POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria
Scuola di Ingegneria Industriale e dell'Informazione

MARC

Model and Analysis of Resource Consumption

NECSTLab
Novel, Emerging Computing System Technologies Laboratory
presso il Politecnico di Milano

SWARM Lab
at University of California, Berkeley

Relatore: Prof. Marco Domenico SANTAMBROGIO
Correlatore: Dott. Matteo FERRONI

Tesi di Laurea di:
Andrea Corna, matricola 816737
Andrea Damiani, matricola 804052

Anno Accademico 2014-2015

Appendix D

User Manual

This chapter will describe how to build a MARC XML configuration file.

The XML configuration file gives the user the possibility to define how data are organized and set all the parameters for each computation phase inside MARC. The XML file must be validated with the XSD provided by the following link `http://mpower.necst.it/marc/configuration.xsd`.

D.1 Configurations

The root of the XML configuration file is the `<configurations>` tag. Inside this tag the user can define all the parameters to tune MARC computation phases. The user must define:

- **job**: the **name** and the username of the **owner** of the current computation request.
- **dataset**: the name of the dataset uploaded to MARC infrastructure on which the user wants to perform the job. The name must be the same of the file name uploaded to MARC, without the extension.
- **computation-unit**: a string identifier defined by the user to tag a specific computation over the dataset define above. At every change in the XML configuration file, the user must specify a new **computation unit** identifier; failing to do so will allow MARC infrastructure to use previous cached results when available, ignoring the new configuration settings.
- **phases**: configurations for all MARC computation phases. In particular there are 6 different phases: **phase0**, **phase1**, **phase2a**, **phase2b**, **phase2c** and **phase3**. Each phase configuration will be described in the next subsections of this manual.

Listing D.1: Configurations example

```
<configurations xmlns="http://mpower.necst.it/marc/configuration">
  <job>
    <name>JOB_NAME</name>
    <owner>JOB_OWNER</owner>
  </job>
  <computation-unit>COMPUTATION_STRING_IDENTIFIER</computation-unit>
  <dataset>DATASET_NAME</dataset>
  <phases>
    <!--phases configurations-->
  </phases>
</configurations>
```

D.1.1 Phase 0

In this section it is possible to define the **mappings**, i.e. a detailed description of each column of the dataset. All mappings are collected in the `<features>` tag, using the `<feature>` sub-tag. This is organized with the following schema:

- **name**: the identifier of the feature.
- **col-index**: the index of the column index corresponding to the feature in the dataset. The columns in the dataset are indexed progressively starting from 0.
- **type**: the type of data. There are three possible contents:
  - **categorical**: this value represents a feature represented by discrete distinct values, each of which with a specific associated meaning (e.g. in the mobile device environment, a feature indicating the currently connected GSM network type - 1 means GSM, 2 means GPRS, 3 means UMTS, etc.). Categorical representations are not required to be continuous – i.e. their domain can be 1, 2, 3, ... as well as 10, 100, 1000, ... – since MARC will account for only those which occur in the dataset.
  - **instantaneous**: this value represents a feature depending only on the current time instant (e.g. the speed of a car). The difference with respect to **categorical** is that, in this case, the feature values must reflect a *physical* phenomenon and so are supposed to range on a continuous domain.

  - **cumulative**: this value represents a feature accounting for accumulated effects on a continuous domain (e.g. the distance traveled by a car, the bytes transferred by a network interface).

- **monotony** *(OPTIONAL)*: the monotony of the feature data. There are two possible values: **ascendant** if the feature value always increases during time, otherwise **descendant** (in both cases constant values are allowed).

- **bounds** *(OPTIONAL)*: defines a feature whose values are limited on a specific range. The user can specify **lower** and **upper** bounds, in order to define both the lower and the upper bound values. It is possible to specify only one of the two.

