MDE4IoT: a snapshot on mission-critical IoT systems

Federico Ciccozzi

Associate Professor
Mälardalen University (Sweden) – School of Innovation, Design and Engineering
federico.ciccozzi@mdh.se

Model-Driven Engineering for the Internet of Things (2017), Linz, Austria
Acknowledgements

- Collaborative effort together with
  - Ivica Crnkovic and Patrizio Pelliccione, Chalmers University of Technology and University of Gothenburg
  - Davide Di Ruscio, University of L’Aquila
  - Ivano Malavolta, Vrije Universiteit Amsterdam
  - Romina Spalazzese, Malmö University
Mission-Critical IoT systems (MC-IoT)

- Subset of IoT systems
Mission-Critical IoT systems (MC-IoT)

- Subset of IoT systems
- Criticality is pervasive
  - Simple RFID tag
  - Database
  - A single autonomous vehicle
  - A fleet of autonomous vehicles
Mission-Critical IoT systems (MC-IoT)

- Subset of IoT systems

- Criticality is pervasive
  - Simple RFID tag
  - Database
  - A single autonomous vehicle
  - A fleet of autonomous vehicles

- Failure leads to severe (possibly catastrophic) consequences
Mission-Critical IoT systems (MC-IoT)
Mission-Critical IoT systems (MC-IoT)
Mission-Critical IoT systems (MC-IoT)
Mission-Critical IoT systems (MC-IoT)
Mission-Critical IoT systems (MC-IoT)

User

SMART object

Connected physical entity

Network

Connects to 1..*

Has

Context

Has or defines 1..*

Interoperates with 1..*

Interoperates with 1..*

Computational service

Uses 1..*

Interoperates with 1..*

Computational capability

Tag

Sensor

Actuator

Storage

User interface
Mission-Critical IoT systems (MC-IoT)
Mission-Critical IoT systems (MC-IoT)

Diagram:
- Smart object
- Connected physical entity
- Network
  - Connects to 1..*
- Context
- User
  - Interacts with 1..*
- Goal
  - Has or defines 1..*
  - Realizes 1..*
- Thing
  - Interoperates with 1..*
- Computational service
  - Runs 1..*
  - Interoperates with 1..*
- IoT application
- Resource
- Communication protocol
- User interface
- Computational capability
  - Tag
  - Sensor
  - Actuator
  - Storage
  - User interface
Mission-Critical IoT systems (MC-IoT)
An example: a smart surveillance system
An example: a smart surveillance system

www.iconexperience.com
An example: a smart surveillance system
An example: a smart surveillance system
An example: a smart surveillance system
An example: a smart surveillance system
An example: a smart surveillance system
An example: a smart surveillance system
Core characteristics of MC-IoT

- Dependability
Core characteristics of MC-IoT

- Dependability
- Safety
Core characteristics of MC-IoT

- Dependability
- Safety
- Security
Core characteristics of MC-IoT

- Dependability
- Safety
- Security
- Timeliness
Core characteristics of MC-IoT: Dependability

- Extent to which users can trust system
- Ultrareliable, fault-tolerant system
- Takes into account reliability and availability
Core characteristics of MC-IoT: Dependability

- Extent to which users can trust system
- Ultrareliable, fault-tolerant system
- Takes into account reliability and availability

Available solutions
- Fault-tree analysis for reasoning on probability of permanent faults in one or more "thing"  

---

Core characteristics of MC-IoT: Dependability

- Extent to which users can trust system
- Ultrareliable, fault-tolerant system
- Takes into account reliability and availability

Available solutions

- Fault-tree analysis for reasoning on probability of permanent faults in one or more ”thing”\(^1\)
- IoT device virtualization to support and implement dependability patterns tailored to target requirements\(^2\)


Core characteristics of MC-IoT: Safety

- Ability to detect and prevent unintended behaviours
- Special focus on actuators
Core characteristics of MC-IoT: Safety

- Ability to detect and prevent unintended behaviours
- Special focus on actuators

- Available solution
  - Hazard analysis and system-safety-engineering principles
Core characteristics of MC-IoT: Safety

