FP7 – 216693 - MULTICUBE Project

MULTI-OBJECTIVE DESIGN SPACE EXPLORATION OF MULTI-PROCESSOR SOC ARCHITECTURES FOR EMBEDDED MULTIMEDIA APPLICATIONS

Deliverable D2.3.2: Refined and extended multi-objective evaluation metrics

Revision [2]

Delivery due date: M24 (December 2009)
Actual submission date: February 2nd, 2010
Lead beneficiary: POLIMI

Dissemination Level of Deliverable		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	
Nature of Deliverable		
R	Report	X
P	Prototype	
D	Demonstrator	
O	Other	

Author(s):	*Cristina Silvano, Vittorio Zaccaria, Gianluca Palermo* (POLIMI)		
Reviewer(s):	*Carlos Kavka, Enrico Rigoni, Alessandro Turco (ESTECO), Giovanni Mariani* (ALARI) *and Prabhat Avasare (IMEC)*		
WP/Task No:	2.3	**Number of pages:**	38
Identifier:	D2.3.2_POLIMI_12.2009	**Dissemination level:**	Public
Issue Date:	February 2nd, 2010		

Keywords: Analytical Modeling, Response Surface Modeling, Exploration

Abstract: This deliverable (issued at M24) presents the results of the activity carried on in Task 2.3 (High-level evaluation metrics) under the leadership of POLIMI, with contributions from POLIMI and ALARI and reviewed by ESTECO and IMEC. In particular, this deliverable is an updated version of the one released at M18 D2.3.1 – Initial multi-objective evaluation metrics.

Based on the metrics defined in Task 1.2, the main goal of this deliverable is to present the analytical techniques (also called Response Surface Methods, RSMs) adopted to model the system metrics in the design space without requiring the simulation of all the possible architectural configurations.

The analysis presented in this deliverable has been extended with respect to those done in D2.3.1 producing the final deliverable for the task 2.3, in terms of analytical techniques, validation on the methods on the MULTICUBE use cases and integration within the MULTICUBE design space exploration framework.

This report contains:

- A summary of the motivations to use analytical techniques when facing the problem of design space exploration;
- The description of the analytical techniques analyzed and implemented;
- Validation results also on MULTICUBE use cases;
- The description of the integration of the presented techniques within the MULTICUBE exploration framework.

The Project Coordinator

Approved by the Project Coordinator: 	**Date:** February 2nd, 2010

Table of contents

I. Executive summary

This deliverable presents the use of some analytical techniques to support the design space exploration. In particular, the deliverable outlines the need of an alternative way to evaluate system metrics instead of simulations, presenting both interpolative and regressive techniques.

The analytical techniques considered for supporting the design space exploration and described in this deliverable are the following:

- **Shepard Interpolation**
  It is an interpolative technique that evaluates the response function in unknown points as the sum of the value of the response function in known points weighted with the inverse of the distance.

- **Radial Basis Functions**
  It is an interpolative technique that evaluates the response function in unknown points by the sum of a set of radial functions each one centered in known points.

- **Linear Regressions**
  It is a regressive technique that evaluates the response function in unknown points by modeling a linear relationship between the dependent response function and the independent design variables (or combinations of them)

- **Artificial Neural Networks**
  It is a non-linear regressive technique that evaluates the response function in unknown points by modeling the system with a set on interconnected processing elements (neurons), inspired by the way biological nervous systems work.

- **Spline-based Regressions**
  It has been recently proposed by Lee and Brooks [16][17] as a powerful method for the prediction of system metrics in the context of microprocessor architectures. In particular, the technique implements a linear regression using spline functions of the independent variables instead of the plain independent variables.

Currently, the analytical techniques have been implemented as software modules and integrated within the current version 1.0 of the open-source M3Explorer exploration framework. The analysis presented in this deliverable is an extension of the first deliverable related Task 2.3 (D2.3.1), where the new spline-based regression method, the validation of the analytical technique on to MULTICUBE use cases and the integration into the M3Explorer exploration framework have been added.

This report is structured as follows. Section II outlines the differences between this deliverable and the previous deliverable (D2.3.1) related to task 2.3. Section III outlines why there is the need of analytical techniques in the design space exploration problems faced by MULTICUBE and some problems related to the analytical modeling. Sections IV to VIII present the interpolative and regressive techniques developed up to M18, while Sections IX and X present some validation results of the previously defined techniques. Finally, while Section XI outlines the interfaces defined for the development of the analytical models in software modules, Section XII shows the integration of the analytical techniques within M3Explorer v1.0.

II. What's new with respect to D2.3.1 (***NEW***)

This section has been introduced in the deliverable to outline the main differences with respect to the previous version (D2.3.1 – Initial multi-objective evaluation metrics – released at M18).

Instead of writing a completely new deliverable including only the new parts developed in the last 6 months of T2.3, we decided to update the previous version of the deliverable with the new contents. In this way, this deliverable results self-contained without too much cross-references with the previous deliverable.

