ISTANBUL SOFTWARE TESTING CONFERENCE

SPL Testing at Industrial Scale: Minimal Configuration Selection

Burcu Ergun, PhD
Test Automation Team Leader, Airties

#ISTC2026

Talk Overview

SPL Testing

Motivation & Problem

Challenges in SPL Testing

Approach

Minimal Configuration

Experimental Results

Industrial Impact & Conclusion

Software Product Line Testing

What is an SPL (Software Product Line)?

A software engineering approach used to develop a family of products from a single, shared software core (baseline). Instead of writing separate software for every new product, features are systematically turned ON or OFF based on the specific hardware or model requirements.

Imagine developing firmware for a company's **Access Point (AP)** product line:

The Shared Core: Every AP model shares the same foundational software stack (e.g., Linux Kernel, Basic IP Routing, Device Management).
The Variations (SPL Customization):

- **Home AP:** Configured to only run basic Wi-Fi and simple WPA2 security.
- **Enterprise AP:** Configured from the *same* codebase to activate advanced features like WPA3, Fast Roaming, Guest Captive Portal, and VLAN Management.

Feature Model Relationships

Mandatory: Must be included in every product variant. *Wi-Fi Example:* **IP Routing / DHCP Server** — foundational software that every single router must have to function.

Optional: Can be included or excluded based on the hardware model/tier. *Wi-Fi Example:* **Parental Controls or Guest Network** — premium software features enabled only on high-end or family-focused models.

XOR (Alternative): Exactly **one** option must be chosen from the group. *Wi-Fi Example:* **Wi-Fi Standard Mode** — The device core chip operates in *Wi-Fi 6* OR *Wi-Fi 7* hardware mode (cannot be both simultaneously at the base level).

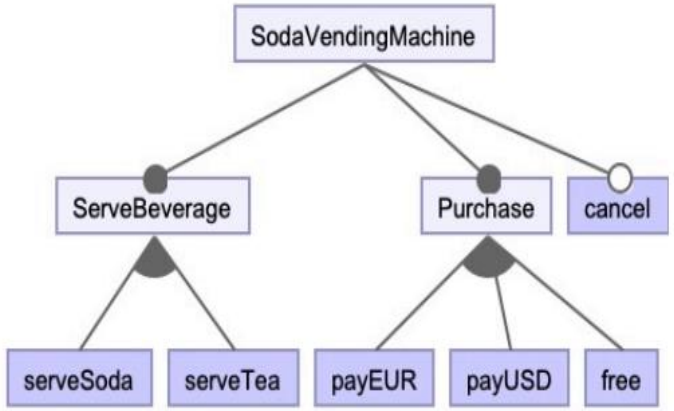
OR: At least **one or more** options can be selected at the same time. *Wi-Fi Example:* **Security Protocol Support** — A router can support *WPA2*, *WPA3*, or *both* simultaneously to allow older and newer devices to connect.

Set Of SPL Feature Types

Four Feature Types in SPLs:

Mandatory (M)	Optional (O)	XOR groups (Exclusive OR)	OR groups

Constraint-aware processing is essential for valid configuration generation.



Motivation & Problem

In Software Product Lines (SPL), the number of possible configurations grows exponentially.

Why One Configuration is Not Enough for Testing?
Testing only an enterprise-grade router (all options enabled) misses bugs where a home router crashes because it lacks a specific optional module. We need an algorithmic way to cover them all with the fewest configurations.

Exhaustive testing becomes infeasible. Different combinations of Wi-Fi standards (XOR) and Security protocols (OR) create dozens of firmware variants.

Traditional regression testing approaches:
- Ignore feature constraints
- Miss behavioral diversity across configurations

As a result:
- Redundant tests are executed
- Critical feature interactions may be missed

👉 **Key Problem:**
How can we select a minimal set of valid configurations
that still ensures full feature coverage?

