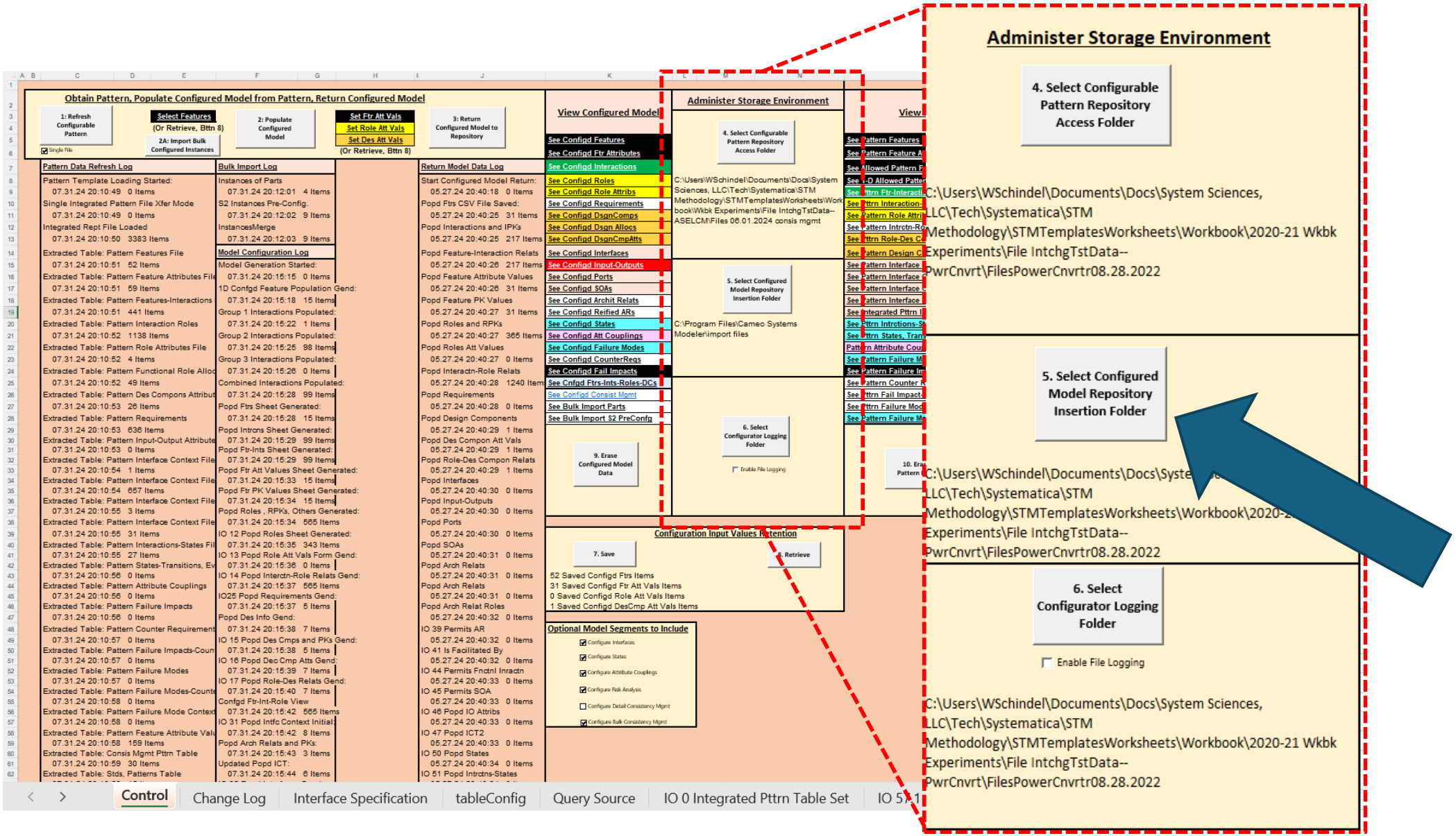
Only these columns are user inserted.

6 Links (buttons) for user viewing of resulting Configured Model in the Wizard, prior to transfer back to Model Repository:



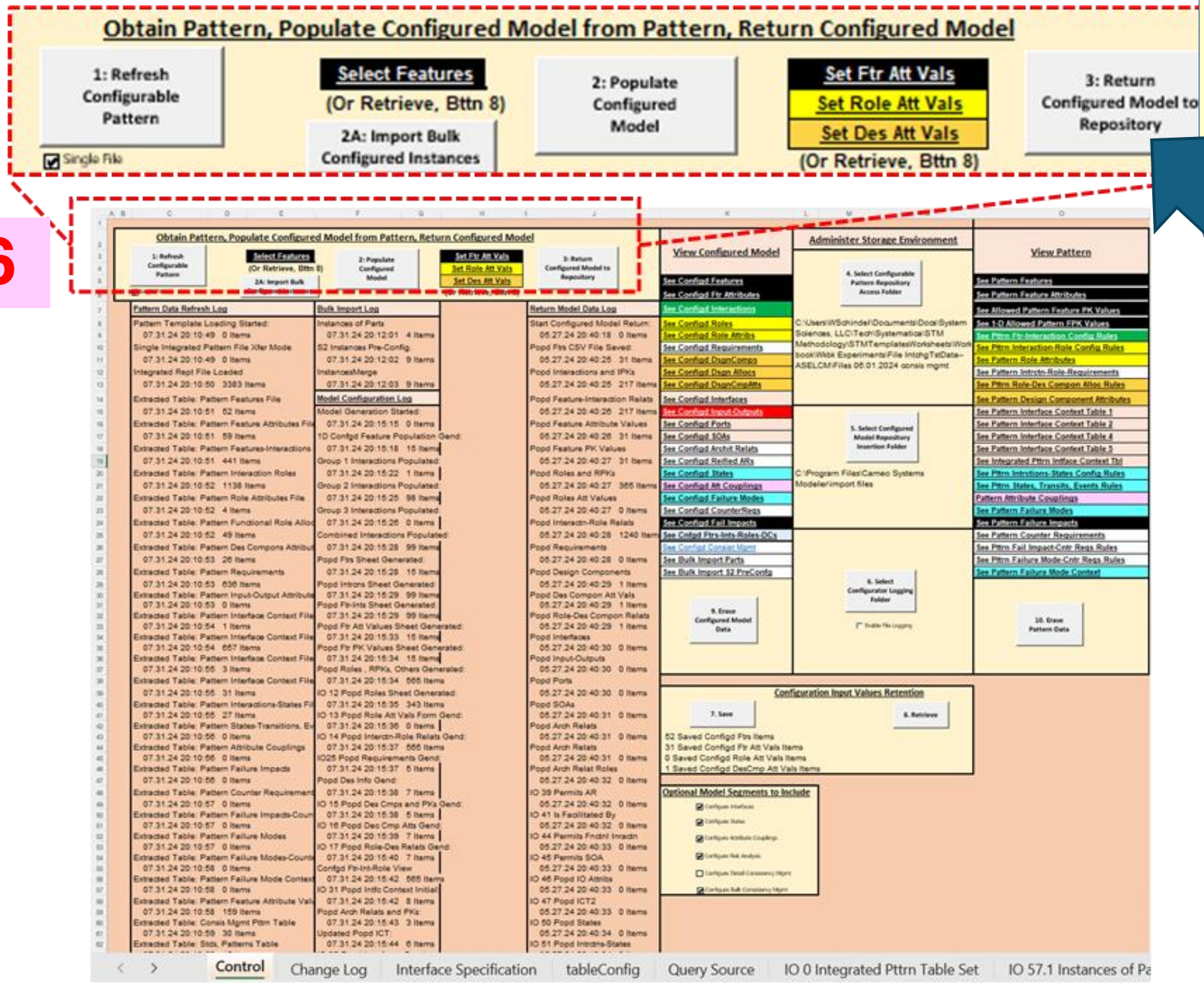
Configured Model File Output Location Selection