- **reduction** *(OPTIONAL)*: allows to change the feature's granularity, converting a continuous feature into a discrete one. We provide three different granularity reductions: **linear**, **logarithmic** and **fixed**. For the first two options it is necessary to indicate **min** (the lower bound), **max** (the upper bound) and **splits** (the number of discrete values in output). As for the **fixed** reduction, the user must specify a list of **border**s, that represent the lower and upper boundaries of each continuous sub-domain associated to a discrete output value. The minimum number of borders is two, resulting in a single discrete value. (e.g. the screen brightness of a device, ranging from 0 to 255, can be reduced from 0 to 3 by applying a linear reduction setting 0 as min, 255 as max and 3 as splits; the intensity of a recorded sound, ranging from $-\infty$ to 30dB, can be reduced from 0 to 5 by applying a logarithmic reduction setting the lower interesting level as min, 30 as max and 6 as splits)

The last parameter of this phase is **timestamp-feature**, that defines the identifier of the feature representing time in the dataset.

Listing D.2: Phase0 configuration example

```
<phase0>
  <mappings>
    <features>
      <!--minimal feature definition example-->
      <feature>
        <name>feature_1</name>
        <col-index>0</col-index>
        <type>cumulative</type>
      </feature>
```

```
   <!--a feature without upper bound example-->
   <feature>
     <name>feature_2</name>
     <col-index>1</col-index>
     <type>instantaneous</type>
     <monotony>ascendant</monotony>
     <bounds>
        <lower>30</lower>
      </bounds>
   </feature>

<!--a linearly reduced feature example-->
<feature>
   <name>feature_3</name>
   <!-- column index are not required to be ordered or continuous
       -->
   <col-index>3</col-index>
   <type>categorical</type>
   <bounds>
      <lower>0</lower>
        <upper>4</upper>
      </bounds>
      <reduction>
        <linear>
             <min>-31.73</min>
             <!--number formats are compliant with Java double
                 toString()-->
             <max>+73.4e10</max>
             <splits>5</splits>
        </linear>
      </reduction>
   </feature>

   <!--a fixed reduction application example-->
   <!-- 0: [-30,18)
        1: [18,51)
        2: [51,107] -->
 <feature>
     <name>feature_4</name>
     <col-index>4</col-index>
     <type>instantaneous</type>
     <bounds>
       <lower>0</lower>
       <upper>2</upper>
```

```
        </bounds>
        <reduction>
          <fixed>
            <border>-30</border>
            <border>18</border>
            <border>51</border>
            <border>107</border>
          </fixed>
        </reduction>
      </feature>
    </features>
  </mappings>
  <timestamp-feature ref="feature_1"/>
</phase0>
```

D.1.2 Phase 1

In this section it is possible to decide which data conditioning operations will be applied to prepare data for modeling computations.

Pre-processing

The first type of operation is *coherence correction*. This operation is OPTIONAL and its tag can be safely omitted. This operation is defined using rules and is scoped on each single sample. The user has to add a **coherence** tag to specify the **condition**, i.e. the data characteristics that activate the rule, and the **corrections**. In the latter it is possible to specify more than one **actuation**, representing the action to perform over the dataset; at least one actuation is required. For each sample, all the rules are applied once in order of appearance in the configuration file. As soon as a condition evaluates `true`, its related corrections are applied to the sample, before continuing with the following rules. In the end, every sample is checked again against every rules and if not coherent, i.e. if any of the rules evaluates `true`, the sample will be discarded since incoherent.[1]

Listing D.3: Coherence condition grammar

```
<FORMULA> :=   <TERM> |
                      <TERM>||<FORMULA>
<TERM>      :=    <FACTOR> |
```