Challenge in SPL systems

Fast response required for evolving products
→ WiFi (protocols, devices) & Battery systems (EMS, PCS, capacity)

Limited testing resources
→ Same test infrastructure, multiple product families

High variability in configurations
→ Feature combinations grow exponentially

Diverse deployment environments
- Different customers, regions, and regulations

Test explosion
→ Thousands of valid configurations

Key challenge
👉 Selecting the right configurations to test

Approach

Traditional approach:
Test many configurations → high cost

Our approach:
Select a minimal set of valid configurations that still ensures full feature coverage
→ Fewer tests
→ Same coverage
→ Faster regression

Based on:
Most Complex Product (MCP)
Core & Variability Coverage (CVC)

Minimal Configuration Approach

Why Minimal Configuration is Needed

Single configuration is NOT enough

→ Due to feature constraints (XOR, excludes)
→ Not all features can coexist in one product

Therefore, we select a minimal set of valid configurations that together ensure full feature coverage

→ Complete coverage
→ No invalid combinations
→ No redundant configurations

Benefits:

→ Reduced number of configurations
→ Faster regression cycles
→ Improved fault localization

Step 1: Generate the Most Complex Product (MCP)

Goal:
Maximize feature and interaction coverage in a single configuration

Includes:
- All mandatory features
- Optional features (if valid)
- One feature from each XOR group
- At least one from each OR group

👉 Maximizes coverage in one pass

How Do We Generate the Complex Product?

We traverse the feature model tree

At each step:
- Add valid features (respect constraints)
- Select one feature from XOR groups
- Select features from OR groups

Invalid paths are pruned early

👉 No exhaustive enumeration required

Why Start with a Complex Product?

Captures maximum feature interaction
in a single execution

Provides a strong baseline configuration

Reveals feature conflicts early

👉 Guides the selection of remaining configurations

Step 2: Identify Uncovered Features

After generating the complex product:
Some features remain uncovered
(due to XOR, excludes, constraints)

Remaining features:
R = F – Coverage(P)

Used to:
- Identify missing feature coverage
- Guide additional configuration selection

👉 Ensures full feature coverage

Step 3: Generate Simple Products

For each uncovered feature f ∈ R:

Generate a minimal valid configuration including:

- Mandatory features
- Feature f
- Required dependencies (Dep(f))
- One OR-group member (if needed)

👉 Covers missing features with minimal overhead

How Are Simple Products Generated?

Focus only on uncovered features (R)

For each feature:
- Add required dependencies
- Respect XOR / OR constraints
- Validate incrementally

Invalid configurations are pruned early

👉 Efficient, constraint-aware generation

Final Configuration Set

$P = \{p_c\} \cup \{p_f \mid f \in R\}$

Where:
- p_c: complex product (max coverage)
- p_f: simple products (target missing features)

Outcome:
- All features covered at least once
- All configurations are valid
- Redundancy minimized

👉 **Minimal set, full coverage**

Ensuring Full Feature Coverage

Validation:

Coverage(P) = F

Constraint Satisfaction:
- XOR: at most one feature selected
- OR: at least one feature selected