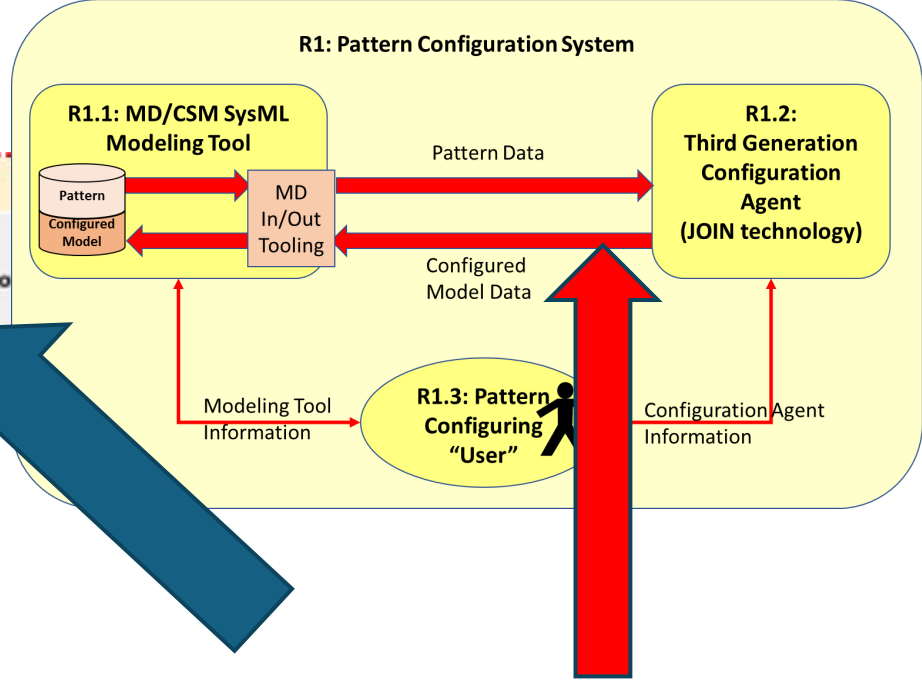
B



Configured Model Output to Files

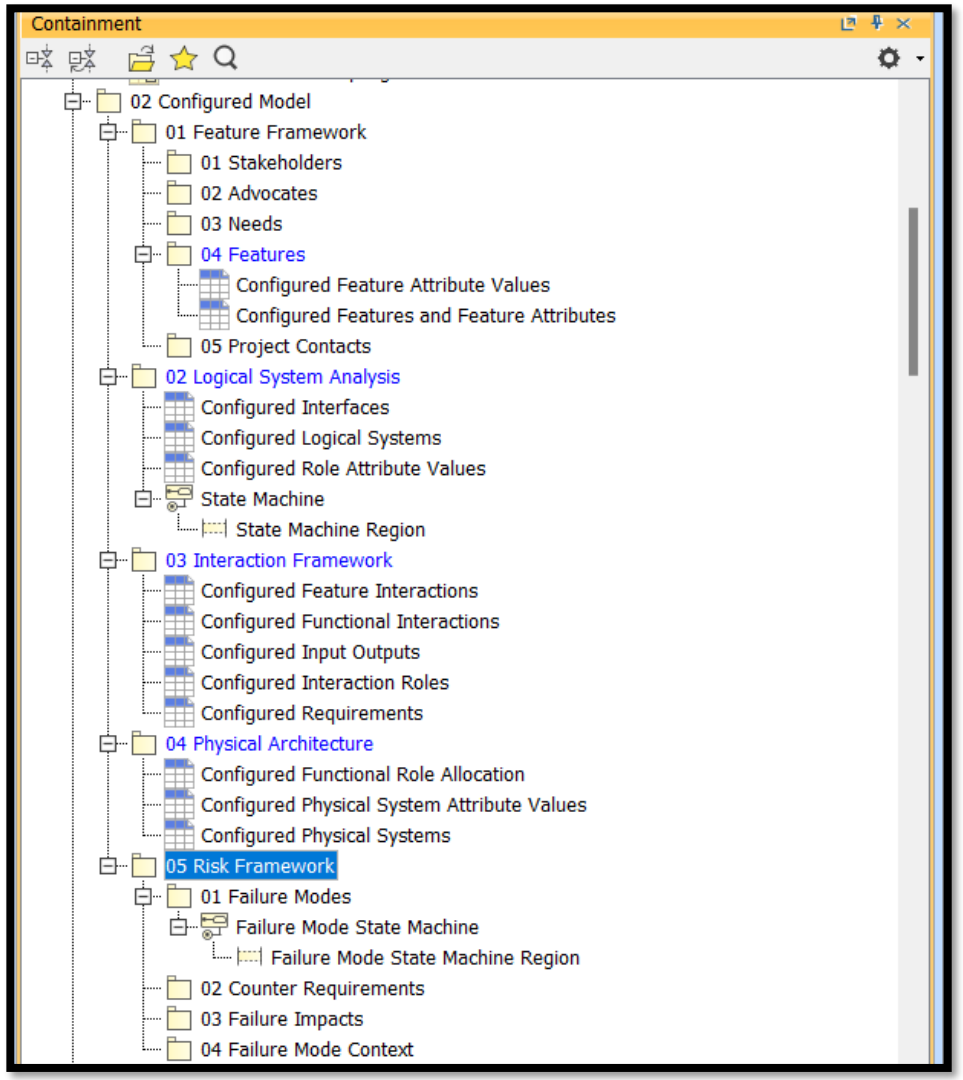
6





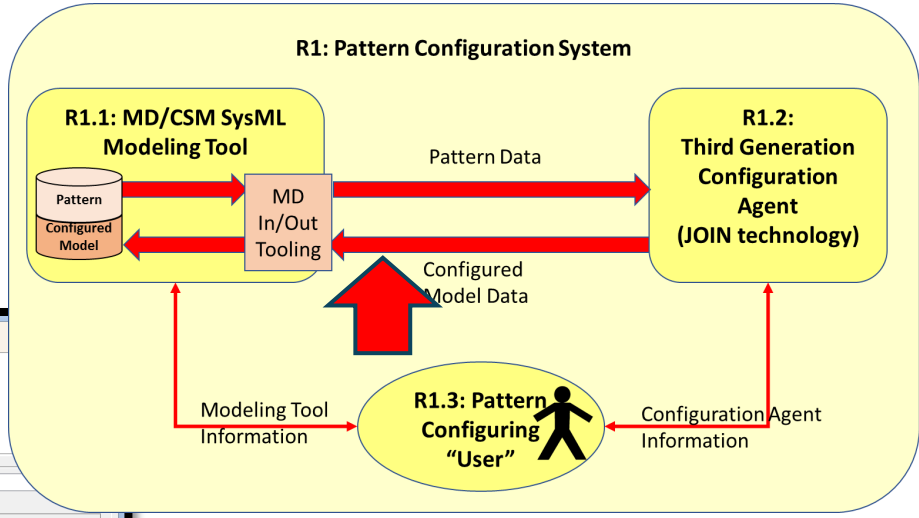
Emptying the Configured Model Packages Before the Import

- Prior to importing a configured model into the repository, any previous model content should be removed from there, leaving the basic framework elements in place.
- Once it is emptied of previous model data, the empty 02 Configured Model package of the Containment Browser should look like the example shown here.