- Ability to detect and prevent unintended behaviours
- Special focus on actuators

- Available solution
  - Hazard analysis and system-safety-engineering principles

- Challenges
  - Heterogeneity of things
  - Lack of standardization
  - Ineffectiveness of traditional safety approaches

---

Core characteristics of MC-IoT: Security

- Prevention of physical-level malicious attacks
- Enforcement of system-level trust
- Privacy assurance
Core characteristics of MC-IoT: Security

- Prevention of physical-level malicious attacks
- Enforcement of system-level trust
- Privacy assurance

Available solutions
- Encryption mechanisms, secure communication, sensor data protection
Core characteristics of MC-IoT: Security

- Prevention of physical-level malicious attacks
- Enforcement of system-level trust
- Privacy assurance

- Available solutions
  - Encryption mechanisms, secure communication, sensor data protection

- Challenges\(^1\)
  - Security structures combining control and information access
  - Encryption keys management
  - Security laws and regulations
  - Requirements for emerging applications/behaviours/configurations

Core characteristics of MC-IoT: Timeliness

- System functional correctness depends on the timing of its actions (together with the functionality itself)
Core characteristics of MC-IoT: Timeliness

- System functional correctness depends on the timing of its actions (together with the functionality itself)

- Available solutions
  - real-time data collection, on-the-fly data manipulation, efficient data discovery, and real-time data visualization
  - OpenIoT open source platform (www.openiot.eu)
    - Integration in cloud-based infrastructures
    - Elastic real-time computation
    - High performance real-time online data analysis
Core characteristics of MC-IoT: Timeliness

- System functional correctness depends on the timing of its actions (together with the functionality itself)

- Available solutions
  - real-time data collection, on-the-fly data manipulation, efficient data discovery, and real-time data visualization
  - OpenIoT open source platform (www.openiot.eu)
    - Integration in cloud-based infrastructures
    - Elastic real-time computation
    - High performance real-time online data analysis

- Challenges
  - Self-adaptation and reconfiguration at runtime
  - Hard real-time constraints (not so common, but life-important)
MDE 4 (MC-)IoT

- MDE to tackle MC-IoT challenges for software development and runtime management
MDE 4 (MC-)IoT

- MDE to tackle MC-IoT challenges for software development and runtime management
  - high-level abstraction to address heterogeneity and system complexity
MDE 4 (MC-)IoT

- MDE to tackle MC-IoT challenges for software development and runtime management
  - high-level abstraction to address heterogeneity and system complexity
  - separation of concerns
    - taming complexity
    - collaborative development
    - maintainability
MDE 4 (MC-)IoT

- MDE to tackle MC-IoT challenges for software development and runtime management
  - high-level abstraction to address heterogeneity and system complexity
  - separation of concerns
    - taming complexity
    - collaborative development
    - maintainability
  - automated mechanisms to enable runtime self-adaptation
MDE 4 (MC-)IoT

- MDE to tackle MC-IoT challenges for software development and runtime management
  - high-level abstraction to address heterogeneity and system complexity
  - separation of concerns
    - taming complexity
    - collaborative development
    - maintainability
  - automated mechanisms to enable runtime self-adaptation
  - reusability for sustainable development (time, costs, and effort)
MC-IoT software challenges and MDE solutions

- Heterogeneity
MC-IoT software challenges and MDE solutions

- Heterogeneity

- Large-scale and emergent properties
MC-IoT software challenges and MDE solutions

- Heterogeneity
- Large-scale and emergent properties
- Context awareness and uncertainty
MC-IoT software challenges and MDE solutions

- Heterogeneity
- Large-scale and emergent properties
- Context awareness and uncertainty
- Dynamic discoverability of available resources
MC-IoT software challenges and MDE solutions

- Heterogeneity
- Large-scale and emergent properties
- Context awareness and uncertainty
- Dynamic discoverability of available resources
- Reusability
MC-IoT software challenges and MDE solutions

- Heterogeneity
- Large-scale and emergent properties
- Context awareness and uncertainty
- Dynamic discoverability of available resources
- Reusability
- Security and trust
Heterogeneity – currently

