# THE NEXT EVOLUTION OF MDE: a Seamless Integration of Machine Learning into Domain Modeling

Thomas Hartmann*, Assaad Moawad+, Fouquet François* and Yves le Traon*

(*) Interdisciplinary Centre for Security, Reliability and Trust

University of Luxembourg

(+) DataThings S.A.R.L.

Luxembourg

# Some words about us...

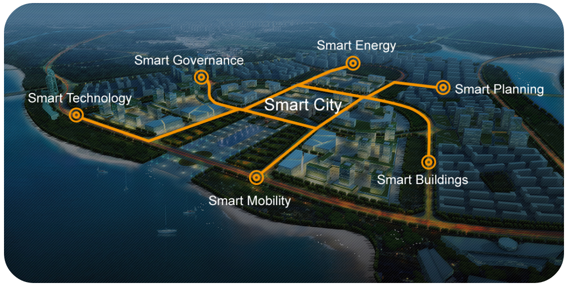
- research conducted *in Luxembourg* focused on Computer Science
- working *in close* collaboration with *industrial partners*
  - Smart grid
  - Metal industry
  - Industrie 4.0
  - *Transportation systems*
  - *Bank, trading and wealth management*

- ultimate goal: near-real-time analytics
  - designing tools to ease operational decision-making
- joined work with start-up DataThings
  - specialized in custom data analytics

DATATHINGS

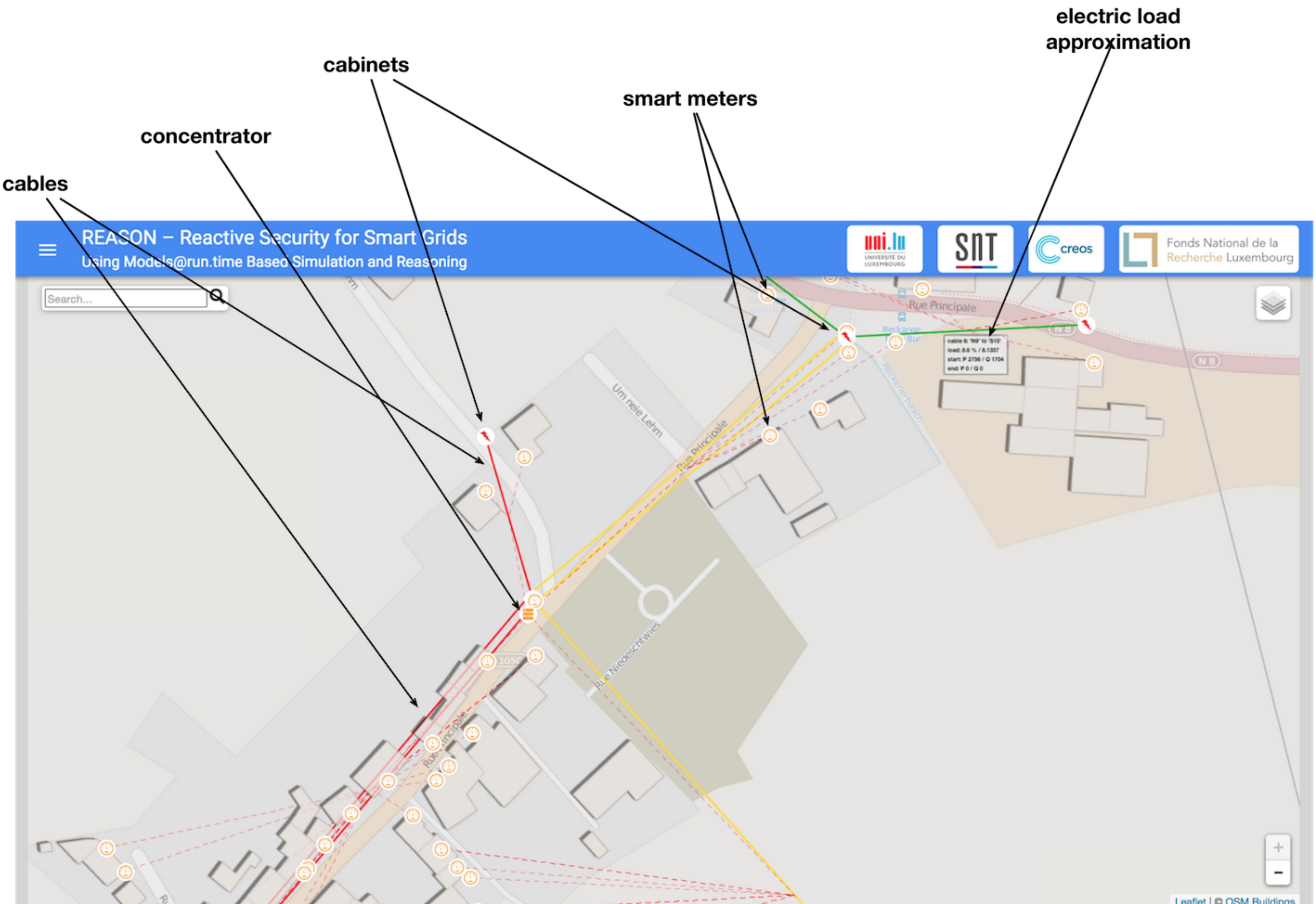
# The next generation of smart cyber-physical systems

> *CPS interact with their environments via sensors and actuators, and monitor and control physical processes, using feedback loops, where physical processes and computations affect each other (E. A. Lee)*





*...has the potential to lay the foundations for our critical infrastructures of tomorrow*


electric load
approximation
cabinets
smart meters
concentrator
cables
REASON – Reactive Security for Smart Grids
Using Models@run.time Based Simulation and Reasoning
creos
Fonds National de la
Recherche Luxembourg
Search...
Rue Principale
Um neie Lehm
Rue Principale
Rue Niedeschtwies
Leaflet | © OSM Buildings

# But what kind of analytics is needed?

**descriptive**

- mining historical data
- statistics
- data aggregation
- averaging and summarizing
- number of posts, likes, checkins, …