- If it contains elements or relations other than what is shown here, those should be deleted, retaining the package structure and tables shown.

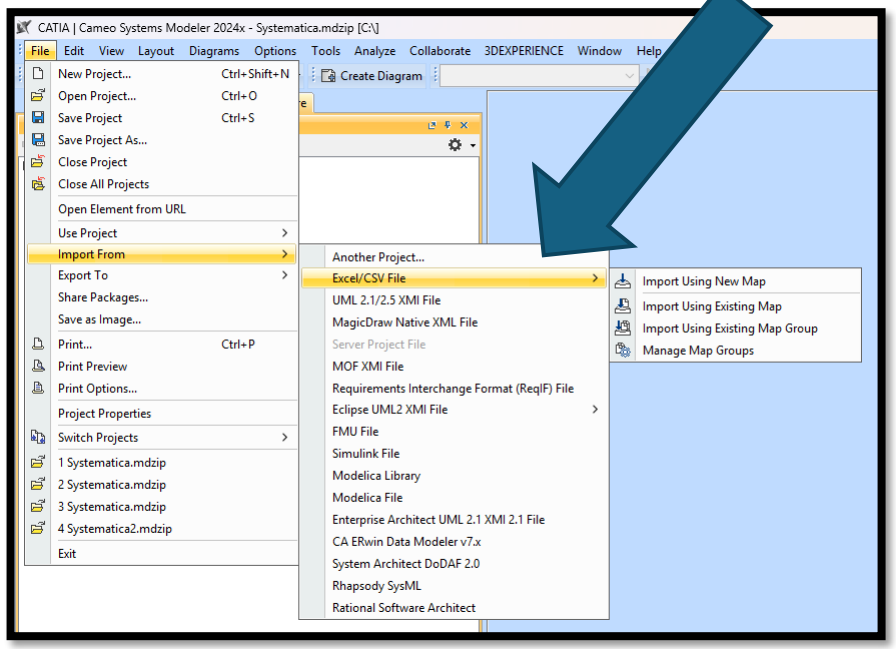


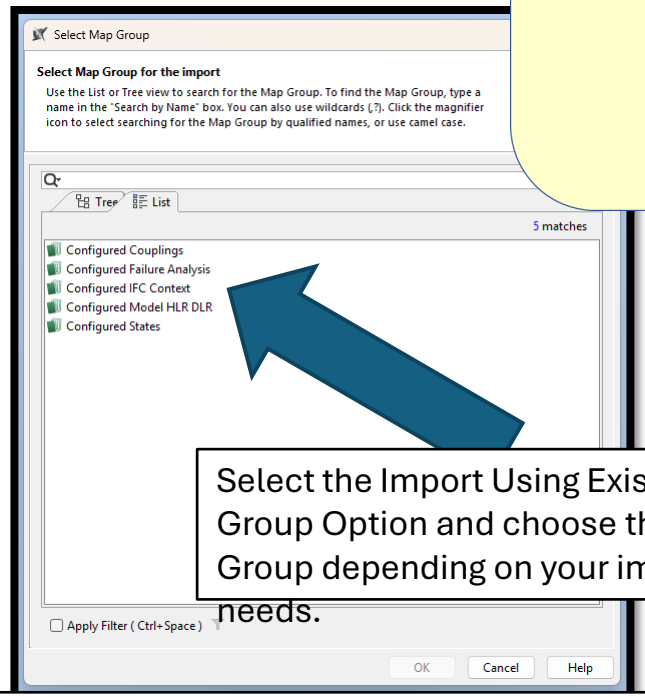
7

Importing the Configured Model Data



Choose the Import From > Excel/CSV File option.





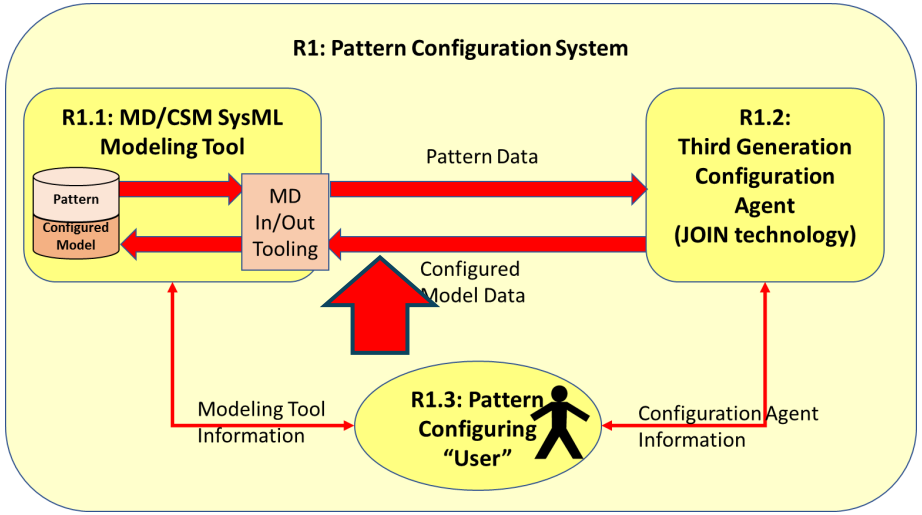
Select the Import Using Existing Map Group Option and choose the Map Group depending on your import needs.