To increase the readability to people that have already read the previous deliverable, in the document we marked the new content in the following way:

- Sections that are completely new in this deliverable have been marked with the tag ***(***NEW***)***. In this case all the subsections (if any) related to the new section are new and have been tagged with ***(***NEW***)***;
- Sections and subsections that in this deliverable have been updated with new contents have been marked with the tag ***(***UPDATE***)***;
- Subsections that are completely new in this deliverable have been marked with the tag ***(***NEW***)*** and the related Section have been marked with ***(***UPDATE***)***.

In particular this deliverable improves the previous version by adding the following analysis developed during the last 6 months of the Task 2.3:

1. The state of the art non-linear regression technique based on Spline has been added in the pool of the analytical techniques. It has been presented recently [16][17] as a powerful method for the prediction of power consumption and performance metrics in the context of microprocessor architectures with large design space. (SECTION VIII).
2. The validation results presented in D2.3.1 have been extended introducing the Spline based regression. The results have been obtained by using the same design space and evaluation methodology as done for the previous techniques. (SECTION IX).
3. The validation results have been further extended on two MULTICUBE use cases. The two use cases have been selected considering two different size of the design space: a small and a large one. (SECTION X).
4. The analytical techniques have been completely integrated within the M3Explorer framework. In fact, in the release of M3Explorer 1.0 the RSM modules are downloadable in the same package. (SECTION XII).

III. Analytical techniques for fast evaluation of the system metrics

Nowadays, Multi-Processor Systems-on-Chip (MPSoCs) and Chip-Multi-Processors (CMPs) [1] represent the de facto standard for both embedded and general-purpose architectures. In particular, customizable MPSoCs supported by parallel programming have become the dominant computing paradigm for application-specific processors. In fact, they represent the best compromise in terms of a stable hardware platform that is software programmable, thus customizable, upgradable and extensible. In this sense, the MPSoC paradigm minimizes the risk of missing the time-to-market deadline while ensuring greater efficiency due to architecture customization and software compilation techniques.

In this context where the Design Space Exploration covers a central role, it is important to underline the problem of the efficiency of this design phase.

In fact, in computer architecture research and development, simulation still represents the main tool to predict performance of alternative architectural design points. If we consider cycle-accurate system-level simulation, it requires a lot of simulation time and the exploration of the design alternatives can exceed practical limits. Furthermore, the growing trend towards chip multi-processor architectures amplifies this problem because the simulation speed linearly decreases by increasing the number of cores to be simulated (as shown in Figure 1).

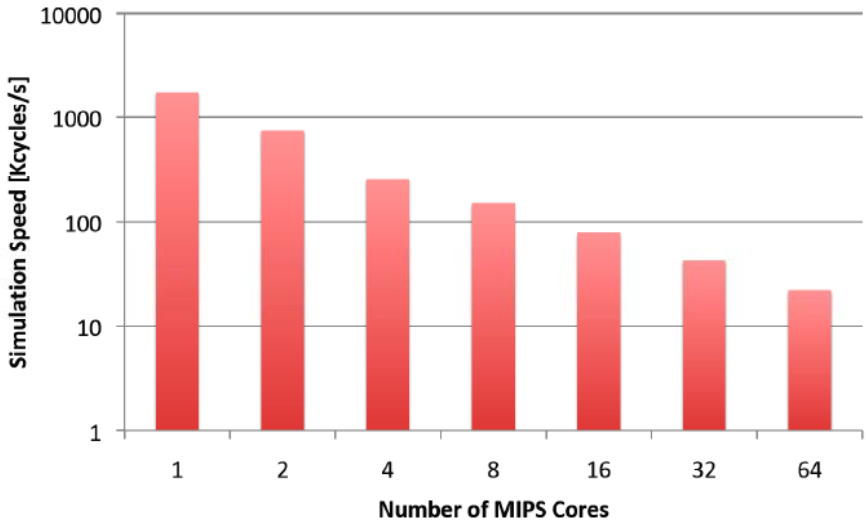

Figure 1 - Simulation speed of the SESC multiprocessor simulator [12] when executing the FFT kernel by varying the number of cores from 1 to 64 (Host machine: two Intel Xeons quad-core at 3Ghz).

While from the simulation point of view, the use of statistical sampling techniques seems to represent the best solution (e.g. SIMPOINT [2]), from the design space exploration point of view efficient solutions that reduce the number of architectural alternatives to be analyzed are only recently under analysis.

To face the problem of an efficient design space exploration within MULTICUBE, we decided to try to adopt techniques coming from the fields of statistic and machine learning, to model in an analytical way (Response Surface Methods) the behavior of the system metrics into design space.

III.1. Advantages in using analytical techniques

Response Surface Modeling (RSM) techniques allow determining an analytical dependence between several design parameters and one or more response variables (what we called system metrics). The working principle of RSMs is to use a set of simulations either generated ad hoc by a Design of Experiment (DoE) [3] or obtained by an exploration strategy applied before to the design space, in order to obtain a response model.

A typical RSM flow involves two main phases: a training phase and a prediction phase. While in the training phase, known data (also known as training set) is used to identify the RSM configuration, in the prediction phase the RSM is used to forecast unknown system response. RSMs are an effective tool for analytically predicting the behavior of the system platform without resorting to a system simulation.