**what happened?**
**why did it happen?**

**predictive**

- statistical forecasting
- predictive modeling
- machine learning

**what might happen?**

**prescriptive**

- $\boxed{\text{what-if}}$
- simulation
- decision modeling
- optimization

**what should we do?**

# Reactive Models@Run.time can handle such mix

**model-driven engineering**

**purpose**: **domain definition**
meta-model: **defined as** EMF/Ecore, UML, DSL, textual/graphical formalism, …

**structure + behavior**

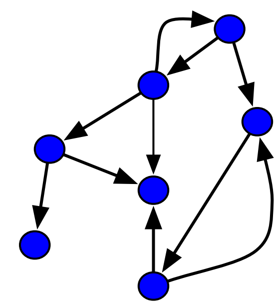
**arbitrary number of transformation steps**

**model**

**models@ run.time**

**purpose: runtime usage**
*represent the context of CPSs during runtime to reason about a systems state*

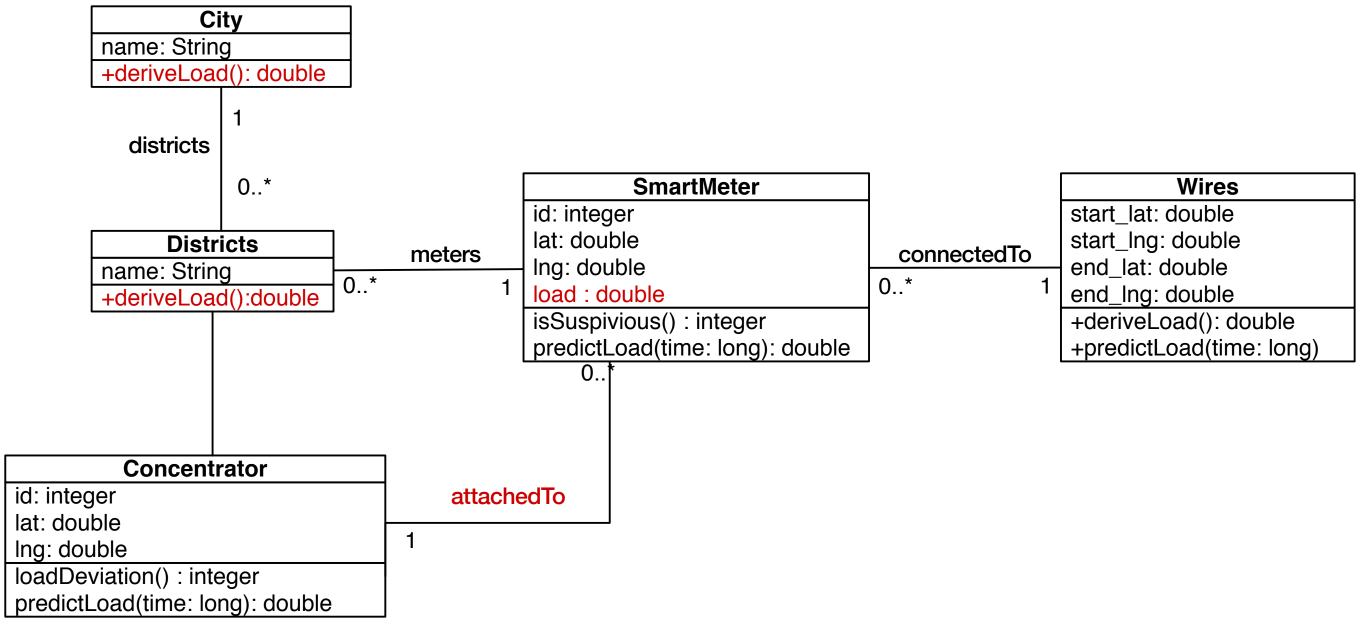
**models@run.time / object graph**



> *behavior modeling defines all domain known simulation and prediction functions ( e.g. Kirchoff laws, ohm laws..) through code or sub-models.*