[1]Note that the granularity reduction configured in subsection D.1.1 will be applied only after the sample is validated as coherent and within the bounds also specified in subsection D.1.1

```
                    <FACTOR>&&<TERM>
<FACTOR>    :=    <DIRECT-EXPRESSION> |
                    <NEG-EXPRESSION>
<DIRECT-EXPRESSION>  :=   <EXPRESSION> |
                                (<FORMULA>)
<NEG-EXPRESSION> :=  <EXPRESSION> |
                         !(<FORMULA>)
<EXPRESSION> := <MEMBER>=<MEMBER> |
                    <MEMBER>!=<MEMBER> |
                    <MEMBER>>=<MEMBER> |
                    <MEMBER><=<MEMBER> |
                    <MEMBER>><MEMBER> |
                    <MEMBER><<MEMBER>
<MEMBER>    : =     <M-TERM> |
                    <M-TERM> + <MEMBER> |
                    <M-TERM> - <MEMBER>
<M-TERM>    :=    <M-FACTOR> |
                    <M-FACTOR> * <M-TERM> |
                    <M-FACTOR> / <M-TERM>
<M-FACTOR>  :=    |<MEMBER>| | //Absolute value
                    (<MEMBER>) |
                    [LOG](<NUMBER>,<MEMBER>) |
                    [POW](<NUMBER>,<MEMBER>) |
                    -<MEMBER> |
                    <POINTER> |
                    <NUMBER>
<POINTER>      := [a-zA-Z_]\w*
<NUMBER>    := (0|[1-9][0-9]*([.][0-9]+)?)
```

Listing D.4: Coherence correction grammar

```
<RULE>      :=      <POINTER>=<MEMBER>
<MEMBER>    : =     <M-TERM> |
                    <M-TERM> + <MEMBER> |
                    <M-TERM> - <MEMBER>
<M-TERM>    :=    <M-FACTOR> |
                    <M-FACTOR> * <M-TERM> |
                    <M-FACTOR> / <M-TERM>
<M-FACTOR> :=   |<MEMBER>| | //Absolute value
                    (<MEMBER>) |
                    [LOG](<NUMBER>,<MEMBER>) |
                    [POW](<NUMBER>,<MEMBER>) |
                    [MIN](<MEMBER>,<MEMBER-LIST>) |
                    [MIN](<MEMBER>,<MEMBER-LIST>) |
                    -<MEMBER> |
```

```
                    <POINTER> |
                    <NUMBER>
<MEMBER-LIST>  :=   <MEMBER> | <MEMBER>,<MEMBER-LIST>
<POINTER>      := [a-zA-Z_]\w*
<NUMBER>    := (0|[1-9][0-9]*([.][0-9]+)?)
```

Data manipulation

After all the samples have been pre-processed, it is possible to apply further transformations on the entire dataset as a whole. These are defined in the **manipulation** section. Two data manipulation operations are currently available:

- **standardization**: standardizes the feature specified as **target-feature**. Multiple instances are allowed.
- **quantization-correction**: causes all instantaneous features to be converted into cumulative in order to overcome poorly quantized output features. This manipulation is necessary when the granularity of the output feature defined in subsection D.1.3 is too coarse to promptly account for instantaneous variations in input features. By transforming each instantaneous feature into cumulative, no instantaneous contribution will be lost even in such a case.[2]

If there is no need for dataset-wide manipulation operations, an empty `<manipulation/>` tag is accepted.

Feature selection

Finally, it is possible to fuse multiple dataset features and create a new synthetic one. This can be done by adding **fuse** rules in the **selection** section under **features** section. A fuse rule is composed by the specification of the new synthetic **feature**, exactly as described in subsection D.1.1, and by a fusion **operation**.

Listing D.5: Fusion operation grammar

```
<MEMBER>    : =     <M-TERM> |
                    <M-TERM> + <MEMBER> |
                    <M-TERM> - <MEMBER>
<M-TERM>    :=   <M-FACTOR> |
```