If coverage is incomplete:
→ generate additional configurations

👉 Guarantees valid and complete coverage

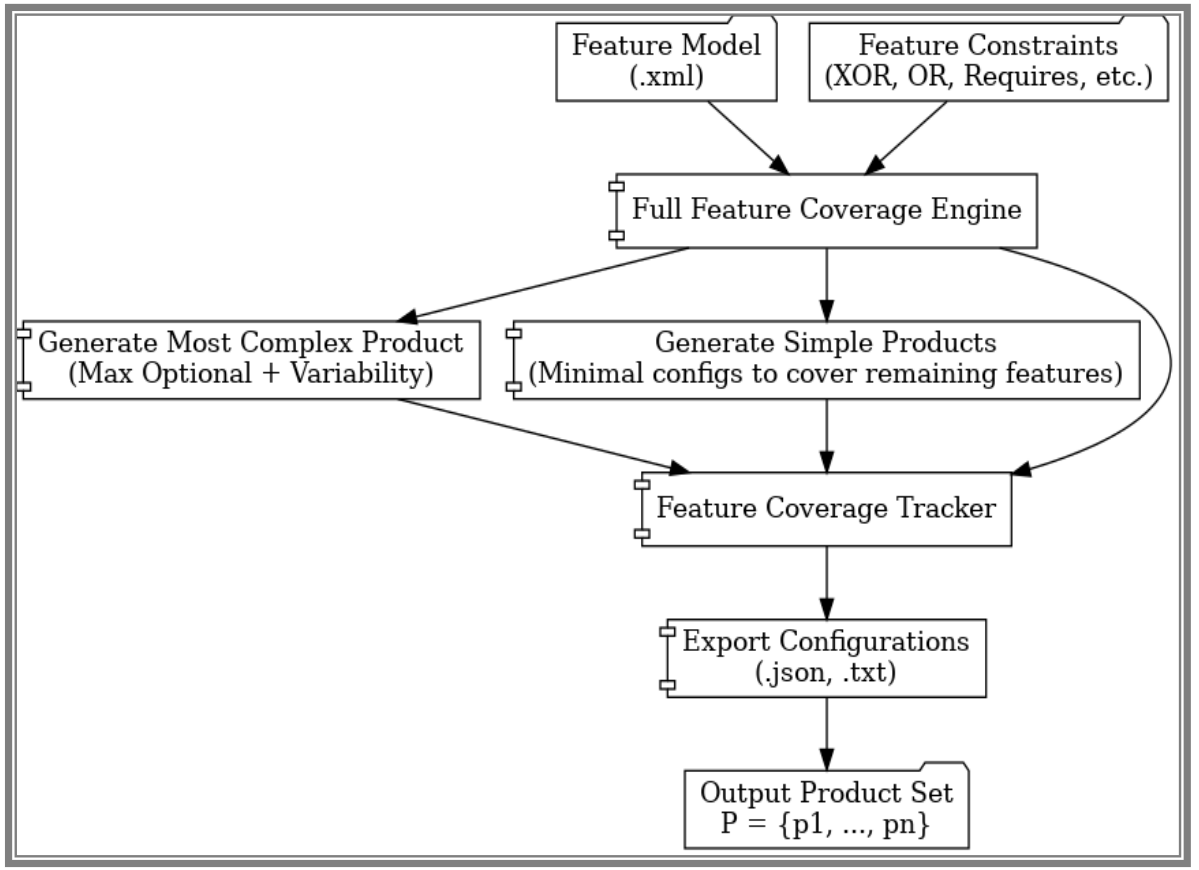
Overall Approach

Goal:
Achieve full feature coverage with minimal configurations

Process:
1. Generate Complex Product (MCP)
2. Identify uncovered features (R)
3. Generate Simple Products
4. Validate coverage

Constraint Handling:
Ensured at every step (XOR, OR, dependencies)

👉 Scalable and practical for real SPLs



Why Traditional Testing Fails — and Our Approach Scales

Traditional Methods:

Exhaustive Testing
✓ Full coverage
✗ 2^66 configurations → infeasible

Pairwise Testing
✓ Reduces combinations
✗ Still requires 132 (AP) / 144 (Battery) configurations

Combinatorial Testing
✓ Better interaction coverage
✗ 45K+ configurations → costly

Our Approach:

- AP SPL → 3 configurations (vs 132 pairwise)
- Battery SPL → 5 configurations (vs 144 pairwise)

All configurations:
- Fully cover features
- Respect all constraints
- Avoid redundancy

👉 From hundreds (or billions) to just a few configurations

Real-World SPL Complexity (XOR Analysis)