> *related to: GEMOC initiative, executable modeling, model-based simulation…*

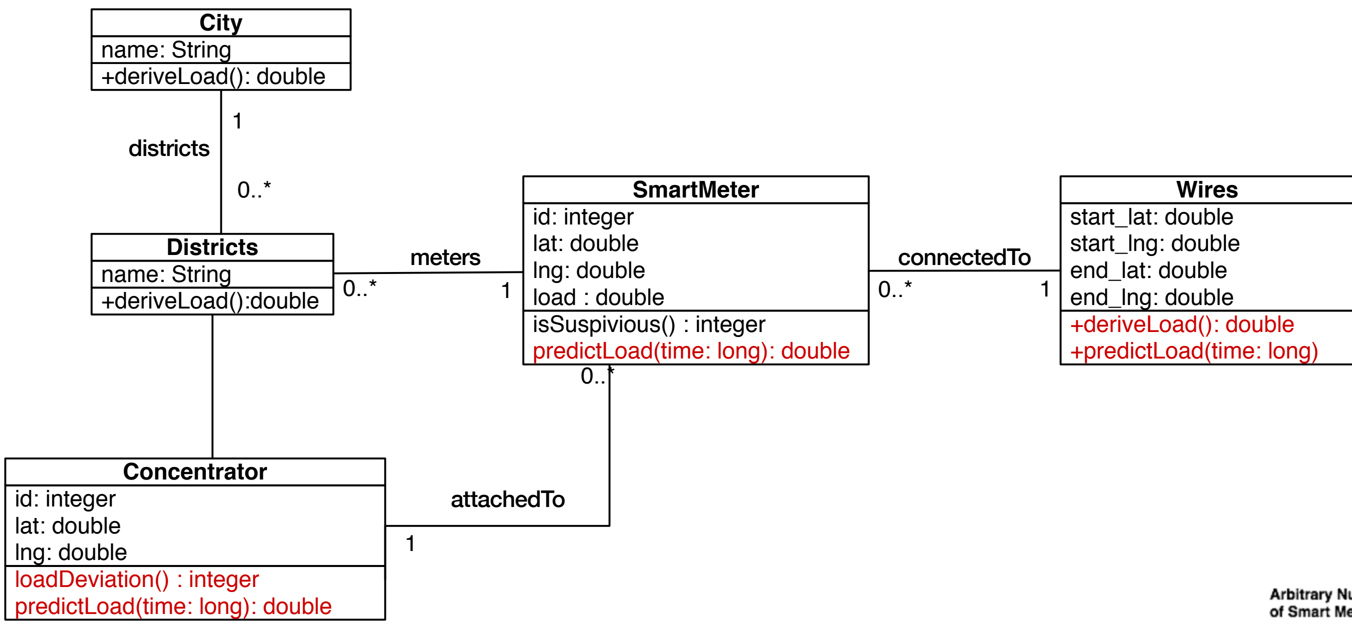
# Smart grid data are in motion



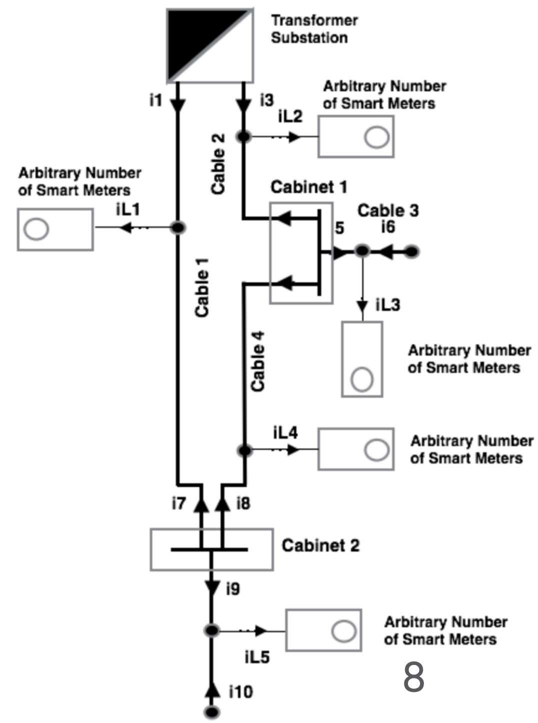

> *Red elements have different values over time (like a time-series). In a nutshell, our model defines all its entities as a function of time. More details in our MODELS'14 and SEKE'17 papers.*

> *readProperty($ID_{elem}$, $TP_{timepoint}$) => {$Attrs_{ID,TP}$, $Rels_{ID,TP}$}*

# Behavior model potentially contains known-unknowns

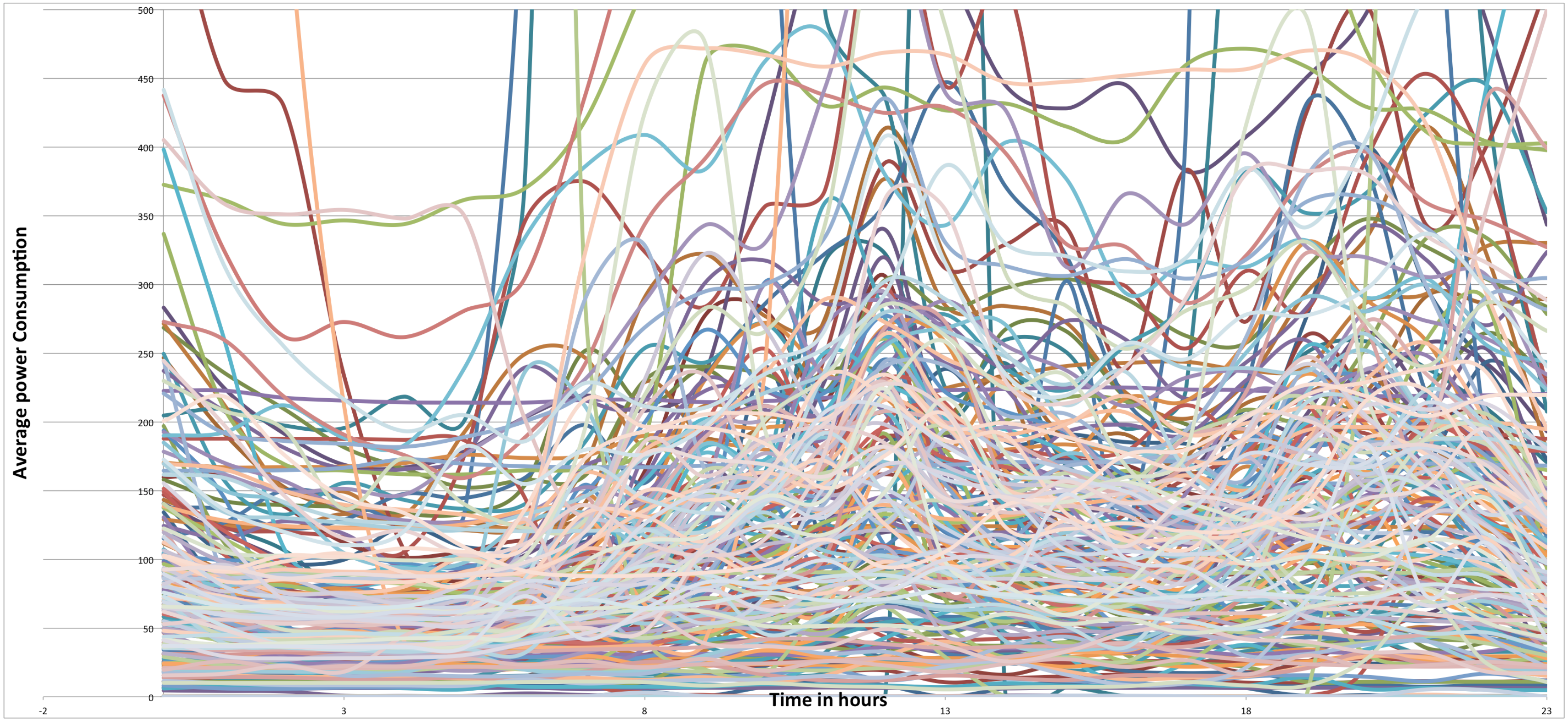



> *Wire load prediction is a well-known problem but relies on meters' electric consumption (Wh) which are obviously not known at design time.*
> *The same applies for suspicious value detection…*



