The **software architecture** of **instrument clusters** and **cockpit systems** varies significantly depending on the **vehicle powertrain type**—ICE, BEV, Hydrogen ICE, and Fuel Cell EV—due to differences in system complexity, data sources, and user expectations.

Here’s a breakdown of the **types of software architectures** typically used, categorized by powertrain:

**🔧 1. Internal Combustion Engine (ICE)**

**🧱 Architecture Type:**

* **Distributed ECU Architecture**
* **Classic AUTOSAR-based**

**🧠 Characteristics:**

* Multiple ECUs for cluster, infotainment, HVAC, etc.
* CAN-based communication
* Real-time OS (RTOS) for safety-critical tasks
* Limited OTA capability

**📊 Cluster Focus:**

* RPM, fuel level, engine temp, gear position
* Analog-digital hybrid clusters

**⚡ 2. Battery Electric Vehicle (BEV)**

**🧱 Architecture Type:**

* **Centralized or Zonal Architecture**
* **Adaptive AUTOSAR or POSIX-based**

**🧠 Characteristics:**

* High compute domain controllers (Cockpit DCU, ADAS DCU)
* Ethernet backbone, high-speed data exchange
* OTA updates for multiple domains
* Integration with cloud services

**📊 Cluster Focus:**

* Battery SOC, range, regen braking, energy flow
* Fully digital, customizable UI

**💨 3. Hydrogen ICE**

**🧱 Architecture Type:**

* **Hybrid Distributed Architecture**
* **Classic AUTOSAR with Hydrogen-specific extensions**

**🧠 Characteristics:**

* Similar to ICE but with hydrogen-specific diagnostics
* Additional sensors for hydrogen combustion and NOx
* Limited OTA, evolving toward modular software

**📊 Cluster Focus:**

* Hydrogen level, combustion status, NOx alerts

**🔋 4. Fuel Cell Electric Vehicle (FCEV)**

**🧱 Architecture Type:**

* **Centralized/Service-Oriented Architecture (SOA)**
* **Adaptive AUTOSAR or custom Linux-based stacks**

**🧠 Characteristics:**

* High integration with energy management systems
* Real-time monitoring of fuel cell stack and hydrogen tanks
* OTA and cloud-based diagnostics
* AI/ML for predictive maintenance

**📊 Cluster Focus:**

* Fuel cell efficiency, hydrogen pressure, water emission
* Dynamic energy flow visualizations

Here’s a breakdown of **common** and **domain-specific features** for instrument clusters across different vehicle types: **ICE**, **BEV**, **FCEV**, and **Hybrid**.

**🚗 Common Features (All Powertrains)**

These features are typically present in all types of vehicles:

| **Feature** | **Description** |
| --- | --- |
| Speedometer | Displays vehicle speed (km/h or mph) |
| Odometer | Tracks total distance traveled |
| Turn Signal Indicators | Left/right blinkers |
| Warning Lights | Check engine, ABS, airbag, etc. |
| Fuel/Battery Gauge | Shows fuel or battery level |
| Gear Position Indicator | PRND display for automatic vehicles |
| Ambient Temperature | External temperature display |
| Trip Computer | Trip distance, average speed, etc. |
| Seatbelt & Door Warnings | Alerts for unfastened belts or open doors |

**⚙️ Domain-Specific Features**

**1. ICE (Internal Combustion Engine)**

| **Feature** | **Description** |
| --- | --- |
| Tachometer | Displays engine RPM |
| Engine Temperature Gauge | Monitors coolant temperature |
| Fuel Consumption Display | Real-time and average fuel usage |
| Oil Pressure Warning | Alerts for low oil pressure |
| Glow Plug Indicator (Diesel) | Pre-heating status for diesel engines |

**2. BEV (Battery Electric Vehicle)**

| **Feature** | **Description** |
| --- | --- |
| State of Charge (SoC) | Battery percentage remaining |
| Regenerative Braking Indicator | Shows energy recovery during braking |
| Range Estimator | Estimated distance remaining |
| Charging Status Display | Charging speed, time to full, etc. |
| Eco Driving Score | Feedback on efficient driving |

**3. FCEV (Fuel Cell Electric Vehicle)**

| **Feature** | **Description** |
| --- | --- |
| Hydrogen Tank Level | Displays hydrogen fuel level |
| Fuel Cell Status | Operational status of the fuel cell stack |
| Water Emission Indicator | Shows when water is being expelled |
| Range Estimator (Hydrogen) | Distance based on hydrogen level |
| Stack Temperature Monitor | Monitors fuel cell stack temperature |

**4. Hybrid (HEV/PHEV)**

| **Feature** | **Description** |
| --- | --- |
| Power Flow Diagram | Shows energy flow between engine, battery, and wheels |
| EV Mode Indicator | Indicates when running on electric power only |
| Charge/Assist Gauge | Shows when battery is charging or assisting |
| Engine Start/Stop Status | Indicates auto start/stop activity |
| Battery Level + Fuel Gauge | Dual indicators for both power sources |

Would you like a **comparison chart** or a **visual dashboard layout** showing how these features appear in real clusters?