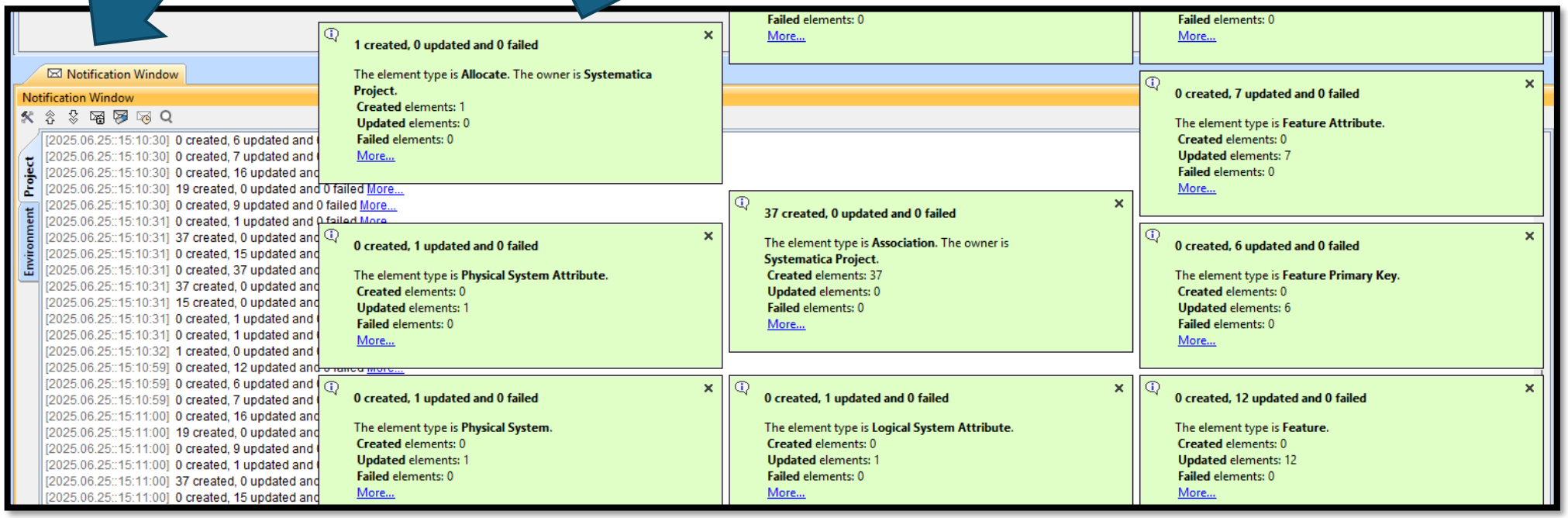
-The Configured Model HLR DLR is the first one to select and imports a subset of the entire model including Features, Interactions, Roles, Requirements, and Design Components.
-Configured States includes importing States, Forks/Joins, Events, and Transitions.
-Configured IFC Context includes importing Interfaces, Input-Outputs, Architectural Relationships, Systems of Access, and Ports
-Configured Couplings includes Couplings, Constraint Parameters, and Connectors
-Configured Failure Analysis includes Counter Requirements, Failure Modes, and Failure Impacts
The Configuration Wizard control panel allows for similar configuration grouping choices. (See Wizard Guide)

7

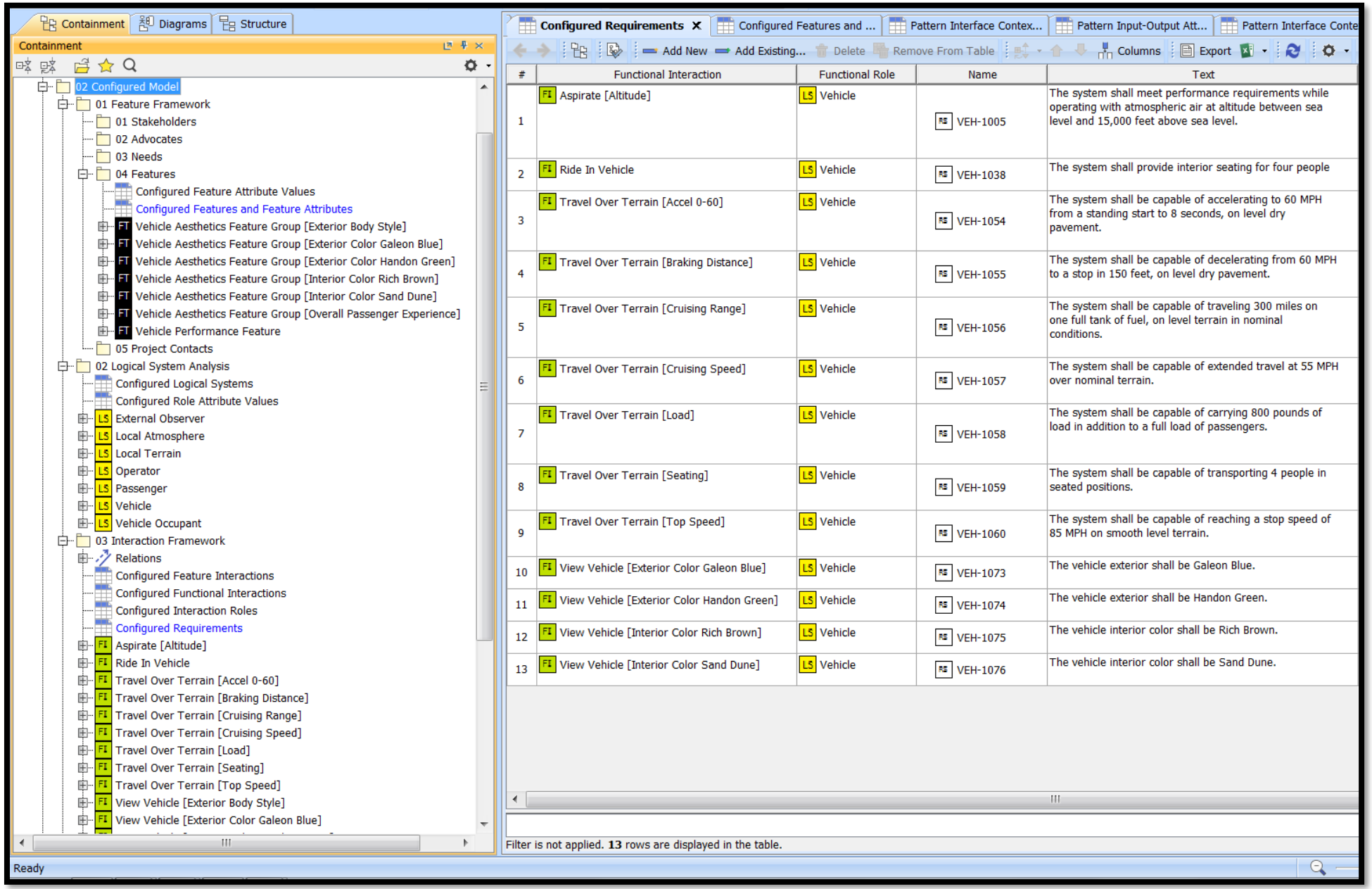
Importing the Configured Model Data

See the Notification Window and/or the individual import log pop-ups for the CSV Import log details. Green shading indicates a successful import, yellow indicates partial success, and red indicates all of the elements failed to import.

