KOKKOLAN
YLIOPISTOKESKUS
CHYDENIUS

KOKKOLA
UNIVERSITY CONSORTIUM
CHYDENIUS

# Enhancing IoT Requirements Through Layered Contextual Information

Lasse Harjumaa

Kokkola University Consortium Chydenius, University of Jyväskylä

lasse.m.harjumaa@jyu.fi

IARIA

# About the Author

- Lasse Harjumaa gained his PhD in Information Processing Science in 2005
- Main research interests include software engineering, software quality, IoT systems and IoT development.
- He has extensive experience teaching university courses in software engineering, including project management, object-oriented development, and the Internet of Things.

# Motivation

- IoT systems operate in dynamic, heterogeneous environments
- Behavior depends on contextual assumptions (e.g. connectivity, data quality, intent)
- These assumptions often remain implicit, harming validation and traceability
- Con$^2$ introduces explicit, reusable context contracts linked to executable scenarios

# Why Context Matters in IoT

- Identical sensor inputs may require different actions depending on context
- Context accumulates across device → edge → cloud → application layers
- Many IoT requirements are conditional, not universally valid
- Implicit assumptions lead to ambiguity and misaligned expectations

# Accumulating Context

| Layer | Data | Context Type | Meaning |
|---|---|---|---|
| Device | ax, ay, az | Physical | Acceleration values |
| Edge | RMS, FFT | Operational | Vibration level |
| Cloud | Anomaly score | Analytical | Equipment health |
| Application | Maintenance action | Business / intent | Recommended action |

# Con² Overview

- Lightweight extension to Gherkin + BDD
- Separates context from behavior
- Introduces
  - Intent contexts
  - Context contracts (preconditions, invariants, postconditions)
  - Contextual scenarios
- May enable automated testing and runtime monitoring

# Intent-Driven Context Modeling

- Intent defines why the system operates and under which assumptions
- Formal structure:
  - Intent (I)
  - Internal signals (Si)
  - External inputs (Se)
  - Data quality constraints (Q)
  - Operational/safety constraints (O)
  - Optimization policies (P)

# Contracts

- Inspired by Design-by-Contract
- Applied at system level, not just method level
- Contract elements:
  - Preconditions → context assumptions
  - Postconditions → guarantees
  - Invariants → safety & quality constraints
- Supports continuous validation

# Predictive Maintenance Context (Example)

```
Context: PredictiveMaintenance
Intent: Early degradation detection with
minimal downtime
Preconditions:
- vibration_stream freshness <= 30s
- rpm_available OR rpm_quality >= 0.9
- calibration_status == valid
- sampling_rate_hz >= 5000
Invariants:
- alarm if rms_vibration_g >= 0.80 for >=
10s
- data_completeness >= 98% /day/asset
- units: acceleration=g, frequency=Hz
- anomaly_score uses approved_model_version
Postconditions:
- health_state every <= 5 min
- anomaly_score <= 10s per window
- maintenance_recommendation includes
evidence
ExternalInputs:
- asset_registry
- maintenance_history
- operating_conditions
- spare_parts_status (optional)
@context(Monitoring)
Scenario: Excessive vibration alert
Given vibration_rms > 12 mm/s
When sustained for 5s
Then maintenance alert generated
```

# Contextual Scenarios

- Standard Gherkin scenarios annotated with @context
- Scenarios inherit all assumptions from the context
- Reduces duplication
- Improves traceability from intent → context → behavior

# Process Integration

- Aimed to fit naturally into Agile/DevOps workflows

# Case Study: Smart Building Lighting

- Voice-controlled indoor lighting
- Multiple stakeholders: residents, presenters, maintainers
- Contextual assumptions:
  - User location
  - Connectivity
  - Device availability
  - Response time
  - Authorization

# Potential benefits

- Explicitness: assumptions become structured & visible
- Traceability: contexts link directly to scenarios
- Verifiability: supports automated tests & runtime monitoring
- Modularity: reusable context blocks
- Clarity: separates what the system does from under which conditions

# Conclusion

- IoT systems require context-aware requirements
- Con$^2$ demonstrates a lightweight, executable, and scalable method
- Potential in improving clarity, maintainability, and continuous validation
- It is necessary to manage evolving IoT environments and stakeholder needs