# What is normal? What is a fraud?

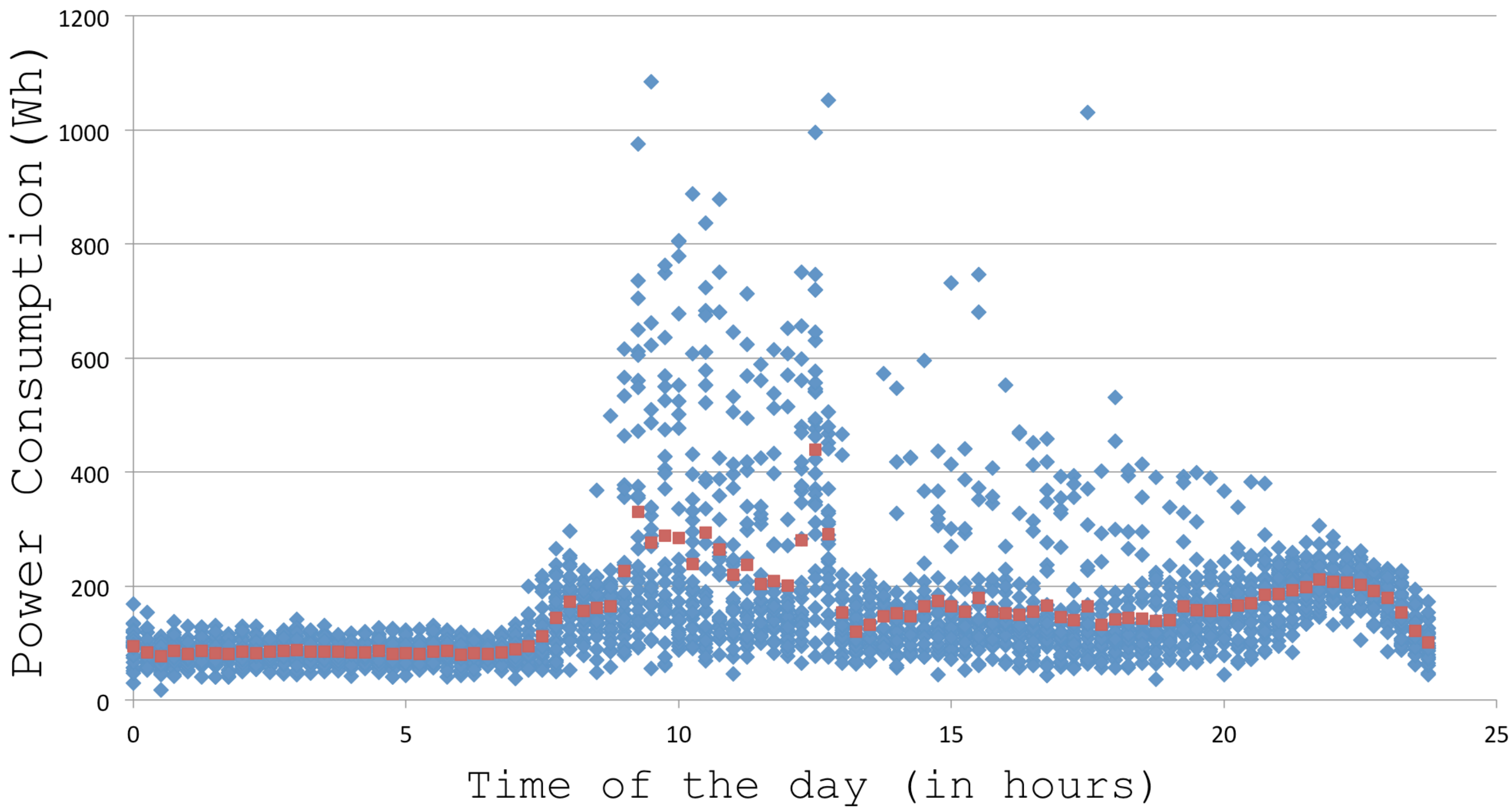
> *Here are ~200 electric consumption curves of a district in Luxembourg.*
> *Where are issues? Is there a common function?*




> *Not obvious, even for a deep learning algorithm*

# Each can be turned independently into a probability space

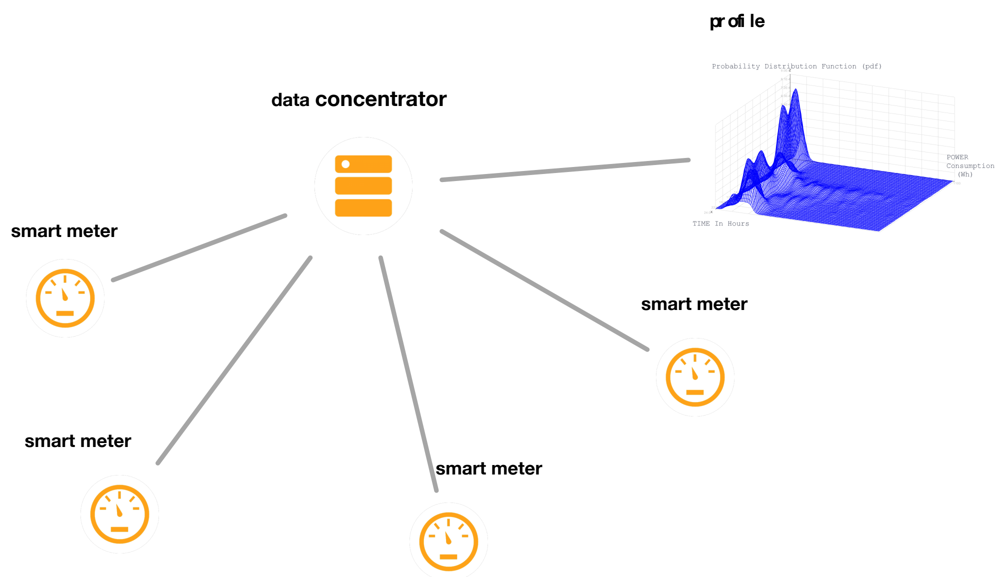
> *Per customer, we can build a daily and/or weekly consumption profile using machine learning*



# The need for flexible micro-learning

- Coarse-grained can just learn statistics at the concentrator level
  - Problem: *connections from meters to concentrators vary*
  - *Network protocols can logically reconfigure grids hourly!*
- a concentrator profiler depends on connected meter profiles
  - *how can we model dependencies between learned/derived?*