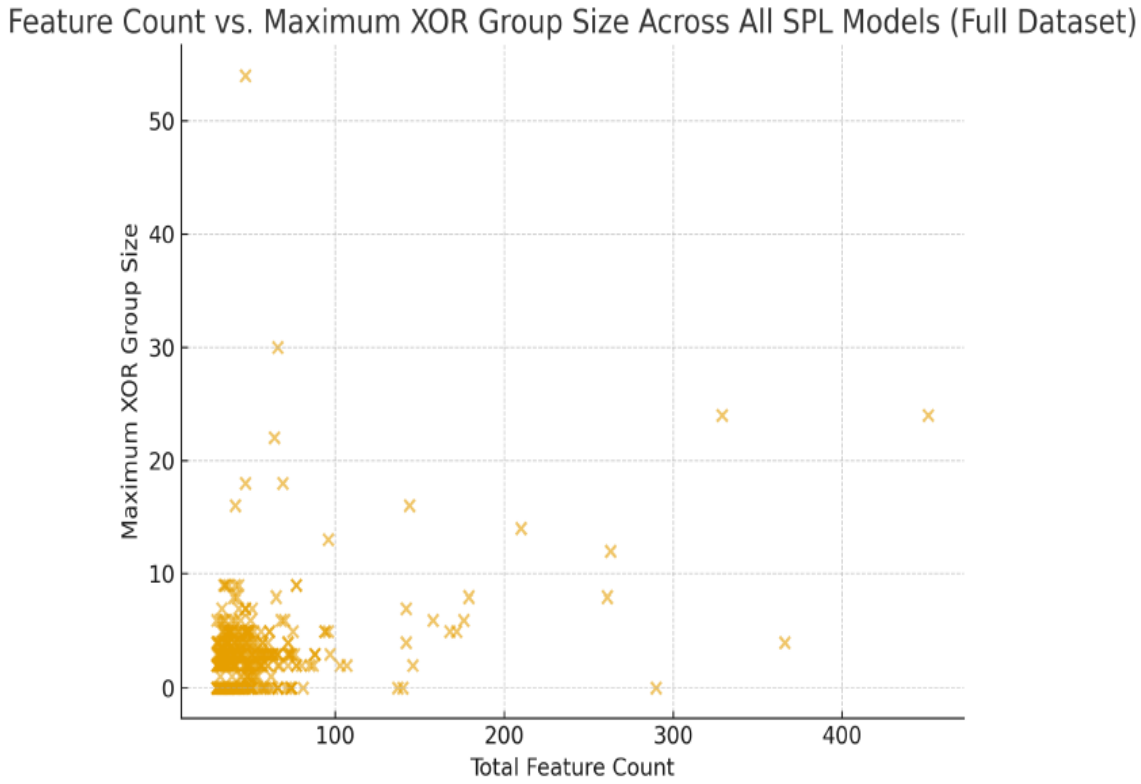
Analysis of 376 real-world SPL models reveals:

- Most XOR groups are small (2–6 features)
- Large XOR groups (20–54 features) are rare but critical

Key Insight:
Even a few large XOR groups drastically increase configuration complexity

Implication:
Traditional testing strategies fail to scale in these cases

👉 Constraint-aware selection becomes essential



Experimental Results

From Model to Configurations

Input:
Feature Model (XML)

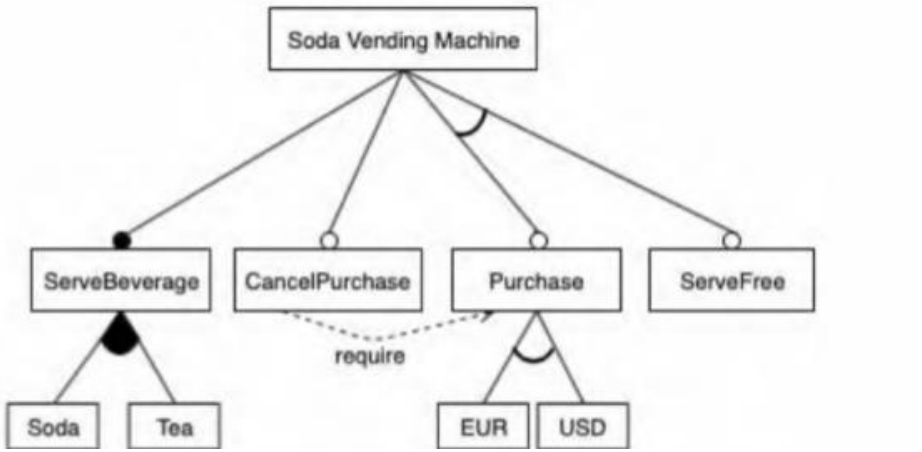
Process:
Iterative feature selection under constraints