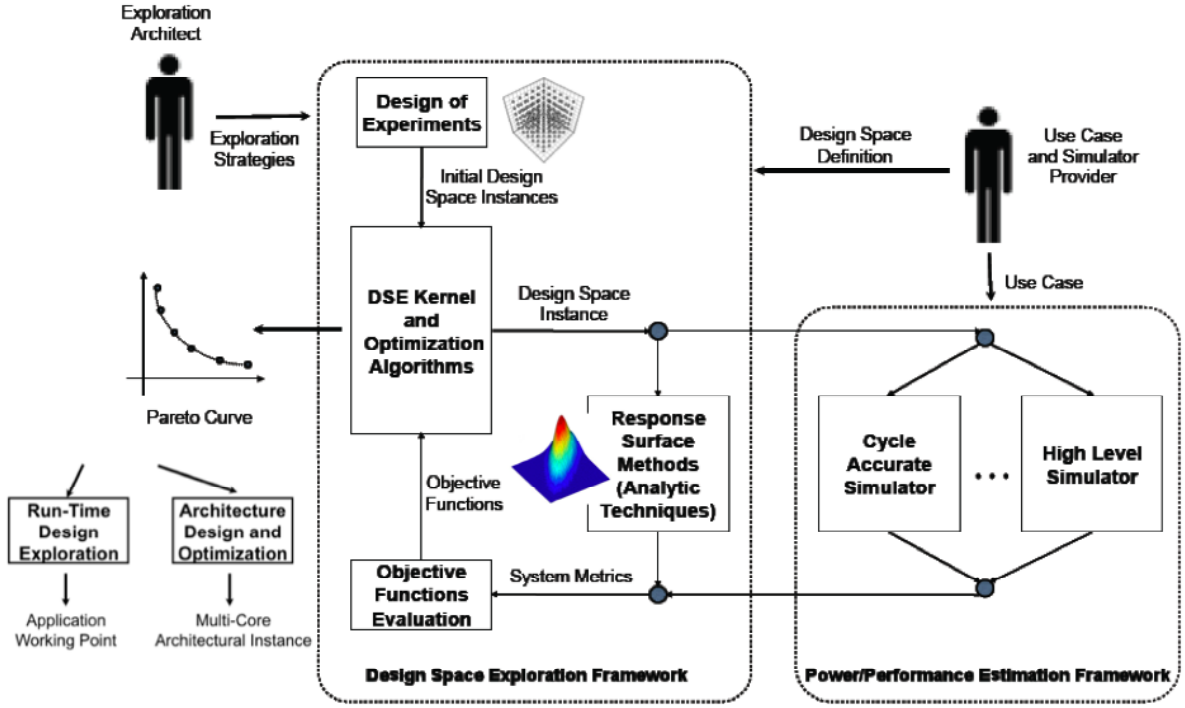

Figure 2 - Detailed view of the MULTICUBE design flow

Figure 2 shows the MULTICUBE design flow (with details regarding the design space exploration framework) where it is easy to identify the analytical techniques as an alternative to the actual simulation. In this way, the exploration strategy can selectively use simulation-based evaluation or analytical estimation.

III.2. Considerations on the analytical techniques

The advantage in using analytical techniques instead of actual simulation is very clear. In fact, it is possible to predict the value of the system metrics and their behavior without requiring simulation, or more precisely only requiring a small number of simulations to be used in the training phase.

On the other hand, the main problem in using analytical techniques is also very clear and it is related to the accuracy of the prediction. In fact, if the system behaviors predicted by the analytical techniques are not accurate, all the exploration and analysis done by using RSM are useless. In using analytical techniques, we have to take care that during the prediction phase it is not possible to understand what is the accuracy of the predictions and so how useful are the analysis that we are performing on that predictions.

III.2.1. Design of Experiments

The training data distribution is one of the most important problems for the prediction of an analytical model especially when the number of training data is limited.

The term Design of Experiments (DoE) [4] is used to identify the planning of an information-gathering experimentation campaign where a set of variable parameters can be tuned. Design of experiments is a discipline that has very broad application across natural and social sciences and encompasses a set of techniques whose main goal is the screening and analysis of the system behavior with a small number of simulations. Each DoE plan differs in terms of the layout of the selected design points in the design space. Several design of experiments have been proposed in the literature so far. Here in the following we will cite only three among the most used, showing different characteristics:

- *Random*. In this case, design space configurations are picked up randomly by following a Probability Density Function (PDF).

- *Full factorial*. In statistics, a factorial experiment is an experiment whose design consists of two or more parameters, each with discrete possible values or ”levels”, and whose experimental units take on all possible combinations of these levels across all such parameters. Such an experiment allows studying the effects of each parameter on the response variable, as well as the effects of interactions between parameters on the response variable. The most important full-factorial DoE is called *2-level full factorial*, where the only levels considered are the minimum and maximum for each parameter.

- *Central composite design*. A Central Composite Design is an experimental design specifically targeted to the construction of response surfaces of the second order (quadratic) without requiring a three-level factorial.

It is important to note that, once the design space has been defined, while factorial and central composite DoE layouts require a fixed number of point, for the Random DoE the number of points is one of the parameter of the DoE.

III.2.2. Overfitting

One of the critical issues in developing analytical models is the generalization of the models, and in particular for models trained by a set of known data, how well the model will be able to make predictions for cases that are not in the initial training set?