➔ (Micro) learning units can be composed together, on-demand



# Motivation and needs

- Many things are known at design time
  - *e.g., mathematical and physical models, domain knowledge, …*
- However, some *information* can only be learned from *live data*
  - *e.g., consumption behavior of customers, failure rates, …*
  - Often, what can be learned is known at design time by *experts*

➔ *stands in relation to domain knowledge*

- WARNING: Machine Learning only reflect past values!
  - Who wants to measure a black-out to prevent it?
  - Instead of *pure learning*, initial function are needed

➔ *how to express, i.e., model what can/should be learned?*

# Weaving learning into domain modeling: requirements

> *R1: modeling learning together with and at the same level than domain data*

> *R2: ⚠ learning should be encapsulated into independently computable small learning units: -> micro learning*

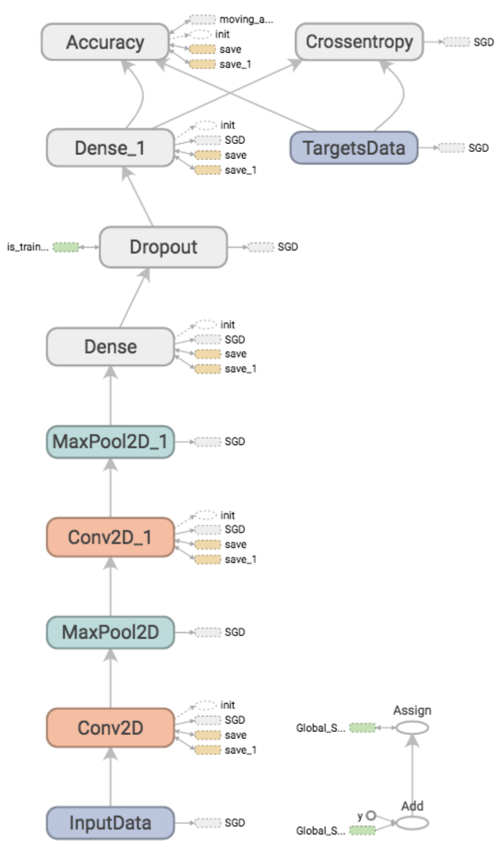
> *R3: learning units, domain data, derived attributes can be mixed and chained*

> *R4: automated mapping from the domain representation to the internal mathematical representation required by ML algorithms*

> *R5: learning must be updated in live (e.g. incremental learning)*

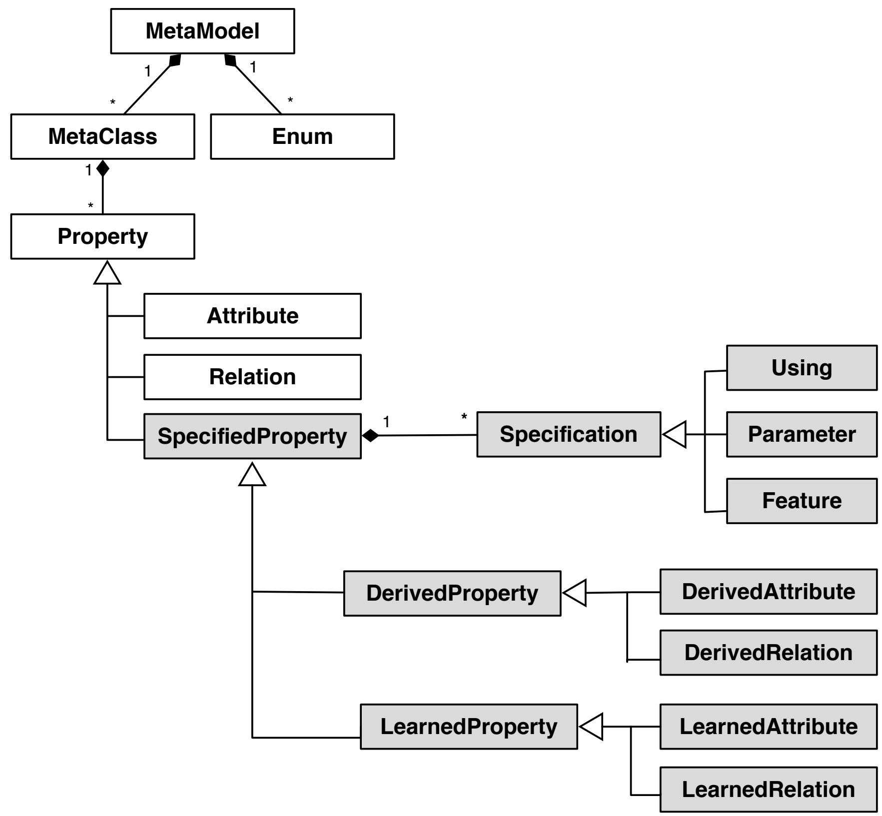
# Modeling Learning != Modeling with Learning

> *Abstractions are required to ease the learning algorithm development. They mostly leverage procedure-like flows such as TensorFlow model (below). Despite it could be in future complementary, we only wrap learning units using their contract (input/output).*



# Proposition

# Meta Meta Model *(MOF-like)* extension




> *Features act as **extractors**, virtually a relationship to other properties...*

# Extended Meta-Model Textual Syntax

> *We use a classic base such as TextCore, EMFFacade, KM3, Kermeta...*

```
metaModel ::= (class | enum)

enum ::= 'enum' ID '{' ID (',' ID)* '}'

class ::= 'class' ID ('extends' ID (',' ID)*)? '{' property* '}'

property ::= annot* ( 'att' | 'rel' | 'ref' ) ID : ID spec?
```

> *We extend it with learning/deriving behavior definition, STRING=expression*

```
annot ::= ( 'learned' | 'derived' | 'global')

spec ::= '{' (feature| using | param)* '}'

param ::= 'with' ID ( STRING | NUMBER )

feature ::= 'from' STRING

using ::= 'using' STRING /* NeuralNetwork, GaussianMixture, Bayesian*, DecisionTree... */
```