Output:
- Complex Product (MCP)
- Simple Products covering remaining features

👉 Generates valid configurations step-by-step

feature_tree.xml

```
Bind to grammar/schema...
<Feature name="Soda Vending Machine" group="
    <Feature name="ServeBeverage" group="Ma
        <Feature name="Soda" group="OR"/>
        <Feature name="Tea" group="OR">
        </Feature>
    </Feature>
    <Feature name="CancelPurchase" group="O
        <RequiredFeature name="Purchase"/>
    </Feature>
    <Feature name="Purchase" group="XOR">
        <Feature name="EUR" group="XOR"/>
        <Feature name="USD" group="XOR"/>
    </Feature>
     <Feature name="ServeFree" group="XOR">
    </Feature>
```



Configuration Reduction in Industrial SPLs

Wireless Access Point (AP SPL):
- 3 configurations vs 132 (pairwise)

→ 44× reduction

Battery Management System SPL:
- 5 configurations vs 144 (pairwise)

Despite:
- High feature variability (WiFi systems)
- Dense constraints & safety-critical dependencies (Battery systems)

Our approach achieves:
- Full feature coverage
- Valid configurations only
- Minimal redundancy

👉 Drastic reduction in testing effort at industrial scale

Feature	P_C	P_1	P_2
AP_SPL	✓	✓	✓
ACS	✓		
Apps	✓		
Band_Steering	✓		
Blue	✓		
CAC	✓	✓	✓
ChannelManagement	✓	✓	✓
ClientSteering	✓		
DFSReEntry	✓		
DHCP	✓		
Datapath	✓		
Device	✓	✓	✓
EnergySave	✓		
EnergySaver_App	✓		
GHz24	✓		
GHz5	✓	✓	✓
GuestNetwork	✓		
IGMP	✓		
IPTv	✓		
LAN	✓	✓	✓
LED	✓		
Login	✓	✓	✓
MacFiltering	✓		
Multicast	✓		
Network	✓	✓	✓
NTP	✓		
Parental	✓		
Parental_App	✓		
Password	✓		
Password5	✓	✓	✓
Password_Guest	✓		
PatienceTime	✓		
QoS	✓		
Radio	✓	✓	✓
RadarDetection	✓		
Red	✓		
Reset	✓	✓	✓
SSID	✓		
SSID5	✓	✓	✓
SSID_Guest	✓		
SecurityType	✓		
SecurityType5	✓	✓	✓
SimultaneousUpgrade	✓		
Steering	✓	✓	
SteeringOld	✓		
Telnet	✓		
TelnetPassword	✓		
TelnetPort	✓		
TelnetUsername	✓		
TemperatureSense	✓		
UnifiedSteering		✓	
Upgrade	✓		
Vision	✓		
WEBExpirationTime		✓	✓
WEBPassword	✓		
WEBUI	✓	✓	✓
WEBUsername	✓	✓	✓
WEBCredential		✓	
White	✓		
Wireless	✓	✓	✓
WPS	✓		
WPS_App	✓		
ZeroWaitDFS	✓		