- Heterogeneity in resources, protocols, hardware and software, platforms, programming languages, etc
- Lack of standardized platform-agnostic software solutions
- Cross-platform development intractable
Heterogeneity – currently

- Heterogeneity in resources, protocols, hardware and software, platforms, programming languages, etc
- Lack of standardized platform-agnostic software solutions
- Cross-platform development intractable

Needs
  - platform neutrality
  - standardization of common basic system functions
  - scalability to different platform variants
  - transferability throughout the network
  - integration from multiple suppliers
  - maintainability throughout the system lifecycle
  - software updates and upgrades over the system’s lifetime
Heterogeneity – with MDE

- DSMLs and abstraction allow modelling of heterogeneous aspects using a family of correlatable languages, or even the same language
Heterogeneity – with MDE

- DSMLs and abstraction allow modelling of heterogeneous aspects using a family of correlatable languages, or even the same language
  - Example: the GEMOC approach, [http://gemoc.org](http://gemoc.org)
Heterogeneity – with MDE

- DSMLs and abstraction allow modelling of heterogeneous aspects using a family of correlatable languages, or even the same language
  - Example: the GEMOC approach, http://gemoc.org
  - Plus: metamodelling approach for definition of DSMLs with rich concurrency semantics (unique characteristic of MC-IoT)
Heterogeneity – with MDE

- DSMLs and abstraction allow modelling of heterogeneous aspects using a family of correlatable languages, or even the same language
  - Example: the GEMOC approach, [http://gemoc.org](http://gemoc.org)
  - Plus: metamodelling approach for definition of DSMLs with rich concurrency semantics (unique characteristic of MC-IoT)
  - Minus: rudimentary analysability of relations between logical, physical, and domain time (crucial for timeliness of MC-IoT)
Heterogeneity – with MDE

- DSMLs and abstraction allow modelling of heterogeneous aspects using a family of correlatable languages, or even the same language
  - Example: the GEMOC approach, [http://gemoc.org](http://gemoc.org)
  - Plus: metamodelling approach for definition of DSMLs with rich concurrency semantics (unique characteristic of MC-IoT)
  - Minus: rudimentary analysability of relations between logical, physical, and domain time (crucial for timeliness of MC-IoT)

- Separation of concerns in terms of multi-view modelling for tackling complexity of models with high heterogeneity
Heterogeneity – with MDE

- DSMLs and abstraction allow modelling of heterogeneous aspects using a family of correlatable languages, or even the same language
  - Example: the GEMOC approach, http://gemoc.org
  - Plus: metamodelling approach for definition of DSMLs with rich concurrency semantics (unique characteristic of MC-IoT)
  - Minus: rudimentary analysability of relations between logical, physical, and domain time (crucial for timeliness of MC-IoT)

- Separation of concerns in terms of multi-view modelling for tackling complexity of models with high heterogeneity
  - Plus: Effective way to model view-specific aspects and verify consistency among modelled artefacts
Heterogeneity – with MDE

- DSMLs and abstraction allow modelling of heterogeneous aspects using a family of correlatable languages, or even the same language
  - Example: the GEMOC approach, [http://gemoc.org](http://gemoc.org)
  - Plus: metamodelling approach for definition of DSMLs with rich concurrency semantics (unique characteristic of MC-IoT)
  - Minus: rudimentary analysability of relations between logical, physical, and domain time (crucial for timeliness of MC-IoT)

- Separation of concerns in terms of multi-view modelling for tackling complexity of models with high heterogeneity
  - Plus: Effective way to model view-specific aspects and verify consistency among modelled artefacts
  - Minus: Currently no precise way to verify consistency of implicit assumptions across view-specific models
Heterogeneity – more MDE research needed

- Timeliness should be integral part of DSMLs for (MC-)IoT
- Implicit assumptions across views should be verifiable
- View-specific analysis leverageable at system level to verify global properties
Large-scale and emergent properties – currently

- MC-IoT runtime evolution is very challenging
- Device-specific functionality hard to reallocate to a different device type
Large-scale and emergent properties – currently

- MC-IoT runtime evolution is very challenging
- Device-specific functionality hard to reallocate to a different device type