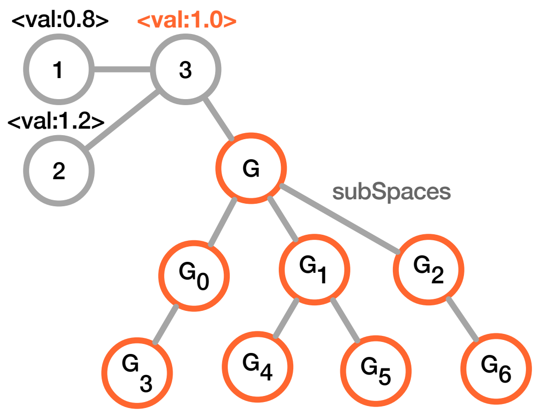
# Modeling Learning Patterns *(1/3)*

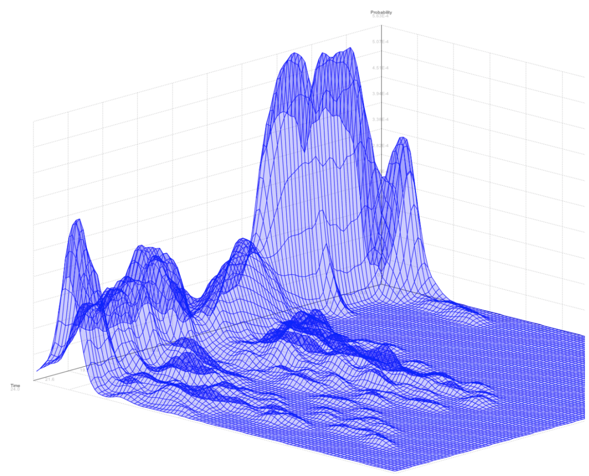
> *Embeded, micro-learned classifier*

```
class SmartMeter {
    att activeEnergy: Double
    att reactiveEnergy: Double
    rel customer : Customer
    learned att anomaly: Boolean {
        from "activeEnergy"
        from "reactiveEnergy"
        using "GaussianClassifier"
    }
}
```

# Modeling Learning Patterns *(2/3)*

```
class SmartMeterProfile {
    rel meter : SmartMeter
    @timeSensitivity "{{daily}}"
    from "HOURS(meter.time)" //round time in hour
    from "meter.activeEnergyConsumed" //+ type of day, temporatures...
    using "GaussianMixture"
}
class SmartMeter { rel profiles : SmartMeterProfile }
```



# Modeling Learning Patterns *(3/3)*

> *Derived attributes and learned attributes can be mixed (both ways)*

```
class Concentrator {
  rel connectedMeters : SmartMeters
  ref profile : ConcentratorProfile
}
class ConcentratorProfile {
  ref host : Concentrator
  derived att powerProbabilities : Double[] {
    from "host.connectedMeters.profile"
    using "aggregator" }
}
```

> *Similarly **Recommendation Systems** can be build (full example in the paper)*

# Experimental validation

# Experimental goal and setup

> *Integrating Machine Learning within MDE tools eases manipulation of dozen of learning units. How effective/efficient are they?*

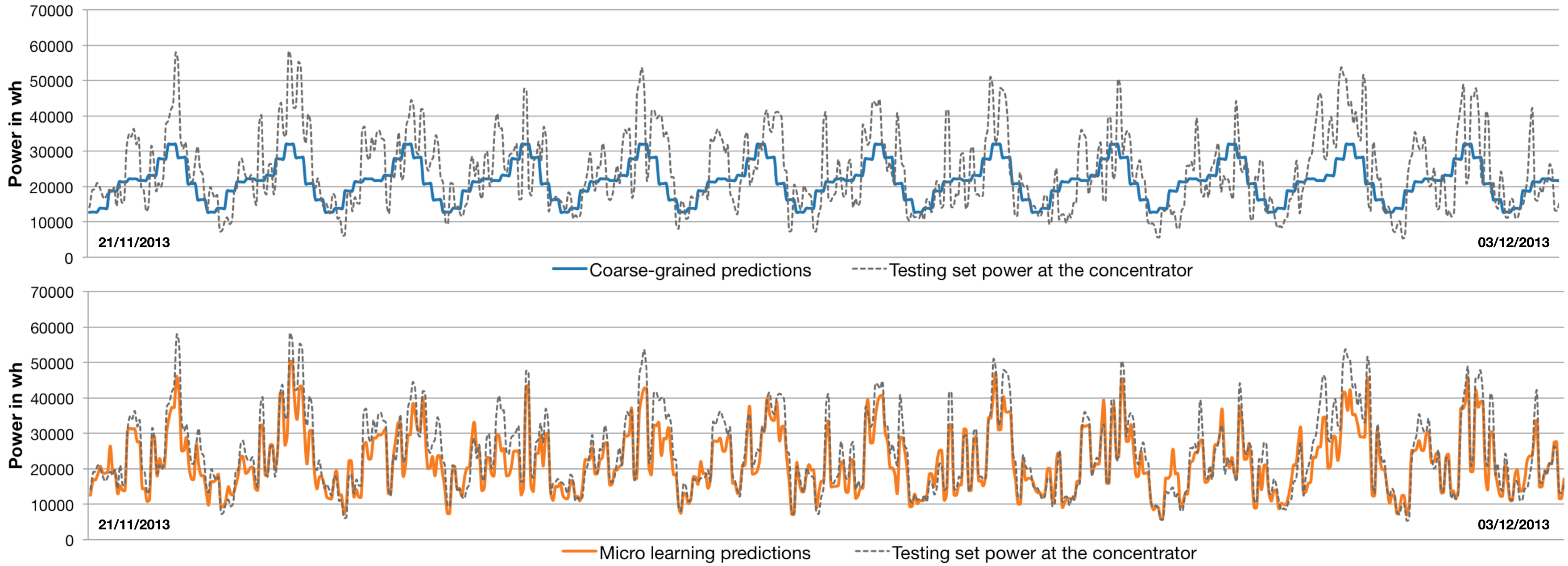
- are micro learning units *more accurate* than *coarse-grained ones?*
- are such extended models fast enough to be used for *live analytics*?