Feature	P_C	P_1	P_2	P_3	P_4
BatteryModel	✓	✓	✓	✓	✓
Battery	✓	✓	✓	✓	✓
Management System	✓	✓	✓	✓	✓
BMS	✓	✓	✓	✓	✓
Voltage Monitoring	✓	✓	✓	✓	✓
Temperature Monitoring	✓	✓	✓	✓	✓
Current Monitoring	✓				
SOC Estimation	✓				
SOH Estimation	✓				
Fault Detection	✓				
Cell Balancing Support	✓				
Communication Interface	✓				
CAN Bus	✓				
Modbus	✓				
UART	✓				
BLE	✓				
Safety Features	✓	✓	✓	✓	✓
Overcharge Protection	✓	✓	✓	✓	✓
Overdischarge Protection	✓	✓	✓	✓	✓
Overcurrent Protection	✓				
Short Circuit Protection	✓				
Thermal Runaway Detection	✓				
Fuse	✓	✓	✓	✓	✓
Contactor	✓				
Charging	✓	✓	✓	✓	✓
Fast Charging	✓				
Wireless Charging	✓				
Dual Charging Ports	✓				
CC/CV Charging	✓	✓	✓	✓	✓
Charging Profile Selection	✓	✓	✓	✓	✓
Standard	✓				
High Temperature		✓			
Low Temperature			✓		
Environmental Adaptation	✓	✓	✓		
Operating Temperature Range	✓	✓	✓		
-20 to 60C		✓			
-10 to 45C			✓		
-40 to 85C	✓				
Physical Characteristics	✓	✓	✓	✓	✓
Size Category	✓	✓	✓		
Small		✓			
Medium			✓		
Large	✓				
Weight Class	✓	✓	✓		
1-5kg		✓	✓		
¿5kg	✓				
Enclosure Type	✓	✓	✓		
Hybrid		✓	✓		
Plastic	✓				
Cell Configuration	✓	✓	✓	✓	✓
Series Connection	✓	✓	✓	✓	✓
Parallel Connection	✓				
Cell Balancing	✓	✓			
Passive Balancing	✓				
Active Balancing		✓			
Cell Chemistry	✓	✓	✓	✓	✓
Lithium-Ion	✓				
Lithium-Polymer		✓			
LFP			✓		
NMC				✓	
LTO					✓
Optional Features	✓				
Integrated Inverter	✓				
Solar Charging Support	✓				
GPS Tracking	✓				
Cloud Connectivity	✓				
Firmware OTA Update	✓				
Shock Resistance	✓	✓	✓		
Vibration Resistance	✓				
Waterproofing	✓				

Configuration Reduction in Industrial SPLs

Wireless Access Point (AP SPL):
- 3 configurations vs 132 (pairwise)

→ 44× reduction

Battery Management System SPL:
- 5 configurations vs 144 (pairwise)

Despite:
- High feature variability (WiFi systems)
- Dense constraints & safety-critical dependencies (Battery systems)

Our approach achieves:
- Full feature coverage
- Valid configurations only
- Minimal redundancy

👉 Drastic reduction in testing effort at industrial scale

Feature	P_C	P_1	P_2
AP_SPL	✓	✓	✓
ACS	✓		
Apps	✓		
Band_Steering	✓		
Blue	✓		
CAC	✓	✓	✓
ChannelManagement	✓	✓	✓
ClientSteering	✓		
DFSReEntry	✓		
DHCP	✓		
Datapath	✓		
Device	✓	✓	✓
EnergySave	✓		
EnergySaver_App	✓		
GHz24	✓		
GHz5	✓	✓	✓
GuestNetwork	✓		
IGMP	✓		
IPTv	✓		
LAN	✓	✓	✓
LED	✓		
Login	✓	✓	✓
MacFiltering	✓		
Multicast	✓		
Network	✓	✓	✓
NTP	✓		
Parental	✓		
Parental_App	✓		
Password	✓		
Password5	✓	✓	✓
Password_Guest	✓		
PatienceTime	✓		
QoS	✓		
Radio	✓	✓	✓
RadarDetection	✓		
Red	✓		
Reset	✓	✓	✓
SSID	✓		
SSID5	✓	✓	✓
SSID_Guest	✓		
SecurityType	✓		
SecurityType5	✓	✓	✓
SimultaneousUpgrade	✓		
Steering	✓	✓	
SteeringOld	✓		
Telnet	✓		
TelnetPassword	✓		
TelnetPort	✓		
TelnetUsername	✓		
TemperatureSense	✓		
UnifiedSteering		✓	
Upgrade	✓		
Vision	✓		
WEBExpirationTime		✓	✓
WEBPassword	✓		
WEBUI	✓	✓	✓
WEBUsername	✓	✓	✓
WEBCredential		✓	
White	✓		
Wireless	✓	✓	✓
WPS	✓		
WPS_App	✓		
ZeroWaitDFS	✓		