- Need
  - orchestration mechanisms to suitably control concurrency
Large-scale and emergent properties – with MDE

● Models@runtime¹
  ● models of runtime system and environment to effectively manage complexity of evolving behaviors

Large-scale and emergent properties – with MDE

- **Models@runtime**\(^1\)
  - models of runtime system and environment to effectively manage complexity of evolving behaviors

- **MC-IoT as a specific case of dynamically adaptive systems (DASs)**\(^2\)
  - MDE combined with aspect-oriented techniques to specify and execute DASs

---


\(^2\) B. Morin et al., Models@run.time to Support Dynamic Adaptation, Computer, vol. 42, no. 10, 2009, pp. 44–51.
Large-scale and emergent properties – more MDE research needed

- Dynamic MAPE-K-style (MAPE-K*) self-adaptation in critical conditions
Context awareness and uncertainty – currently

- Ability to adapt to changing context (physical, computational, user-related)
Context awareness and uncertainty – currently

- Ability to adapt to changing context (physical, computational, user-related)
- Context collected from different sources
  - different sources differ in quality (failure prone, can lead to system failure)
Context awareness and uncertainty – currently

- Ability to adapt to changing context (physical, computational, user-related)
- Context collected from different sources
  - different sources differ in quality (failure prone, can lead to system failure)

- Needs
  - manage different context information types and their interplay
  - deduce MC-IoT actions to meet environmental constraints
  - prevent defective and malicious adaptations
  - isolate faults
  - maintain desired security, performance, and dependability despite adaptation
Context awareness and uncertainty – with MDE

- Automatic generation of model alternatives to cope with different context conditions\(^1\) (known uncertainty)

- Identification of functional and non-functional tradeoffs between models (functional and non-functional uncertainty)

- MDE for adaptation of DASs to context changes (known and unknown uncertainty)

Context awareness and uncertainty – more MDE research needed

- Smarter functional and non-functional tradeoff mechanisms
- More powerful, fully automated mechanisms for selecting most suitable configuration
- Improved model-based self-reconfiguration capabilities (e.g. smart models@runtime)
Dynamic discoverability of available resources – currently

- New, unknown, or recovered things can show up any time
- Very little support for efficient discoverability
Dynamic discoverability of available resources – currently

- New, unknown, or recovered things can show up any time
- Little support for efficient discoverability

- Needs
  - Better exploitation of newly available things
  - Dynamic discovery of available resources and constraints
  - Recognition of, communication with, and adjustment of devices
Dynamic discoverability of available resources – with MDE (or rather models..)

- Service-oriented modeling and service-oriented architectures (SOAs)¹
  - models for identifying and specifying services
  - model transformations for realizing, composing, and orchestrating services

¹ A. Arsanjani, “Service-Oriented Modeling and Architecture,” IBM Developer Works, 2004
Dynamic discoverability of available resources – more MDE research needed

- Models for dynamic discoverability
- Models and transformations for realization of new resources and services
Reusability – current situation

- Divergent implementations of same functionality in MC-IoT, such as communication protocols and controllers
- Lack of systematic, disciplined, and quantifiable software engineering methodology
- Lack of comprehensive abstraction mechanisms for handling MC-IoT complexity
- Countless similar not congruent, isolated solutions hard to reuse and combine
Reusability – current situation

- Divergent implementations of same functionality in MC-IoT, such as communication protocols and controllers
- Lack of systematic, disciplined, and quantifiable software engineering methodology
- Lack of comprehensive abstraction mechanisms for handling MC-IoT complexity
- Countless similar not congruent, isolated solutions hard to reuse and combine

- Needs
  - Systematic reusability
  - Sustainable MC-IoT software development
  - Continuous deployment and integration
Reusability – with MDE

- Combination of MDE with component-based software engineering\(^1\)
  - System in terms of reusable and replaceable self-contained model entities

- Model entities integrated through architectures, connectors, and integration patterns

- Model transformations to guarantee runtime preservation of quality attributes in isolation and combination

Reusability – more MDE research needed

- Components cannot be freely used
Reusability – more MDE research needed

- Components can not be freely used
- Robustness hard to achieve due to pervasive criticality and uncertainty
Reusability – more MDE research needed