Here’s a structured guide to writing tests across **all levels of software testing** for an **instrument cluster in a multipower terrain architecture** (like hybrid/electric vehicles):

**🧪** **1. Unit Testing**

**Goal**: Test individual functions/components in isolation.

**✅ What to Test:**

* Speedometer logic
* Battery level calculation
* Warning light control
* Power mode transitions

**🛠️ Tools:**

* Google Test (C++)
* CppUTest
* Python unittest / pytest

**🔗 2. Integration Testing**

**Goal**: Test interactions between modules (e.g., CAN bus ↔ Display logic).

**✅ What to Test:**

* CAN signal decoding and display update
* Communication between cluster and powertrain module
* Sensor data integration (e.g., temperature, speed)

**🛠️ Tools:**

* Vector CANoe / CANalyzer
* NI PXI
* Simulink Test (for model-based systems)

**📄 Sample Scenario:**

* Inject CAN signal for 100 km/h
* Verify speedometer updates correctly
* Simulate power mode switch and check display change

**🧩 3. System Testing**

**Goal**: Validate the complete instrument cluster system in a real or simulated environment.

**✅ What to Test:**

* Full boot-up sequence
* All gauges and indicators
* Display behavior under different power modes
* User interactions (buttons, touch, etc.)

**🛠️ Tools:**

* HIL (Hardware-in-the-Loop)
* Automated vision systems
* Test benches with real ECUs

**📄 Sample Scenario:**

* Start vehicle in EV mode
* Accelerate to 60 km/h
* Switch to ICE mode
* Validate all transitions and display updates

**🧪 4. Acceptance Testing**

**Goal**: Ensure the system meets business and user requirements.

**✅ What to Test:**

* Compliance with OEM specifications
* Regulatory standards (e.g., ISO 26262)
* User experience and readability
* Localization (units, languages)

**🛠️ Tools:**

* Manual test scripts
* OEM validation tools
* UX feedback sessions

**📄 Sample Scenario:**

* Verify speed units switch correctly between km/h and mph
* Check visibility of warning lights in daylight/night

**🔄 5. Regression Testing**

**Goal**: Ensure new changes don’t break existing functionality.

**✅ What to Test:**

* Re-run all unit/integration/system tests after updates
* Focus on areas affected by recent changes

**🛠️ Tools:**

* CI/CD pipelines (Jenkins, GitLab CI)
* Automated test suites

**📋 Summary Table**

| **Test Level** | **Focus Area** | **Tools Used** | **Example** |
| --- | --- | --- | --- |
| Unit | Individual functions | gTest, CppUTest, pytest | Speed logic |
| Integration | Module interactions | CANoe, NI PXI, Simulink Test | CAN ↔ Display |
| System | Full cluster behavior | HIL, test benches | EV ↔ ICE mode |
| Acceptance | User & business requirements | OEM tools, manual scripts | UX validation |
| Regression | Stability after changes | CI/CD, automated suites | Re-run tests |

Would you like a **template document** for organizing these test cases or a **sample CI/CD setup** for regression testing?

**Unit Test:**

**Scope**: Individual functions

**Tools**: Test, CppUTest, pytest

**Example**: Speed logic

**Proposal**:

Grouping of common logic function testcases and Domain specific logics function testcases

**Integration Test**

**Scope**: Module interactions

**Tools**: CANoe, Debuggers

**Example**: GPIO Register level verification, Fault injection tests for UDS,

**Proposal**: Maintain Test suits for Different architectures and reuse the common software modules across Multiple Architectures.

**Software Qualification Test**

**Scope**: Simulated Software input and output to cluster

**Tools**: CANoe, SIL benches Diagnostics, Software update

**Example**: Gear info, State of charge, HMI

**Proposal:**

1.Segregate AutoSAR and HMI Features

2. Customize the test scripts

With General functions based on the power trains

**System Test**

**Scope**: Full cluster behavior

**Tools**: HIL, test benches, Labcar

**Example**: HVAC, Engine Status check, Communication between ECU’s.

**Proposal**

High level Generic Test specifications to be deployed based on type of Vehicle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Level** | **Focus Area** | **Tools Used** | **Example** | **Multi Terrain arch proposal** |
| Unit | Individual functions | gTest, CppUTest, pytest | Speed logic | Grouping of common logic function testcases and Domain specific logics function testcases |
| Integration | Module interactions | CANoe, Debuggers | GPIO Register level verification, Fault injection tests for UDS, | Maintain Test suits for Different architectures and reuse the |
| Software Qualification | Simulated Software input and output to cluster | CANoe, SIL benches | Diagnostics, Software update  Gear info, State of charge, HMI | 1.Segregate AutoSAR and HMI Features  2. Customize the test scripts  With General functions based on the power trains |
| System | Full cluster behavior | HIL, test benches, Labcar | HVAC , Engine Status check, Communication between ECU’s | High level Generic Test specifications to be deployed based on type of Vehicle level and labcar |
| Acceptance | User & business requirements | OEM tools, manual scripts | UX validation | 1.FOTA, Software update over UDS and USB common use cases to be defined and tested  2. Feature and Variant specific checklists can be maintained to identify the use cases |
| Regression | Stability after changes | CI/CD, automated suites | Re-run tests | Common automated framework with multi test sections can be proposed to reuse test scripts across multi power terrain architecture |

**Acceptance Test**

**Scope**: User & business requirements

**Tools**: OEM tools, manual scripts

**Example** Validation of Feature Configuration

**Proposal**:

1.FOTA, Software update over UDS and USB common use cases to be defined and tested

2. Feature and Variant specific checklists can be maintained to identify the use cases

**Regression Test**

**Scope**: Stability after changes

**Tools**:CI/CD, automated suites

**Example**: Re-run tests

**Proposal**:

Common automated framework with multi test sections can be proposed to reuse test scripts across multi power terrain architecture