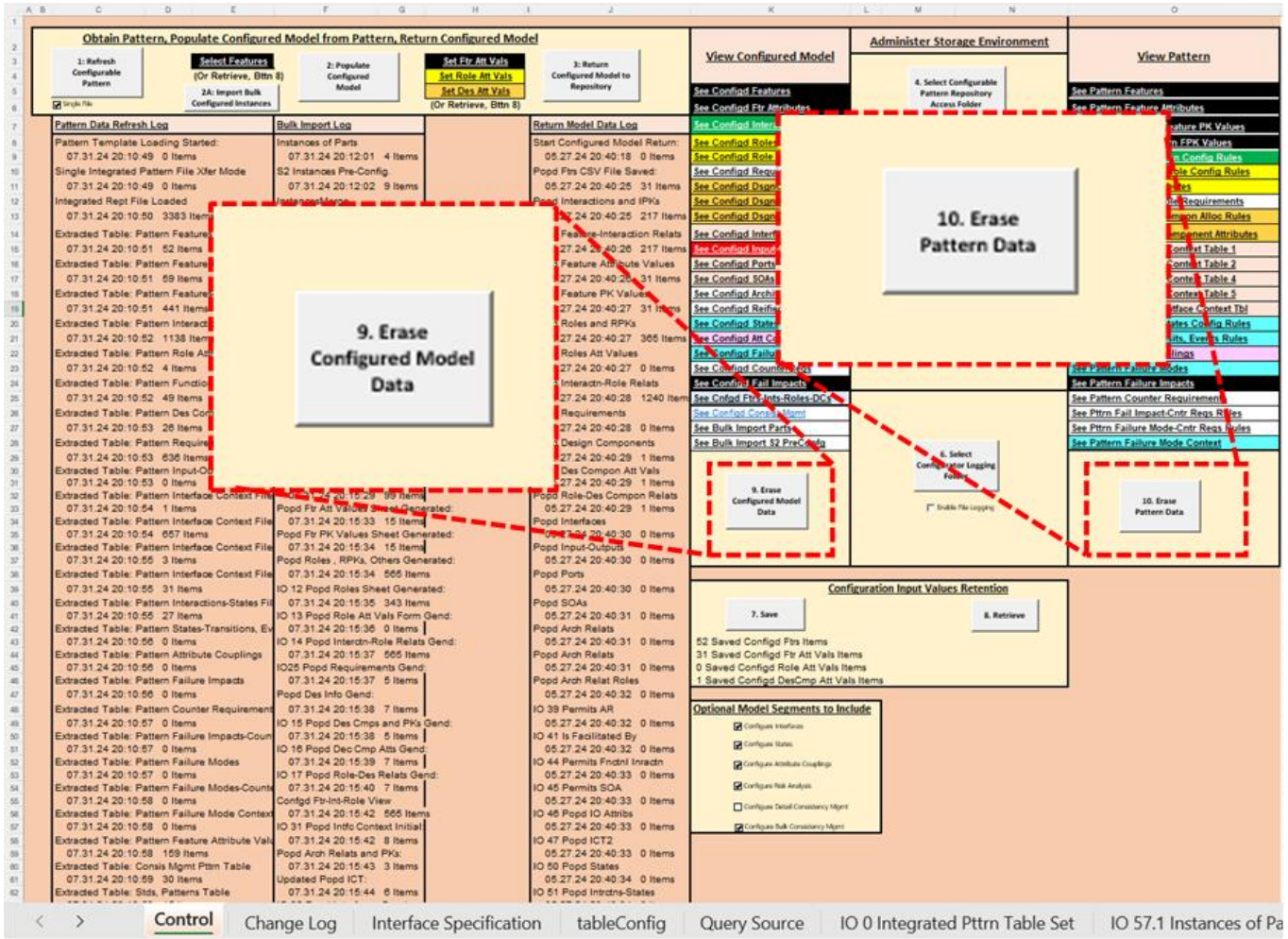




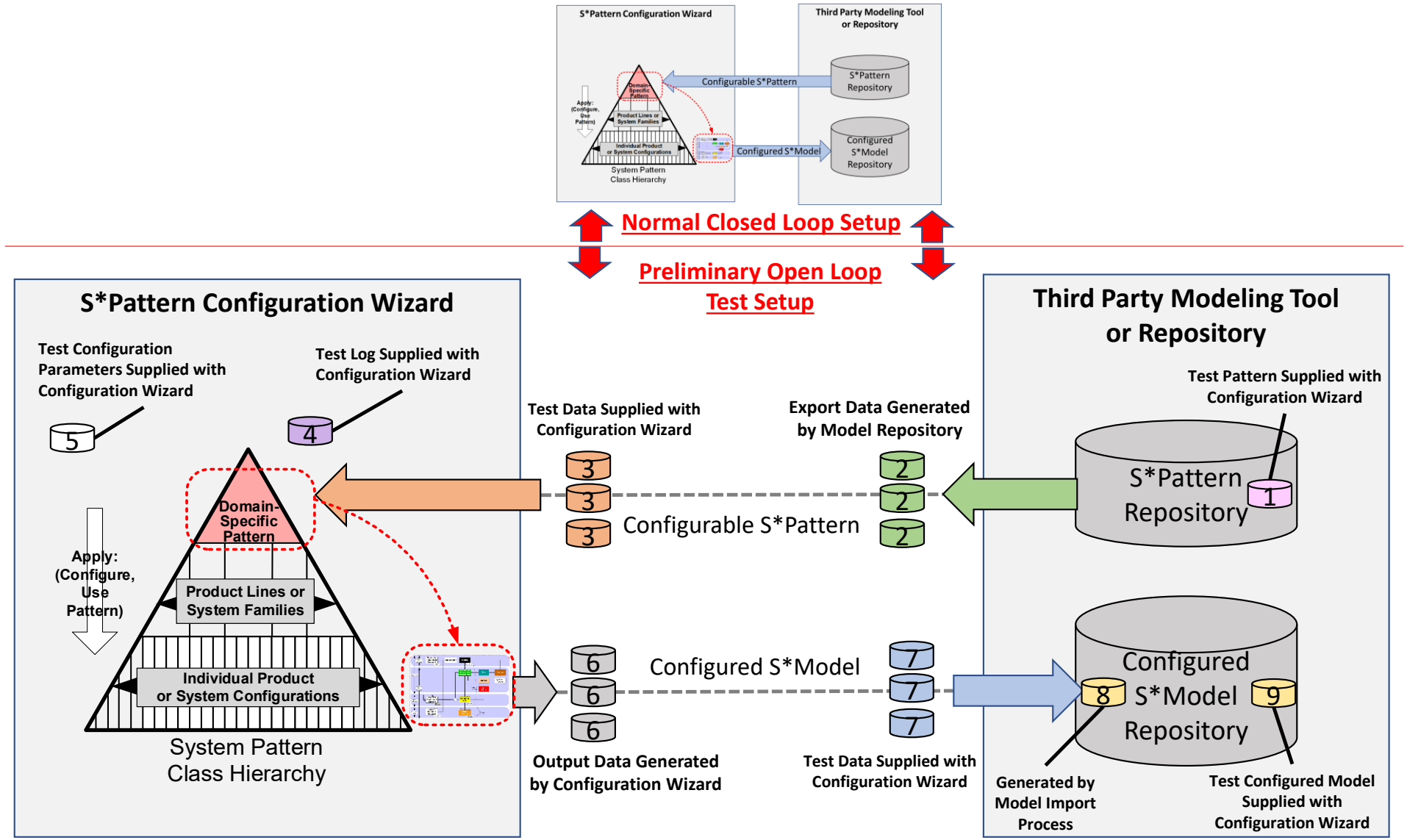
8 Configured SysML Model Details in Modeling Tool



Erasing Configured Model and/or Pattern Data in the Configuration Wizard (this does not erase any pattern or model data in Cameo)



Configuration Wizard User Guide recommends procedure for initial installation and integration test



In-Class Exercise Session 6

1. Use the Configuration Wizard Guide.
    1. See Section 2.3 about performing initial integration and “open loop” testing.
    2. Perform the steps described (summarized below).

2. Use just the Configuration Wizard and the Export Tables report file provided for this session. Use button 4, 5, 1, feature selection, 2, 3 sequence on