- Components can not be freely used

- Robustness hard to achieve due to pervasive criticality and uncertainty

- Need for more effective
  - component replication
  - dynamic disassembly
  - reassembly of components with guarantees of functional and non-functional preservation
Security and trust – current situation

- Heterogeneity of communication protocols and APIs exacerbates security issues
- Things communicate on users’ behalf, effective privacy and trust management mechanisms are crucial
Security and trust – current situation

- Heterogeneity of communication protocols and APIs exacerbates security issues
- Things communicate on users’ behalf, effective privacy and trust management mechanisms are crucial

Need
- to deal with data sharing and information transparency
- to take into account policies and enabling user control over personal, location, and movement information
Security and trust – with MDE

- Model-driven security\(^1\)
  - defines system models together with their security requirements
  - uses both to generate system architectures and configured access control infrastructures.

Security and trust – with MDE

● Model-driven security\(^1\)
  ● defines system models together with their security requirements
  ● uses both to generate system architectures and configured access control infrastructures.

● Model-based trust negotiation and management for web services\(^2\)
  ● grant access based on trust established through negotiation

---


Cross-cutting needs

- Continuous synergy among all MC-IoT stakeholders
  - development
  - use
  - proliferation

- Cultural shift regarding how people are connected and share their personal lives and data
The Smart Street Light Case

Smart Street Lampposts
Saving energy and increasing traffic safety by having your own sphere of light

(https://vimeo.com/137837738)
Each lamppost can:
(1) detect the presence of an “object” (car, bike, or pedestrian);

Smart Street Lampposts
Saving energy and increasing traffic safety by having your own sphere of light
(https://vimeo.com/137837738)
Each lamppost can:
(1) detect the presence of an “object” (car, bike, or pedestrian);
(2) compute object’s speed;

Smart Street Lampposts
Saving energy and increasing traffic safety by having your own sphere of light
(https://vimeo.com/137837738)
Each lamppost can:
(1) detect the presence of an “object” (car, bike, or pedestrian);
(2) compute object’s speed;
(3) increase and decrease the brightness of its own lights;

Smart Street Lampposts
Saving energy and increasing traffic safety by having your own sphere of light
(https://vimeo.com/137837738)
Each lamppost can:
(1) detect the presence of an “object” (car, bike, or pedestrian);
(2) compute object’s speed;
(3) increase and decrease the brightness of its own lights;
(4) compute the number of lampposts that should increase the brightness of their yellow light or turn on their red lights;

Smart Street Lampposts
Saving energy and increasing traffic safety by having your own sphere of light
(https://vimeo.com/137837738)
Each lamppost can:
(1) detect the presence of an “object” (car, bike, or pedestrian);
(2) compute object’s speed;
(3) increase and decrease the brightness of its own lights;
(4) compute the number of lampposts that should increase the brightness of their yellow light or turn on their red lights;
(5) send and receive messages to and from neighbor lampposts.

Smart Street Lampposts
Saving energy and increasing traffic safety by having your own sphere of light
(https://vimeo.com/137837738)
The Smart Street Light Case

- *Emergent configuration (EC):* a set of lampposts that temporarily connect and cooperate to form the sphere of light accompanying a road user.
The Smart Street Light Case

- **Emergent configuration (EC):** a set of lampposts that temporarily connect and cooperate to form the sphere of light accompanying a road user.

- **Emergent property:** when a car traveling over the speed limit is approaching another road user the latter gets a heads-up through the red lights.
The Smart Street Light Case

- **Emergent configuration (EC):** a set of lampposts that temporarily connect and cooperate to form the sphere of light accompanying a road user.

- **Emergent property:** when a car traveling over the speed limit is approaching another road user the latter gets a heads-up through the red lights.