Especially regressive methods and artificial neural networks can suffer the problem defined as

overfitting. In fact, overfitting generally occurs when the model is excessively complex in relation to the amount of data available. A model, which has been overfit, will generally have poor predictive performance, as it can exaggerate minor fluctuations in the training data. Figure 3 shows a typical example of the overfitting risk where the model learned the training data instead of the real system behaviour. The left part of the figure shows the training phase while the right part shows the prediction phase.

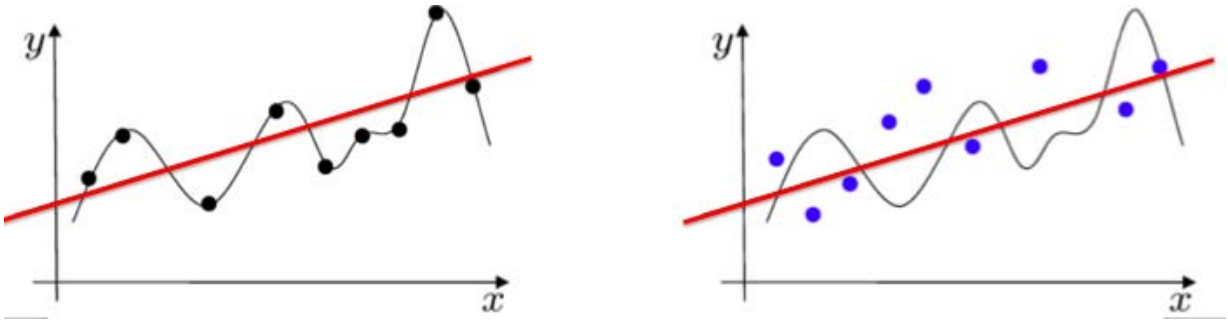

Figure 3 - Simple example showing the overfitting problem

In the left figure, some training data (black dots) have been generated by adding a random noise to a linear dependence between the design variable **x** and the response variable **y** shown graphically by the red line. By using those training data a complex model has been trained obtaining a predicted relationship between the design variable and the response variable carachterized by the black line. In the right figure, other data (blue dots) have been produced with the same technique than before. It is easy to see that the prediction capability of the model obtained during the training phase is very low since the model has been overfit the training data.

There are several methods to avoid the overfitting risk, but two are the most used techniques that are also adopted in the development of the analytical models used in MULTICUBE: the simplest one suggests to use simple models in order to reduce the overfitting risk, while the second one is called early stopping criterion. The early stopping criterion suggests to divide the training data into two sets: a training set and a validation set. The train phase uses only the training set while the validation set is used to verify the prediction capability of the trained model. As the validation set is independent of the training set, the prediction capability done on the validation set is a good measure of generalization for the model.

III.2.3. Pre-processing and scaling data

Data used to derive analytical models, also if originated from the same source/modeled architecture, due to different positions in the design space can have different values distribution and order of magnitudes. Especially when the case is the later, to create better prediction it is better to *preprocess* and/or *scale* data.

Data transformation is very important because in most of the cases the analytical models used to predict the data prefer (the computed prediction will works better) when data distribution follow some rules. As an example, if we consider an analytical model that uses the standard deviation of the training data to predict unknown data, this standard deviation values can be very high if the data distribution is skewed. In this case it is highly recommended to first

transform the data to approach a better symmetry and then to perform the model training and related prediction.

Box-Cox power transformation. A powerful transformation adopted in above-mentioned cases is called Box-Cox power transformation [5]. The Box-Cox power transformation is a useful data preprocessing technique used to reduce data variation, make the data more normal distribution-like and improve the correlation between variables. The power transformation is defined as a continuously varying function, with respect to the power parameter λ:

$$y_k^{(\lambda)} = \begin{cases} (y_k^{\lambda} - 1)/\lambda, & \text{if } \lambda \neq 0 \\ \log y_k, & \text{if } \lambda = 0 \end{cases}$$

In the validation results of the models that we adopted in MULTICUBE we considered a family of transformations as potential candidates λ {1, 0.5, 0,−1}. All the Box-Cox power transformations are only defined with positive values. In case of negative values, a constant value has to be added in order to make them positive.
We remark that to keep the prediction consistent with the actual objective functions of the target problem, an inverse Box-Cox transformation has been applied on the predicted data.

Centering and scaling data. Another pre-processing step that is usually applied to the data after the data transformation is the centering and scaling step. The goal of this step is to remove the bias from the input data (mean equal to zero) and to standardize the variance (standard deviation equal to 1). This transformation is also called *Autoscaling*. When the autoscaling transformation is applied to a set of data, from each value the mean value is removed and it is scaled by the standard deviation: $y_{autoscaled} = (y_{original} - \mu_y) / \sigma_y$.

IV. Shepard Interpolation

The Shepard Interpolation technique [6] is a well-known method for multivariate interpolation. This technique is also called inverse distance weighting (IDW) method because the value of the response function in unknown points is the sum of the value of the response function in known points weighted with the inverse of the distance.