[2]This manipulation requires a time-sorted dataset and so prematurely triggers dataset time sorting, causing a degradation of the computational performance

```
                    <M-FACTOR> * <M-TERM> |
                    <M-FACTOR> / <M-TERM>
<M-FACTOR>  :=    |<MEMBER>| | //Absolute value
                    (<MEMBER>) |
                    [IF](<FORMULA>)[THEN](<MEMBER>)[ELSE](<MEMBER>)
                        |
                    [LOG](<NUMBER>,<MEMBER>) |
                    [POW](<NUMBER>,<MEMBER>) |
                    [MIN](<MEMBER>,<MEMBER-LIST>) |
                    [MIN](<MEMBER>,<MEMBER-LIST>) |
                    -<MEMBER> |
                    <POINTER> |
                    <NUMBER>
<MEMBER-LIST>   :=   <MEMBER> | <MEMBER>,<MEMBER-LIST>
<FORMULA> :=   <TERM> |
                    <TERM>||<FORMULA>
<TERM>      :=    <FACTOR> |
                    <FACTOR>&&<TERM>
<FACTOR>    :=    <DIRECT-EXPRESSION> |
                    <NEG-EXPRESSION>
<DIRECT-EXPRESSION>  :=   <EXPRESSION> |
                                (<FORMULA>)
<NEG-EXPRESSION> :=  <EXPRESSION> |
                            !(<FORMULA>)
<EXPRESSION> := <MEMBER>=<MEMBER> |
                    <MEMBER>!=<MEMBER> |
                    <MEMBER>>=<MEMBER> |
                    <MEMBER><=<MEMBER> |
                    <MEMBER>><MEMBER> |
                    <MEMBER><<MEMBER>
<POINTER>      := [a-zA-Z_]\w*
<NUMBER>    := (0|[1-9][0-9]*([.][0-9]+)?)
```

It is also possible to exclude features from the dataset supplied to the subsequent phases. This can be achieved by setting a **target-feature** reference in the **exclude** section.

If neither fusions nor exclusions are required, it is necessary to acknowledge it explicitly by specifying the `<all-as-is/>` tag, as a replacement for the `<selection>` tag. Future works will enable *automatic feature selection* and this MARC feature will be accessible specifying the `<auto/>` tag, as a replacement for the `<selection>` tag.

Listing D.6: Phase1 configuration example

```
<phase1>
```

```
  <coherence>
    <rule>
      <condition>feature_1 = 1 &amp;amp;&amp;amp; feature_2 != 1</condition>
      <corrections>
        <actuation>feature_2 = 1</actuation>
      </corrections>
    </rule>
  </coherence>
  <manipulation>
   <standardization>
      <target-feature ref="feature_2"/>
   </standardization>
   <quantization-correction/>
  </manipulation>
  <features>
    <selection>
      <fuse>
        <feature>
           <name>new_feature</name>
           <col-index>19</col-index>
           <type>cumulative</type>
           <bounds>
             <lower>0</lower>
           </bounds>
        </feature>
        <operation>
          feature_1 + feature_2
        </operation>
      </fuse>
      <exclude>
        <target-feature ref="feature_1"/>
      </exclude>
    </selection>
  </features>
</phase1>
```

D.1.3 Phase 2A

The objective of this section is to configure the module that will generate an ARX model for each *device configuration*, building the pieces for the reconstruction of the piece-wise linear behavior MARC is able to describe.

In this section it is possible to define all characteristics about the *device* model. In particular, within the **configuration**, the user can describe

the features belonging to the *configuration*. Each combination of those features' values will be used to split the dataset coming from Phase 1. These configuration features are the phase's **target-feature**s and it is legal to specify none. It is also required to specify the **output-feature**, i.e. the feature that represents the resource of interest for consumption modeling. Moreover, time correlation levels between samples must be defined by specifying the regression lags: **ar** considering the output feature (at least 1), **ex** for all other non-configuration features (at least 0). Finally, to avoid improper time correlation, a **splitting delta** must be defined using the `<timestamp-splitting-diff>` in order to define when two consecutive samples are far enough in time to be defined uncorrelated.

Listing D.7: Phase2a configuration example

```
<phase2a>
  <configuration>
    <target-feature ref="feature_2"/>
    <target-feature ref="feature_3"/>
    <output-feature ref="feature_4"/>
  </configuration>
  <lags>
    <ar>1</ar>
    <ex>0</ex>
  </lags>
  <timestamp-splitting-diff>3</timestamp-splitting-diff>
</phase2a>
```

D.1.4 Phase 2B

NOT YET IMPLEMENTED

The objective of this section is to configure the markovian models able to reconstruct the piece-wise linear model by predicting the interleaving of each ARX model computed by Phase 2a.