- Infrastructures grouped into areas, each with an Area Reference Unit (ARU) providing storage capacity and powerful computation capabilities.
Adapting the system

• ARU initiates (i) a repair procedure and (ii) a system adaptation at runtime
Adapting the system

- ARU initiates (i) a repair procedure and (ii) a system adaptation at runtime
- The ARU (ii) continuously monitors ECs
Adapting the system

- ARU initiates (i) a repair procedure and (ii) a system adaptation at runtime

- The ARU (ii) continuously monitors ECs
  - detects a malfunctioning lampposts and a car traveling towards it
Adapting the system

- ARU initiates (i) a repair procedure and (ii) a system adaptation at runtime

- The ARU (ii) continuously monitors ECs
  - detects a malfunctioning lampposts and a car traveling towards it
  - detects car approaching too fast through lamppost sensor
Adapting the system

- ARU initiates (i) a repair procedure and (ii) a system adaptation at runtime
- The ARU (ii) continuously monitors ECs
  - detects a malfunctioning lampposts and a car traveling towards it
  - detects car approaching too fast through lamppost sensor
  - warns cars through navigation system (available resource) using similar graphics and/or acoustics as lamppost would in working conditions
Abstract (MC-)IoT system
Our idea: MDE4IoT

- Modelling languages
- VP₁
  - SW₁
  - HW₁
- VP₂
  - SW₂
  - HW₂
- VP₃
  - SW₃
  - HW₁

- Monitoring
- Execution

SWᵢ = software comp.
HWᵢ = hardware comp.
VPᵢ = design viewpoint
• = allocation
/> = automation
= analysis - planning

Running executable artefacts

(-cell phone, car, IoT)
Our idea: MDE4IoT

- Design
- Deploy
- Run
- Evolve

SW_1 = software comp.
HW_1 = hardware comp.
VP = design viewpoint
\Rightarrow = allocation
\Rightarrow = automation
\leftarrow = analysis - planning
Our idea: MDE4IoT

- Design
- Deploy
- Run
- Evolve
- Adapt
  - affected executable artefacts directly re-deployed
  - re-allocation of functionality at modelling level and re-deployment
Applying MDE4IoT

- (f)UML, ALF, MARTE, design time, run time

```java
while(!this.toWarn)
    if(this.act1.warn() == true)
        this.isWarned = true;
```
Summary
MDE can give a great hand to (MC-)IoT!
MDE can give a great hand to (MC-)IoT!

Especially software for (MC-)IoT
Challenges for MDE 4 (MC-)IoT

- Heterogeneity
  - Timeliness part of DSMLs
  - Verifiable implicit assumptions across views
  - View-specific analysis leverageable at system level to verify system’s global properties
Challenges for MDE 4 (MC-)IoT

- Heterogeneity
  - Timeliness part of DSMLs
  - Verifiable implicit assumptions across views
  - View-specific analysis leverageable at system level to verify system’s global properties

- Large scale and emerging properties (configurations)
  - Dynamic MAPE-K-style (MAPE-K*) self-adaptation in critical conditions
**Challenges for MDE 4 (MC-)IoT**

- **Heterogeneity**
  - Timeliness part of DSMLs
  - Verifiable implicit assumptions across views
  - View-specific analysis leverageable at system level to verify system’s global properties

- **Large scale and emerging properties (configurations)**
  - Dynamic MAPE-K-style (MAPE-K*) self-adaptation in critical conditions

- **Context awareness and uncertainty**
  - More effective mechanisms for smarter functional and non-functional tradeoffs
  - More powerful, fully automated mechanisms for selecting most suitable configuration
  - Improved model-based self-reconfiguration capabilities
Challenges for MDE 4 (MC-)IoT

- Heterogeneity
  - Timeliness part of DSMLs
  - Verifiable implicit assumptions across views
  - View-specific analysis leverageable at system level to verify system’s global properties

- Large scale and emerging properties (configurations)
  - Dynamic MAPE-K-style (MAPE-K*) self-adaptation in critical conditions

- Context awareness and uncertainty
  - More effective mechanisms for smarter functional and non-functional tradeoffs
  - More powerful, fully automated mechanisms for selecting most suitable configuration
  - Improved model-based self-reconfiguration capabilities

- Reusability
  - Need of more effective mechanisms
  - Component replication
  - Dynamic disassembly/reassembly of components with guarantees of functional and non-functional preservation


Thanks for the attention!