as described earlier in this presentation as well. (No CSM pattern file generation in this step, and no CSM import in this step.)

Homework for Session 6

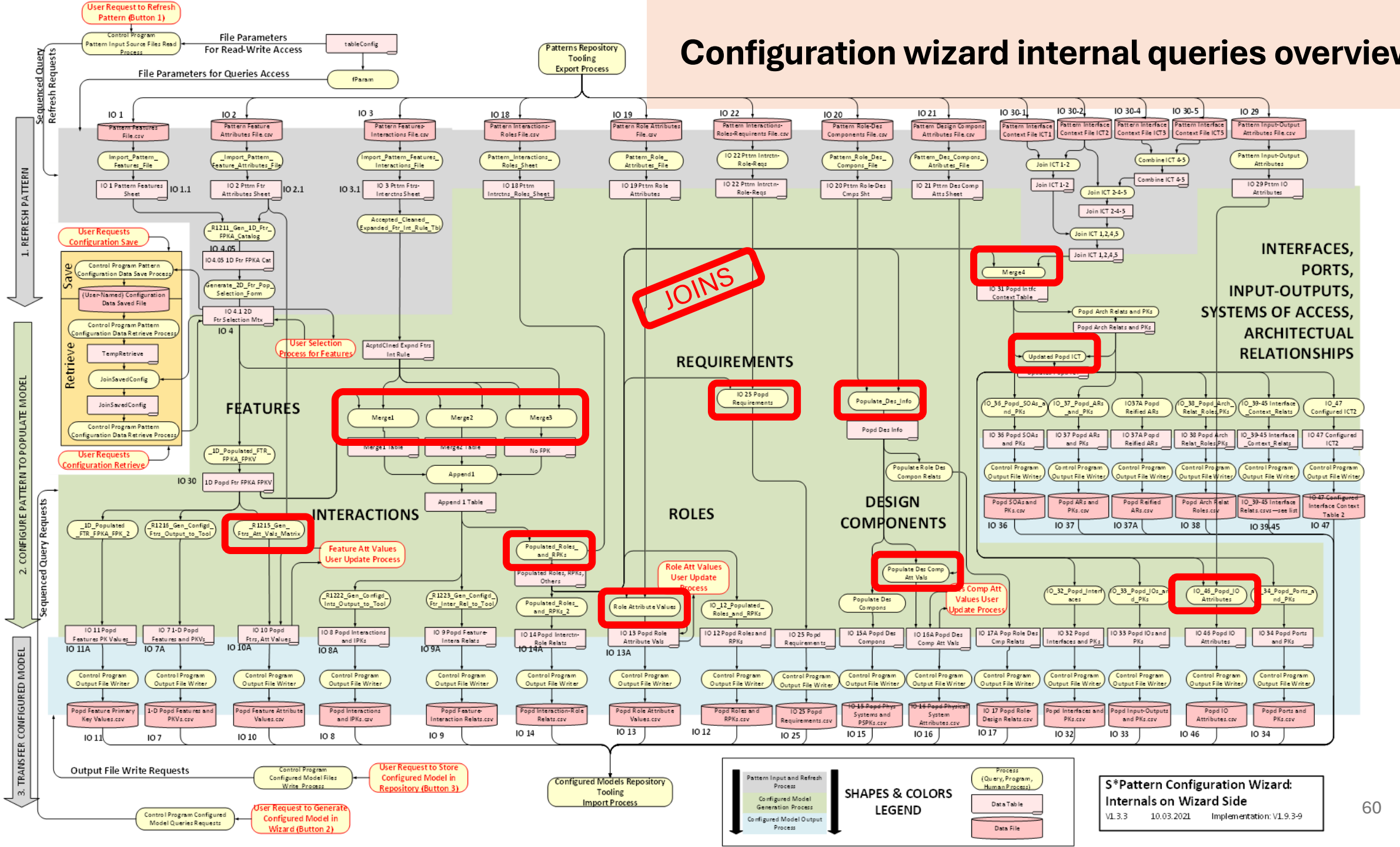
- Use just CSM and the International Power Converter model file, provided for this class/homework session. Do the CSM export tables report generation as described earlier in this table and the S* Patterns CSM Support Guide.
- Use just CSM and the set of configured model files provided for this class/homework session. Run the Model HLR DLR import map group and inspect the resulting CSM model in the CSM browser. Run additional map groups as well.
- For additional HW, perform the all of the steps for a complete end-to-end configuration.