IV.1. General Description

One of the most commonly used techniques for interpolation of scatter points is inverse distance weighted (IDW) interpolation. Inverse distance weighted methods are based on the assumption that the interpolating surface should be influenced most by the nearby points and less by the more distant points. The interpolating surface is a weighted average of the scatter points and the weight assigned to each scatter point diminishes as the distance from the interpolation point to the scatter point increases. The Shepard Interpolation technique is one of the most used methods for inverse distance weighted (IDW) interpolation.
In particular, the value of a response function r(**x**) for an unknown design **x** is computed by using N known observations y_k as follows:

$$r(\boldsymbol{x}) = \frac{\sum_{k=1}^{N} w_k(\boldsymbol{x}) y_k}{\sum_{k=1}^{N} w_k(\boldsymbol{x})}$$

where:

$$w_k(\boldsymbol{x}) = \frac{1}{\mu(\boldsymbol{x}, \boldsymbol{x}_k)^p}$$

is the weighting function defined by Shepard, p is the power of the model and μ is the distance between the known point x_k and the unknown point x. It is important to note that the previous formulas are related only for unknown points, if x is one of the known point x_k the function r(**x**) is equal to the observation y_k.

IV.2. Model Selection

For the case of the Shepard interpolation, the model selection phase is not so complex as for other models. In fact, in this case the only model tuning possibility that we have is the value of the exponent p that is referred in literature as the *power of the model*. Increasing the value of p decreases the influence of known points that are far from the point to be estimated, while decreasing the value of p increases that impact.

V. Radial Basis Function

Radial basis functions (RBF) [7] represent a widely used interpolation/approximation model for multivariate problems. In particular, Radial functions are special classes of functions that have as main characteristic feature the fact that (in most of the cases) their response decreases (or increases) monotonically with distance from a central point.

V.1. General Description

Radial basis functions (RBF) represent a widely used interpolation/approximation model. The interpolation function is built on a set of training configurations x_k as follows:

$$r(\boldsymbol{x}) = \sum_{k=1}^{N} \lambda_k \, \phi(\|\boldsymbol{x} - \boldsymbol{x}_k\|)$$

where ϕ is a scalar distance function, λ_k are the weights of the RBF and N is the number of samples in the training set. In MULTICUBE, we consider the following definitions for ϕ[1]

$$\phi(z) = \begin{cases} z & \text{linear} \\ z^2 \log z & \text{thin plate spline} \\ (1+z^2)^{1/2} & \text{multiquadric} \\ (1+z^2)^{-1/2} & \text{inverse multiquadric} \\ e^{-z^2} & \text{gaussian} \end{cases}$$

The weights λ_k are the solution of a matrix equation which is determined by the training set of configurations x_k and the associated observations y_k:

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

where

$$A_{jk} = \phi(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|), \qquad j,k = 1,2,\dots,N,$$

V.2. Model Selection

As can be easily noted by the general description section, the prediction capability on independent test data of the Radial Basis Functions is strongly related on the type of function ϕ used as radial function. For the case of RBF, the model selection phase have to decide what is the most suitable radial function ϕ for the target problem. Linear, thin plate spline, multiquadric, inverse multiquadric and gaussian are the possible functions we considered in MULTICUBE.

[1] The thin plate spline function is usually defined only for 2 dimension problems. In our analysis we adopt that ϕ function also for other problem sizes.

VI. Linear Regression

Linear regression [5] is a regression method that models a linear relationship between a dependent response function f and some independent variables x_i, i = 1…p plus a random term ε. The linear regression is the most common used analytical technique for understanding the relationship among system metrics and system parameters.

VI.1. General Description

Linear regression is a technique for building and tuning an analytic model r(x) as a linear combination of x's parameters in order to minimize the prediction residual ε.
We apply regression by taking into account also the interaction between the parameters and the quadratic behavior with respect to a single parameter. We thus consider the following general model:

$$r(\boldsymbol{x}) = \alpha_0 + \sum_{j=1}^{n} \alpha_j x_j^2 + \sum_{l=1}^{n} \sum_{j=1, j\neq l}^{n} \beta_{l,j} x_l x_j + \sum_{j=1}^{n} \gamma_j x_j$$

where x_j is the level associated with the j-th parameter of the system configuration. Least squares analysis can be used to determine a suitable approximation of the parameters. The least squares technique determines the values of unknown quantities in a statistical model by minimizing the sum of the squared residuals (the difference between the approximated and observed values).

VI.2. Model Selection

A well known measure of the quality of fit associated with the resulting model obtained by applying the linear regression is called *coefficient of determination* and it is defined as follows:

$$R^2 = \frac{\sum_k (y_k - \bar{y})^2}{\sum_k (r_k - \bar{y})^2}$$

where y_k is the k-th observation, $\bar{y}$ is the average of the observations, and r_k is the prediction for the y_k observation. R^2 corresponds to the ratio of variability in a data set that is accounted for by the statistical model. Usually, the higher R^2 the better is the quality of fit (with $0 \leq R^2 \leq 1$). Although adding parameters to the model can improve the R^2, there is a risk of exceeding the actual information content of the data, leading to arbitrariness in the final (fit) model parameters (overfitting). This reduces the capability of the model to generalize beyond the fitting data, while giving very good result on training-data.