Feature	P_C	P_1	P_2	P_3	P_4
BatteryModel	✓	✓	✓	✓	✓
Battery	✓	✓	✓	✓	✓
Management System	✓	✓	✓	✓	✓
BMS	✓	✓	✓	✓	✓
Voltage Monitoring	✓	✓	✓	✓	✓
Temperature Monitoring	✓	✓	✓	✓	✓
Current Monitoring	✓				
SOC Estimation	✓				
SOH Estimation	✓				
Fault Detection	✓				
Cell Balancing Support	✓				
Communication Interface	✓				
CAN Bus	✓				
Modbus	✓				
UART	✓				
BLE	✓				
Safety Features	✓	✓	✓	✓	✓
Overcharge Protection	✓	✓	✓	✓	✓
Overdischarge Protection	✓	✓	✓	✓	✓
Overcurrent Protection	✓				
Short Circuit Protection	✓				
Thermal Runaway Detection	✓				
Fuse	✓	✓	✓	✓	✓
Contactor	✓				
Charging	✓	✓	✓	✓	✓
Fast Charging	✓				
Wireless Charging	✓				
Dual Charging Ports	✓				
CC/CV Charging	✓	✓	✓	✓	✓
Charging Profile Selection	✓	✓	✓	✓	✓
Standard	✓				
High Temperature		✓			
Low Temperature			✓		
Environmental Adaptation	✓	✓	✓		
Operating Temperature Range	✓	✓	✓		
-20 to 60C		✓			
-10 to 45C			✓		
-40 to 85C	✓				
Physical Characteristics	✓	✓	✓	✓	✓
Size Category	✓	✓	✓		
Small		✓			
Medium			✓		
Large	✓				
Weight Class	✓	✓	✓		
1-5kg		✓	✓		
¿5kg	✓				
Enclosure Type	✓	✓	✓		
Hybrid		✓	✓		
Plastic	✓				
Cell Configuration	✓	✓	✓	✓	✓
Series Connection	✓	✓	✓	✓	✓
Parallel Connection	✓				
Cell Balancing	✓	✓			
Passive Balancing	✓				
Active Balancing		✓			
Cell Chemistry	✓	✓	✓	✓	✓
Lithium-Ion	✓				
Lithium-Polymer		✓			
LFP			✓		
NMC				✓	
LTO					✓
Optional Features	✓				
Integrated Inverter	✓				
Solar Charging Support	✓				
GPS Tracking	✓				
Cloud Connectivity	✓				
Firmware OTA Update	✓				
Shock Resistance	✓	✓	✓		
Vibration Resistance	✓				
Waterproofing	✓				

Configuration Comparison Results

Our approach compared against classical testing approaches

Exhaustive enumeration, pairwise testing, and 3-wise combinatorial generation, as well as lightweight heuristics including one-enabled, one-disabled, most-enabled-disabled, and random sampling

By applying the same criteria to both SPLs, we ensured that coverage, scalability, and efficiency could be directly compared in terms of configuration reduction and test effectiveness.