Discussion, Q&A.

-
-
-
-
-
-



Configuration wizard internal queries overview

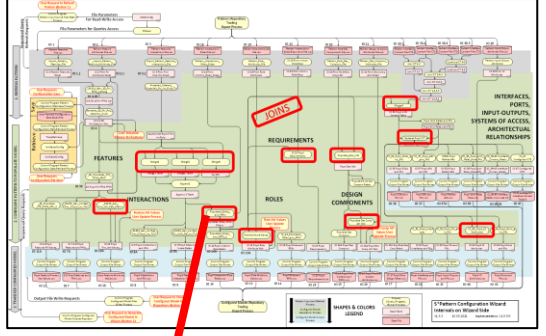


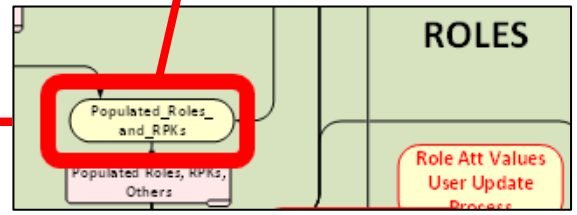
Power Query M-Code

```
////////////////////////////////////////////////////
//
//  Query that generates populated roles from populated interactions joined with Pattern Interactions-Roles Configuration Rules
//
////////////////////////////////////////////////////
// Begin with Interactions-Roles configuration rules from pattern, inner joined with Populated Interactions; joined on Interactions
let
  Source = Table.NestedJoin(#"Pattern Interactions-Roles Sheet", {"Interaction Name"}, Append1, {"Interaction Name"}, "Append1", JoinKind.Inner),
  #"Expanded Append1" = Table.ExpandTableColumn(Source, "Append1", {"IPK Value", "Configured Interaction Name"}, {"IPK Value.1", "Configured Interaction Name"}),
// Separate rows with configuration rule = *ANY*, allowing any interaction PK values
  #"Filtered Rows with IPK Rule=*ANY*" = Table.SelectRows(#"Expanded Append1", each [IPK Value] = "*ANY*"),
// Separate rows with configuration rules that directly match specific IPK values in populated Interactions
  #"Filtered Rows with IPK Value=IPK Rule" = Table.SelectRows(#"Expanded Append1", each [IPK Value] = [IPK Value.1]),
//  Combine the two sets of resulting rows, for combined populated Roles
  #"Combined Rows" = Table.Combine({#"Filtered Rows with IPK Value=IPK Rule",  #"Filtered Rows with IPK Rule=*ANY*" }),
//  For rows with RPK configuration rule equal to "IPK", indicating RPK value should be set to IPK value, do so
  #"Added Conditional Column" = Table.AddColumn(#"Combined Rows", "IPK Flag", each if Text.Contains([RPK Rule], "IPK") then [IPK Value.1] else ""),
//  For rows with RPK configuration rule equal to "I2PK", indicating RPK value should be set to second half of IPK value (portion between "-" and "]", do so
  #"Added Conditional Column3" = Table.AddColumn(#"Added Conditional Column", "I2PK Flag", each if Text.Contains([RPK Rule], "I2PK") then   Text.BetweenDelimiters([IPK Value.1]&"]", "-", "]")  else ""),
//  For rows with text between delimiters in form of /<string>/, extract that string for use in RPK value
  #"Inserted Text Between Delimiters" = Table.AddColumn(#"Added Conditional Column3", "Text Between Delimiters", each Text.BetweenDelimiters([RPK Rule], "/", "/"), type text),
//  Determine whether there are two parts to RPK value, creating a delimiter of (-) if so, otherwise no delimiter
  #"Added Conditional Column1" = Table.AddColumn(#"Inserted Text Between Delimiters", "Delimiter", each if [IPK Flag]&[I2PK Flag] = "" then "" else if [IPK Flag]&[I2PK Flag] = null then "" else if [Text Between
Delimiters] = "" then "" else "-"),
//  Combine strings for RPK value, including delimiter if needed
  #"Inserted Merged Column" = Table.AddColumn(#"Added Conditional Column1", "RPK Value", each Text.Combine({[IPK Flag]&[I2PK Flag], [Delimiter] , [Text Between Delimiters]}), type text),
//  Remove cols no longer needed; rename col hdgs; reorder cols
  #"Removed Columns" = Table.RemoveColumns(#"Inserted Merged Column",{"RPK Rule", "IPK Flag", "I2PK Flag", "IPK Value", "Text Between Delimiters", "Delimiter"}),
  #"Renamed Columns" = Table.RenameColumns(#"Removed Columns",{{"IPK Value.1", "IPK Value"}}),
  #"Reordered Columns" = Table.ReorderColumns(#"Renamed Columns",{"Interaction Name", "IPK Value", "Role Name", "RPK Value"}),
//  Create configured role name as combination of parent pattern role name and RPK value in [brackets].
  #"Added Conditional Column2" = Table.AddColumn(#"Reordered Columns", "Configured Role Name", each if [RPK Value] = null then [Role Name] else if [RPK Value] = "" then [Role Name] else [Role Name]&"
["&[RPK Value]&"]"),
//  Remove any duplicate rows.
  #"Removed Duplicates" = Table.Distinct(#"Added Conditional Column2")
in

  #"Removed Duplicates"
```

JOIN





Wizard internal controls overview

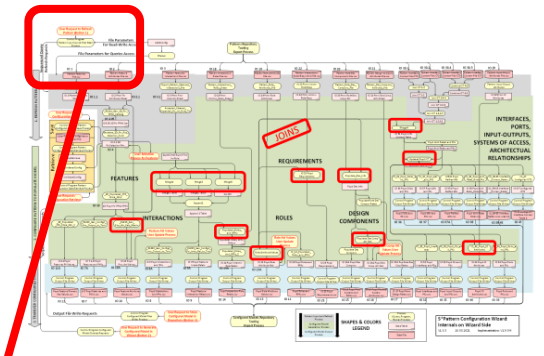
File Edit View Insert Format Debug Run Tools Add-Ins Window Help