To this purpose, an "adjusted" definition of the R^2 has been introduced in the past. This term adjusts for the number of explanatory terms in a model; it increases only if the terms of the model improve it more than expected by chance and will always be less than or equal to R^2; it is defined as:

$$1 - (1 - R^2)\frac{N-1}{N-p}$$

where p is the total number of terms in the linear model (i.e., the set of coefficients α, β, γ), while N is sample set size. Adjusted R^2 is particularly useful in the model selection stage of model building.

In order to understand the optimal number of terms of the linear model and the corresponding model order, we analyzed the behavior of the RSM cross-validation error and adjusted R^2.
In fact, the equation of the adjusted R^2 represents an improved measure of the overall quality of fit of the linear regression: it is inversely proportional to the model's degrees of freedom (i.e., N −p) which, in turn, depend on the order of the chosen polynomial r(x). As a matter of fact, higher degrees of freedom increase the chance of reduced variance of the model coefficients thus improving model stability while avoiding over-fitting.

For this reason, we limited the set of considered models to the following configurations:

1) First order model, without any interaction between parameters
2) First order model, with interaction between parameters
3) Second order model, without any interaction between parameters

Considering this, the model selection phase for the linear regression have the goal to select the more suitable model among the three models listed before.

VII. Artificial Neural Networks

An Artificial Neural Network (ANN) [8] is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the structure of the information processing system that is composed of a large number of highly interconnected processing elements (neurons) working in together. The learning process of ANNs, as in biological systems, involves adjustments to the synaptic connections that exist between the neurons. In particular, ANNs learn by example.

VII.1. General Description

Artificial Neural Networks (ANNs) are machine-learning models that automatically learn to approximate a target function (e.g. application performance, power consumption) based on a set of inputs (e.g. architectural parameters such as cache size or associativity). The following figure shows an example ANN consisting of three input neurons, four hidden neurons, and one output.

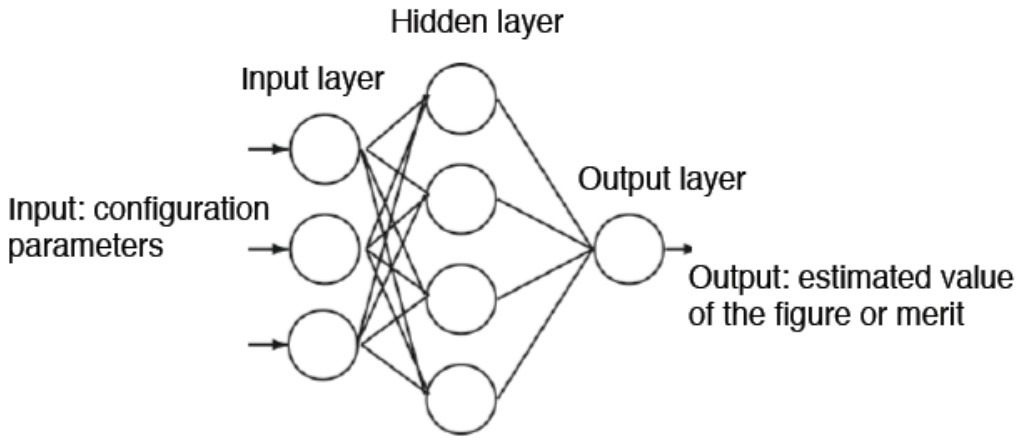

Figure 4 - Example of three layers ANN

In a fully connected feed-forward ANN, an input unit passes the data presented to it to all hidden units via a set of weighted edges. Hidden units operate on this data to generate the inputs to the output unit, which in turn calculates ANN predictions. Hidden and output units form their results by first taking a weighted sum of their inputs based on edge weights, and by passing this sum through a non-linear activation function (see following figure representing a generic neuron).

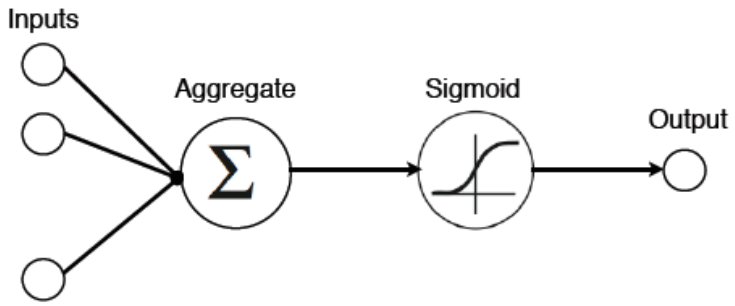

Figure 5 - Graphical representation of a single neuron adopting the sigmoid activation function

Increasing the number of hidden units in an ANN leads to better representational power and the ability to model more complex functions, but increases the amount of training data and time required to arrive at accurate models. ANNs represent one of the most powerful machine learning models for non-linear regression; their representational power is high enough to model multi-dimensional functions involving complex relationships among variables as for the cases faced in the MULTICUBE project.