## Target system

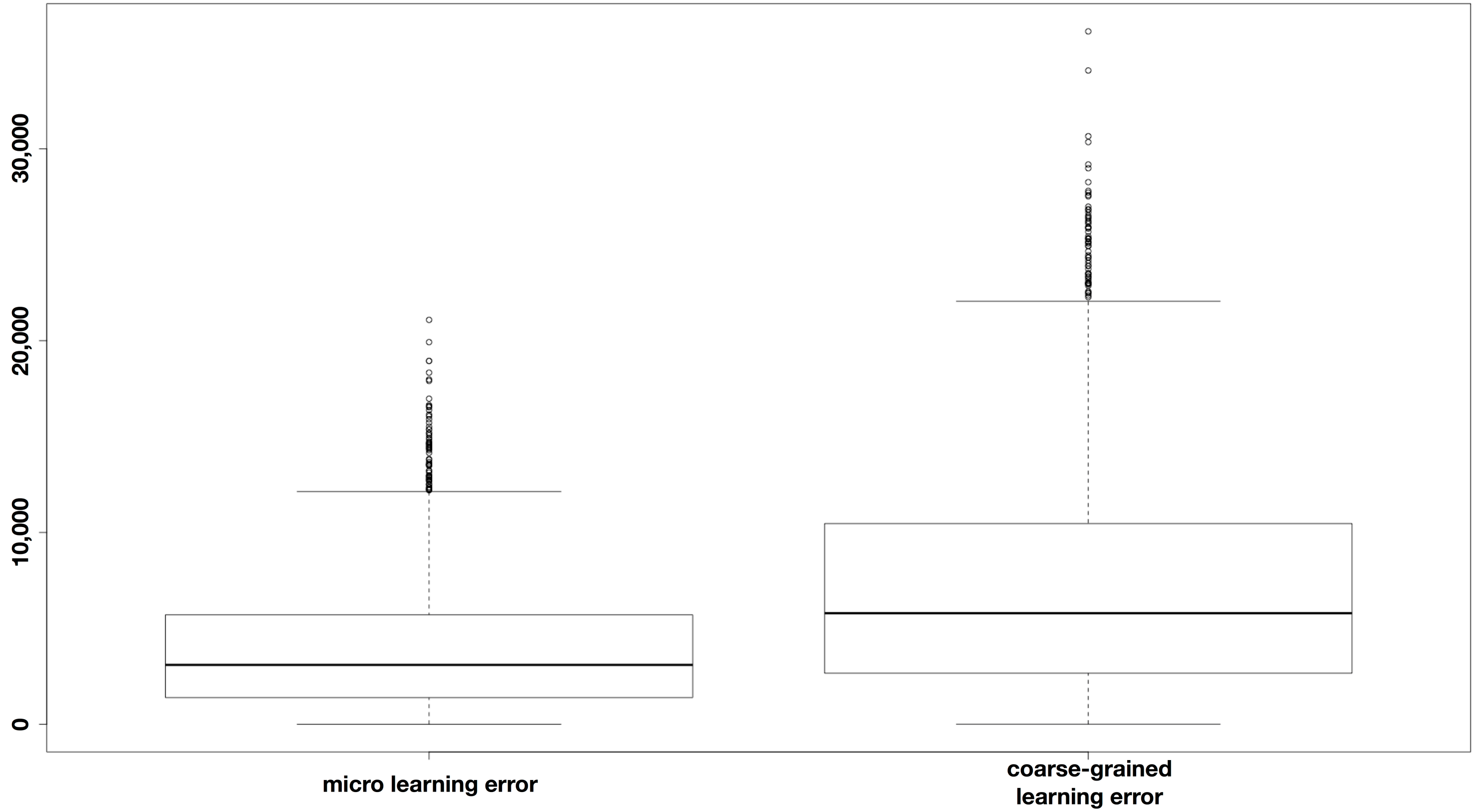
- predicting customers' electric consumption behavior
- *2 concentrators* and *300 smart meters*
- *7,131,766* power records *(6,389,194 for training, 742,572 for testing)*
- every hour, randomly change the *smart meter connections*
  - *Each concentrator has between 50 and 200 connected meters*
- the same algorithms are used for *coarse-grained* and *microlearning*

# Predicting concentrator electric load

> *Implemented as a derived attribute leveraging connected meters relationships and learned attributes. Micro-learning (orange) clearly outperforms coarse-grained learning, confirming the benefits to mix topology and learning units.*