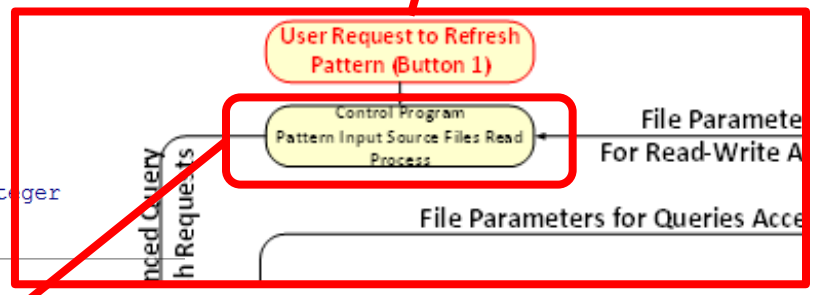
Ln 1, Col 1

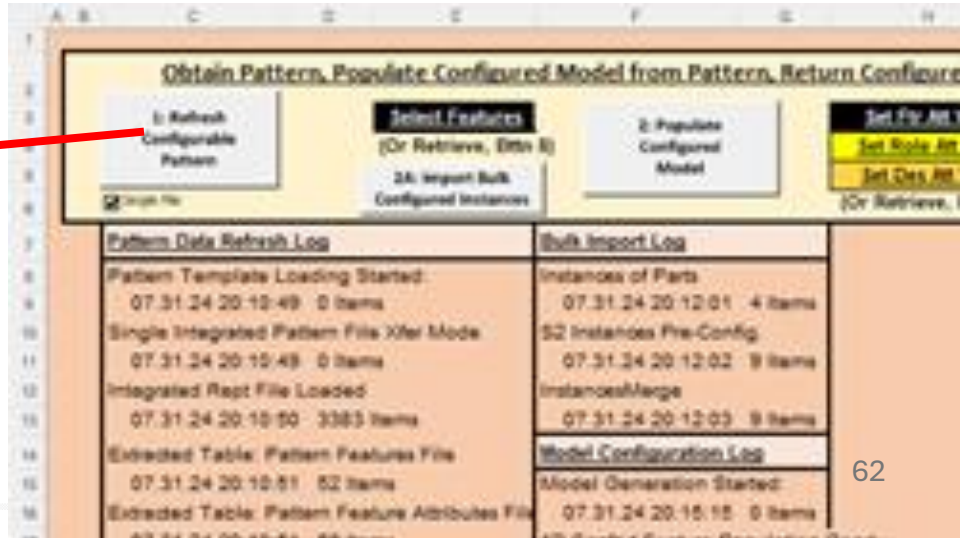
(General) (Declarations)

```
'---------------------------------------
'
' Pattern Configuration Wizard Control Program
'
'     Controls overall flow of pattern configuration process, including queries
'     Obtains local pattern from repository tool
'     Returns configured model, generated from pattern,back to repository tool
'     See also wizard Interface Specification, built into wizard.
'     See also Pattern Configuration Wizard user documentation
'     See also query code and documentation, built into wizard.
'
'  Copyright 2021 System Sciences, LLC.
'  Available under Creative Commons CC SA BY public licensing.
'
'---------------------------------------
Dim SinglePatternFileEnabled, FileLoggingEnabled, BackgroundQueryEnabled As Long
Dim NumOfFileTypes, MaxTableRows, InputFilesQty, IntegratedReptFileRowNum, FirstFileRowNum, LastFileRowNum As Integer
Dim TableName, TableRangetoExtract, SepFileName, ControlFile, ColID, RowID As String
Dim TableRange As Range

Sub Macro5()
'--------------------------------
'
'   Button B1: Refresh Configurable Pattern from Model Repository
'
'--------------------------------
Dim Msg As String
Dim LogPath As String
Dim SourceFile, DestinationFile As String
Dim MyFSO As FileSystemObject
'CalculationsOff    ' Turns off formula propagation for speed during DB operations
ColID = "C"     ' Set column and row of progress log output on control sheet
RowID = 8
FirstRowID = 8  ' Set range of log space to be cleared prior to logging.
LastRowID = 87
ClearProgressLog FirstRowID, LastRowID, ColID  ' Clear the progress log
ProgressLog RowID, ColID, "Pattern Template Loading Started:", 0  ' Start logging.
'Disable_Background_Refresh
BackgroundQuery = False
'-----------------------------------------------
'
' Set Full Path Names to Pattern Input Files, Based on User Admin of Input File Locs
'
```







SysML 2.0 API services, OWL, other technologies

- It is desirable to be able to perform these pattern configurations within a modeling tool, avoiding the in and out mapping issues.
- But we would also like to avoid having to encode the configuration algorithm separately for each COTS modeling tool.
- And, we would like for the technology involved to be efficient in performing what amounts to a series of JOINs.
- Two emerging options suggest this will become even more possible in the future:
  1. The SysML V2.0 API Services may provide an appropriate set of primitives to allow a generic algorithm description to drive different modeling tools through a standards-based API, at least for the SysML case.
  2. The closer integration of SysML and other modeling toolsets with Semantic Technologies implies we may be able to perform SPARQL JOINS or similar operations without the semantic mapping in and out steps.
- The INCOSE ST4SE project was used as a vehicle to explore this in an open community activity.

References

1. “Formal S*Metamodel”, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:systematica_8_metamodel_v8.0.61.pdf
2. “S*Metamodel Mapping for MagicDraw/Cameo Systems Modeler”, Version 19 https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:systematica_mapping_for_magicdraw_csm_v1.12.1.pdf
3. “What Is the Smallest Model of a System?”, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:what_is_the_smallest_model_of_a_system_v1.4.4.pdf
4. “Maps Versus Itineraries: A Systems Engineering Lesson from Ancient Mariners”, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:maps_vs._itineraries_v1.3.1.pdf
5. “Guide to Systematica Patterns in Cameo Systems Modeler”: https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:guide_to_systematica_patterns_in_cameo_systems_modeler_v1.4.6.pdf
6. “Guide to S* Pattern Configuration Wizard”, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:guide_to_the_pattern_configuration_wizard_v1.5.4.pdf

References

7. Schindel, W., Peterson, T. 2016. “Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques”. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:pbse_tutorial_glrc_2016_v1.7.4.pdf
8. Patterns WG. 2019b. “Methodology Summary: Pattern-Based Systems Engineering PBSE), Based On S*MBSE Models” https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:pbse_extension_of_mbse--methodology_summary_v1.6.1.pdf
9. “What Is the Smallest Model of a System?”, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:what_is_the_smallest_model_of_a_system_v1.4.4.pdf
10. Oil Filter Product Line Family Extract, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:extract_from_--_system_requirements_document_v1.2.2a.pdf
11. Extracts from S*Patterns, https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:attachment_1--example_extracts_from_s-patterns_v1.2.4.pdf
12. Patterns WG. 2021. INCOSE MBSE Patterns Working Group Web Site: https://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
13. Schindel, W. 2021. “Variational Forces of Modularity: Coupled Macro and Micro Patterns in the Innovation Ecosystem”. Presented at PLE Momentum 2021 Conference. https://www.omgwiki.org/MBSE/lib/exe/fetch.php?media=mbse:patterns:the_forces_of_modularity_v1.3.3.pdf