VII.2. Model Selection

One of the key concepts for using the Artificial Neural Network is the model selection. In ANN we refer to model selection when we talk about the identification of the number of hidden layers and number of neurons for each hidden layers.
In MULTICUBE, we focus our attention on the Cascade 2 [9] algorithm for identifying a suitable topology for our ANN. The Cascade 2 algorithm is an iterative algorithm that selects candidate neurons to be added to the ANN by populating the hidden layers of the network and training the network weights. In particular, the topology built by Cascade 2 is composed by N hidden layers with only one neuron for layer. By using this algorithm, the number N of hidden layer is the only variable to be tuned during the model selection of the ANN.
To perform the model selection for the ANN avoiding the well-known problem known as over-fitting, the training set is partitioned into two subsets: a learning set and a validation set. While the learning set is used for identifying suitable candidate neurons, deciding its number and updating the network weights, the validation set is used only for computing the estimate prediction error of the neural network. In such way, a training stop (early stop) condition to avoid the over-fitting problem can be identified whenever the validation error starts to increase.

VIII. Spline-based Regression (***NEW***)

Spline-based regressions (also called Non-linear Regression based on spline) have been recently proposed by Lee and Brooks [16][17] as a powerful method for the prediction of power consumption and performance metrics in the context of microprocessor architectures presenting a large design space. In particular, the technique implements a linear regression using spline functions of the independent variables instead of the plain independent variable.
Spline-based regression is used to model and understand non-linear dependencies among system metrics and system parameters.

VIII.1. General Description (***NEW***)

Basic linear regression models assume the response behaves linearly in all predictors. This assumption is often too restrictive and several techniques for capturing non-linearity may be applied. The most simple of these techniques is a polynomial transformation on predictors suspected of having a non-linear correlation with the response (as described for the enhanced Linear Regression technique we used and presented before). However, polynomials have undesirable peaks and valleys. Furthermore, a good fit in one region of the predictor's values may unduly impact the fit in another region of values. For these reasons, we consider splines as a more effective technique for modeling non-linearity.

Doing that, the non-linear regression model *r* we adopted can be represented as follow:

$$r(\boldsymbol{x}) = \alpha_0 + \sum_{j=1}^{n} \alpha_j \phi_j(\boldsymbol{x}_j)$$

where x_j and ϕ_j are respectively the level and the spline function associated with the j-th parameter.

Spline functions are piecewise polynomials used in curve fitting. The function is divided into intervals defining multiple different continuous polynomials with endpoints called knots. The number of knots can vary depending on the amount of available data for fitting the function, but more knots generally leads to better fits. Relatively simple linear splines may be inadequate for complex, highly curved relationships. Splines of higher order polynomials may offer better fits and cubic splines have been found particularly effective [19]. Unlike linear splines, cubic splines may be made smooth at the knots by forcing the first and second derivatives of the function to agree at the knots. However, cubic splines may have poor behavior in the tails before the first knot and after the last knot [18]. Restricted Cubic Splines *RCS* that constrain the function to be linear in the tails are often better behaved and have the added advantage of fewer terms relative to cubic splines. In particular, the Restricted Cubic Splines with 3 knots can be written as follow:

$$RCS(\boldsymbol{x}_i, 3) = \beta_{0,i} + \beta_{1,i}\boldsymbol{x}_i + \beta_{2,i}\boldsymbol{x}_i^2 + \beta_{3,i}\boldsymbol{x}_i^3 + \beta_{4,i}(\boldsymbol{x}_i - k_1)^3 + \beta_{5,i}(\boldsymbol{x}_i - k_2)^3 + \beta_{6,i}(\boldsymbol{x}_i - k_3)^3$$

where β represents the set of polynomial coefficients and k_j is the j-th knot point.

In the RSM technique we implemented, RCS are used as basic spline (see Figure 6 for an example of restricted cubic spline in *x* divided in 5 knots).

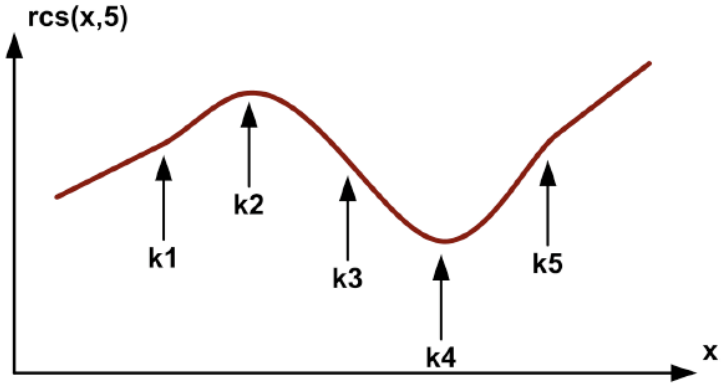

Figure 6 - Restricted cubic spline. Domain of x divided in five knots.

VIII.2. Model Selection (***NEW***)

The model selection of a spline-based regression considers mainly the following three factors:

- Spline function selection
- Number of knot selection
- Domain specific parameters interactions

Here in the follow, we will describe what have been the choice done during the model selection of the non-linear regression technique based on spline.

Spline selection. As presented in the general description we used a Restricted Cubic Spline (RCS) as basic transformation function for modeling a non-linear dependence between a system parameter and a system metric. More in detail, the basic model we used for the spline-based regression is the following where we apply regression by taking into account the spline function of the parameters:

$$r(\boldsymbol{x}) = \alpha_0 + \sum_{j=1}^{n} \alpha_j RCS(\boldsymbol{x}_j, nk_j)$$

where x_j and nk_j are respectively the level and the number of knots associated with the j-th parameter of the system configuration. Least squares analysis can be used to determine a suitable approximation of the parameters. In the particular case where the j-th parameter of the system configuration has only two levels, the spline associated to the parameter is considered as simple linear function.