In this section it is possible to define all characteristics about the *behavioral* model. In particular, within the **configuration**, the user can describe the features belonging to the *Markov chain state*. Each combination of those features' values will identify a state in the Markov chains, each of which identifying one or more ARX models generated by Phase 2A. These configuration features are the phase's **target-feature**s and must be a subset of the ones specified in Phase 2A's configuration; it is legal to specify none. Moreover, the user must specify others parameters related to Markov models:

- **periodicity**: the time span, defined in *simulation ticks*[3], that identifies the full period of development of the behavior. After that time span, the behavior repeats.
- **aggregation-threshold**: the minimum similarity percentage of Markov chains to consider two different time bands within a period as a single one. This parameter is highly dependent on the specific application field.
- **minimum-band**: the initial number of time bands evenly splitting the behavior period.

Listing D.8: Phase2b configuration example

```
<phase2b>
  <configuration>
    <target-feature ref="feature_2"/>
    <target-feature ref="feature_3"/>
  </configuration>
  <!--initially generates 5 Markov chains, one for each of the
      bands lasting 20 ticks (period/minimum-band=100/5=20)-->
  <periodicity>100</periodicity>
  <aggregation-threshold>0.75</aggregation-threshold>
  <minimum-band>5</minimum-band>
</phase2b>
```

D.1.5 Phase 2C

The objective of this section is to configure the model that will be used to estimate the exogenous contribution appearing in Phase 2A's ARX models.

In this section it is possible to define which features must be consider in order to split the *environment* model. In particular, within the **configuration**, the user can describe the features belonging to the *configuration*. Each combination of those features' values will be used to split the dataset coming from Phase 1. These configuration features are the phase's **target-feature**s and must be a subset of the ones specified in Phase 2A's configuration; it is legal to specify none.

Listing D.9: Phase2c configuration example

```
<phase2c>
  <configuration>
```

[3]See subsection D.1.6

```
    <!--Zero or more repetitions:-->
    <target-feature ref="feature_1"/>
    <target-feature ref="feature_2"/>
    <target-feature ref="feature_3"/>
  </configuration>
</phase2c>
```

D.1.6 Phase 3

The objective of this section is to finally configure the generative step of MARC that estimates the consumption of the selected output resource[4] by mixing all the models computed in each Phase 2 subphase.

In this section the user, first of all, defines the charge and discharge simulations conditions. This is done with two tags, `<charge>` and `<discharge>`, each one characterized by:

- **start-from** and **initial-condition** to define the simulations starting values.
- **stop-at** and **threshold** to define the simulations stopping values

We stress the fact that it is necessary one initial condition for each regression degree specified on **ar** tag in Phase 2A configuration. Moreover, some general parameters are required:

- **clock**: the real time, counted using the unit of the time feature[5], that corresponds to a single tick.
- **max-ticks**, the maximum number of ticks in a simulation can last before being considered inconclusive.
- **level-size** RESERVED FOR FUTURE EXTENSIONS
- **simulation-limit** RESERVED FOR FUTURE EXTENSIONS

Listing D.10: Phase3 configuration example

```
<phase3>
  <charge>
    <start-from>
      <initial-condition>0</initial-condition>
    </start-from>
```

[4] See subsection D.1.3
[5] See subsection D.1.1

```
    <stop-at>
      <threshold>100</threshold>
    </stop-at>
  </charge>
  <discharge>
    <start-from>
      <initial-condition>100</initial-condition>
    </start-from>
    <stop-at>
      <threshold>0</threshold>
    </stop-at>
  </discharge>
  <max-ticks>17280</max-ticks>
  <clock>10000</clock>
  <!--RESERVED FOR FUTURE EXTENSIONS: Use always the following
      settings-->
  <level_size>99</level_size>
  <simulation_limit>100</simulation_limit>
</phase3>
```