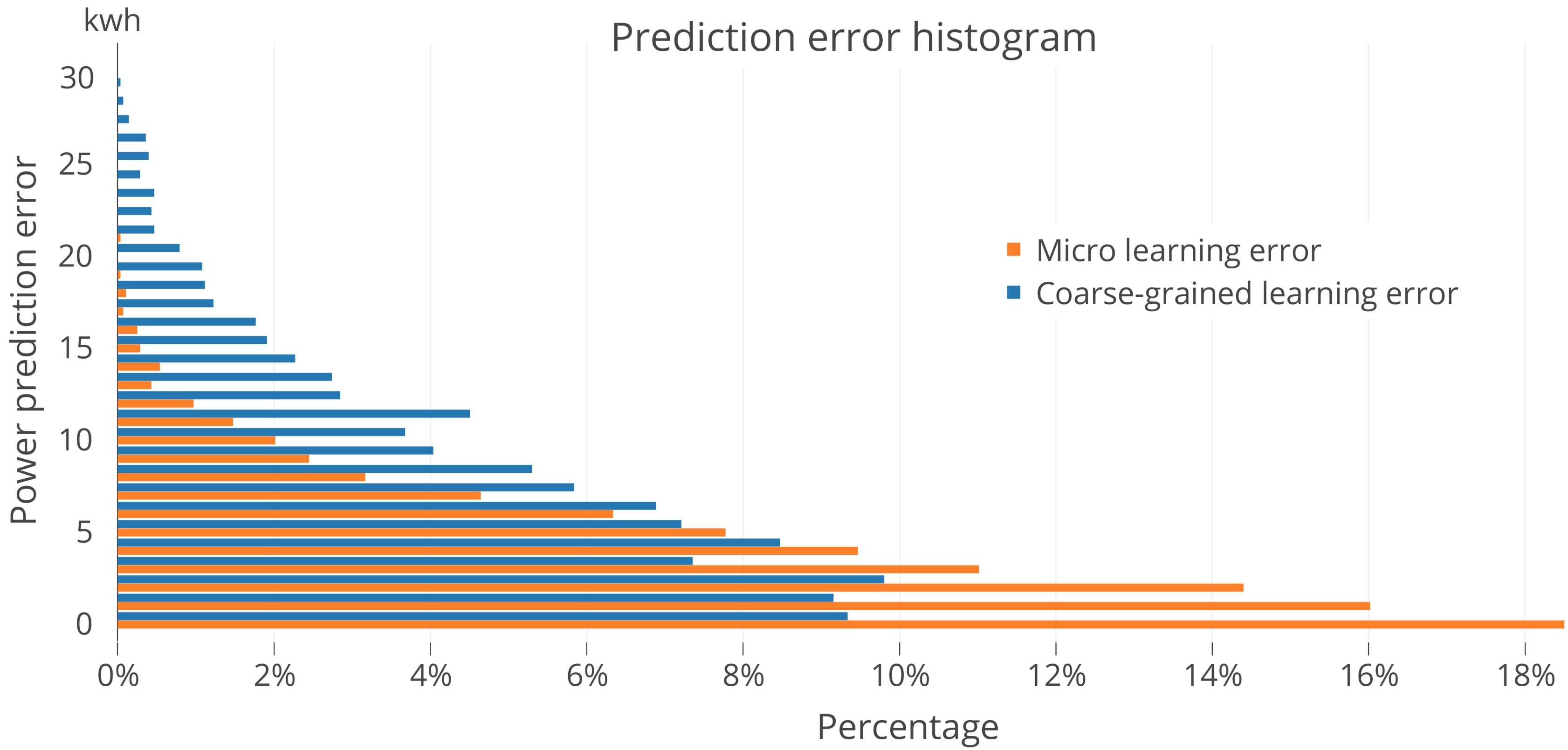


# About predictions effectivness *(1/2)*




> *The overall prediction divergence is highly improved!*

# About predictions effectivness *(2/2)*




> *Micro-learning especially reduces major power prediction mistakes*

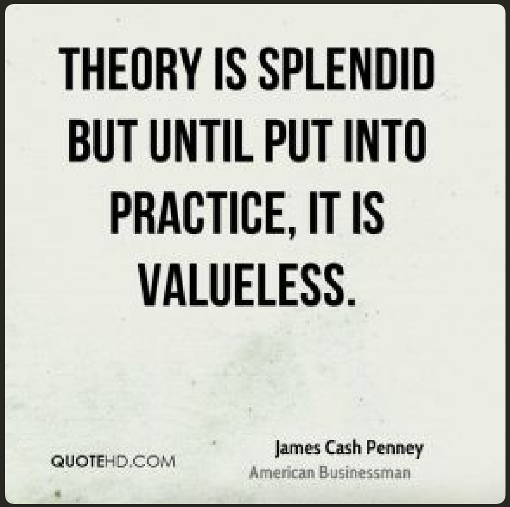
# About such *reactive* model efficiency

| Users | Records | Loading in sec | Profiling in sec |
|---|---|---|---|
| 10 | 283,115 | 4.28 | 1.36 |
| 50 | 1,763,332 | 21.94 | 7.20 |
| 100 | 3,652,549 | 44.80 | 14.44 |
| 500 | 17,637,808 | 213.80 | 67.12 |
| 1000 | 33,367,665 | 414.82 | 128.53 |
| 5000 | 149,505,358 | 1927.21 | 654.61 |

- *roughly linear with the number of records loaded: O(n)*
- *around 60,000 values per second on a single processor*
- *predicting: 1,000 values per second (sum all smart meter profiles)*
- fast enough for live usage!

> *Update: GreyCat now uses ND-Trees leading to 2,000,000 v/s learning speed*

# Production feedback...



# *Declarative* but rich extractors

- Extractors were initially designed as *math expressions*
    - `from "(this.meter.reactiveEnergy + this.activeEnergy) / 2"`
- Learning algorithms need more powerful preprocessor
    - introducing tasks such as `from task_exp`

```
task_exp ::= action_exp ('.' action_exp)

action_exp ::= ID '(' param ( ',' param)* ')' //action ID comes from associated plugins

param ::= (STRING, NUMBER, subTask)

subTask ::= '{' task_exp '}'
```

> *action library contains lambda-like operators, forEach, travelInTime...*

```
from linearReg( { travelInTime("${now} - 2*${hour}").reactiveEnergy },{ this.reactiveEnergy } )
```

# When to learn? Ensuring consistency

- Learning units can be updated through a call to `.learn()`
    - can it be automatic? *(ensuring strong consistency)*
- At call momentum, extractors are executed emitting features shot
    - if P depends on A and B, both should be updated
    - otherwise learning will dramatically diverge *(e.g. classifier)*
- We need learning transactions to update the model
    - similarly to DB storages then enforce consistent points
    - example of Java API:

```
Transaction ml_t = model.newTransaction();
a.setValue(1234); //this will notify p, but learning is in pending
b.setValue(1234);
ml_t.close(); //actually call learn methods on all affected profiles
```

# What's next? Towards Meta-Learning

> *Meta-Learning is about learning optimal parameters of the learning class against a specific problem*

- Currently, we request *complete* modeling of hypothesis about data
- Usually, params are open, often configure using empirical runs

```
class SmartMeterProfile {
    using ("GaussianFixTree | GMM | NeuralNetwork")
    from "parent.activeEnergy"
    @timeSensitivity(" [1:24] {{weeks}} ")
    with learningRate (0.001 | 0.003)
}
```

> *Using extended MetaModel, reflexive exploration can find optimal params*

# Conclusion and *take away* slide

- Analytics requires *heterogenous* and *independents* learning units
  - micro-learning *(profiling)*, taste vectors *(recommendation)*
- Domain modeling and Machine Learning can be greatly combined
  - chaining learning and derived functions
  - MDE at the rescue to *update all learning units even in-live*
- This Paves the way for *prescriptive analytics* & in-depth exploration
- GreyCat is open source! https://github.com/datathings/greycat

Thank You !

Any questions ?

UNIVERSITÉ DU LUXEMBOURG

SnT securityandtrust.lu

Serval securityandtrust.lu