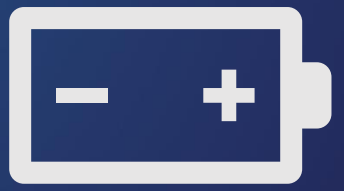
Comparison of Different Testing Approaches on AP SPL

Description	Formula	Explanation	Configuration Count
Exhaustive Testing	$2^n = 2^{66} = 7.38 \times 10^{19}$	Tests all possible configurations.	7.38×10^{19}
Pairwise Testing	$2 \times n = 132$	Ensures every feature pair is tested at least once.	132
3-wise Combinatorial	$C(n, 3) = 45760$	3-way feature combinations are tested.	45,760
One-enabled	$n = 66$	Only one feature is enabled at a time.	66
One-disabled	$n = 66$	All features are enabled except one.	66
Most-enabled-disabled	Fixed = 2	All-enabled and all-disabled configurations.	2
Random (heuristic)	Fixed = 5	Randomly selected configurations.	5
Our Approach (MCP + CVC)	Dependent on $\|XOR\|$	Optimized selection based on constraint and dependency analysis.	**3**

Comparison of Different Testing Approaches on Battery SPL

Description	Formula	Explanation	Config. Count
Exhaustive Testing	$2^n = 2^{72}$	Tests all possible feature combinations.	4.72×10^{21}
Pairwise Testing	$2 \times n = 144$	Every feature pair tested at least once.	144
3-wise Combinatorial	$C(n, 3) = 59{,}640$	Tests all 3-way feature interactions.	59,640
One-enabled	$n = 72$	Only one feature enabled at a time.	72
One-disabled	$n = 72$	All features enabled except one.	72
Most-enabled-disabled	Fixed = 2	All-enabled and all-disabled variants.	2
Random (heuristic)	Fixed = 5	Randomly sampled configurations.	5
Our Approach (MCP + CVC)	Dependent on $\|XOR\|$	Optimized selection of valid configurations.	**5**

Industrial Impact & Conclusion



Industrial Impact

Validated on real-world SPLs:

- Wireless Access Point systems (high variability, complex interactions)
- Battery Management Systems (safety-critical, constraint-dense)

Demonstrates:

- Reliable configuration generation under strict constraints
- **100% Feature Coverage (1-wise):** Ensures every single appears in at least one test configuration.
- Scalability across large and complex product lines

Industrial Relevance:

- **Minimalist & Sufficient Baseline** highly optimized configuration count, significantly reducing traditional test redundancy.
- Ensures only valid and deployable configurations
- Applicable to other large-scale product families

Complementary Strategies

What Requires Complementary Strategies:

• Critical Bound: Achieving 100% feature coverage does not mean the system is entirely bug-free. Real-world systems have dynamic and hidden dependencies.

• Higher-Order Interactions (t-way): This approach provides 1-wise coverage. It does not address complex 2-wise (pairwise) or 3-wise feature interactions, which are left as a scalable trade-off for large, highly-constrained industrial models.

• Behavioral & Hidden Dependencies: Functional, state-transition, or timing faults that occur during runtime cannot be captured by static feature coverage alone.

Takeaways & Limitations

When & Why to Use This Approach?

Fills the Industry Gap: Directly addresses the lack of configuration optimization in traditional model-based testing studies.

High Scalability: Best suited for large-scale, highly-constrained industrial software lines (**AP SPL, Battery SPL**) where testing every variant is cost-prohibitive.

Required Inputs & Core Dependencies

Accurate Feature Constraints: The strategy heavily relies on the initial assumptions of feature dependencies

High-Quality Input Models: Clean, up-to-date feature diagrams are mandatory; errors in modeling constraints directly impact final test effectiveness.

Current Limitations & Future Outlook

Generalization: Evaluated on specific industrial baselines (**AP SPL** and **Battery SPL**). Findings may need further experiments to adapt to entirely different product line architectures.

Next Steps: Future work will explore hybrid strategies to integrate pairwise testing to enhance complex fault detection without breaking scalability.

ISTANBUL
SOFTWARE
TESTING
CONFERENCE

Thank you!

Presented by Burcu ERGUN, PhD