Number of knots selection. The number of knots we used for each spline follows a thumb rule presented in [18]: five knots or fewer are generally sufficient for restricted cubic splines. While fewer knots may be required for small data sets, with a large number of knots increases the risk of over-fitting the data.
In particular, we adopted the following policies for the selection of the number of knots depending on the number of levels of the j-th parameter x_j:

- if the number of levels is greater than 5 the number of knots is equal to 5;
- if the number of levels is smaller than 3 the number of knots is 0 since the spline is substituted by a linear model (see spline selection);
- otherwise the number of knots is equal to the number of levels.

Domain specific parameters interactions. The regression method applied to the RCS functions of the input parameters is a linear model without interactions. This technique has been considered as the default method when the problem domain is unknown. However, if the designer has some knowledge of the design space, he can use it to better tune the model. In fact, as described by Lee and Brooks in [16][17], the basic method can be extended by adding a number of RCS terms including parameters interactions. In that case, the basic model can be extended in the following way:

$$r(\boldsymbol{x}) = \alpha_0 + \sum_{j=1}^{n} \alpha_j RCS(\boldsymbol{x}_j, nk_j) + \sum_{i=1}^{n}\sum_{j=1}^{n} \delta_{i,j}\alpha_{i,j} RCS(\boldsymbol{x}_i\boldsymbol{x}_j, nk_{i,j})$$

where x_j are the level and the number of levels associated with the j-th parameter of the system configuration, nk_x are the number of knots for each RCS and $\delta_{i,j}$ are the model variables defined by the designer that are usually equal to 0 except when the designer knows that there is a parameters interactions between the two input parameters x_i and x_j.

IX. Validation of the analytical techniques with a state of the art use case obtained by using the SESC simulator (***UPDATE***)

This section of the deliverable shows the prediction capability of the previous presented analytical techniques by using the model of a multiprocessor architecture with the SESC simulator. The results will be presented showing the performance of the analytical techniques by varying:

- The model parameters as presented for each technique in the model selection section
- The Box-Cox transformations used to preprocess the training data

The performance metric of the analytical techniques used as prediction capability is the average normalized error by varying the number of point used as training set. Each prediction have been repeated 5 time for each method and the presented results are in terms of average values.

Since at M18 of the project we were not able to make a "deep exploration process" with the MULTICUBE use cases, for the validation step described in D.2.3.1 we used results obtained by using an already existing open-source multiprocessor simulator [12].
In this deliverable, we decide not to remove this section but to update it because in the previous D.2.3.1 the Spline-based regression were not presented. In the updated version, Section IX.6 has been added to present the same validation results also for the spline-based regressions.
As stated in the introduction of this deliverable in section X we will show the validation of the analytical techniques on to the MULTICUBE use cases.

IX.1. Experimental Setup (***UPDATE***)

For the evaluation of the system metrics (mainly in terms of execution time and energy consumption) the SESC [12] simulator has been used. SESC is a fast simulator for chip-multiprocessor architectures with out-of-order processors that can provide energy and performance values for a given application. The evaluation of the energy consumption of the memory hierarchy is supported by CACTI [13], while the energy consumption computation due to the core logic is based on the WATTCH models [14]. The SESC simulator has been chosen also for the validation phase to be provided in the deliverables of Task 2.3 because it offers a good trade-off between simulation speed and modeling accuracy (less than 5% with respect to an actual implementation of the MIPS R10K architecture) and because it is widely adopted by the scientific community for multiprocessor design space modeling and exploration. In this way, the results provided in this deliverable can be considered credible results also if they are not related to MULTICUBE use cases.

The analytical models described in the previous sections have been used to predict the execution time and energy consumption of an MPSoC architecture. The target system architecture is composed of a shared-memory multiprocessor with private L2 caches. In the target architecture, a MESI snoopy-based coherence protocol acts directly among L2 caches, requiring additional invalidates/writes between L1 and L2 caches to ensure coherence of the data. Cache inclusion is maintained explicitly by using the mechanism adopted for propagating the coherence events in the cache hierarchy [10].

To provide a comprehensive yet affordable validation of the proposed methodology, we identified a design space composed of a set of 9 independent platform parameters presented in the following table.

Table 1 - Design space used for the validation phase

Parameter	Min.	Max.
# Processors	2	16
Processor issue width	1	8
L1 instruction cache size	2KB	16KB
L1 data cache size	2KB	16KB
L2 private cache size	32KB	256KB
L1 instruction cache assoc.	1w	8w
L1 data cache assoc.	1w	8w
L2 private cache assoc.	1w	8w
I/D/L2 block size	16	32

The corresponding design space size consists of $|X| = 2^{17}$ configurations (131072). If we assume a simulation time for each configuration of 10 minutes, then we will have more that 900 days of simulation in a single-core machine for the full search exploration. This underlines the need of alternative ways to explore the design space.

The selected target applications have been derived from the SPLASH-2 [11] benchmark suite (U = {FFT, OCEAN, LU, RADIX}); for each application we considered 3 different input data-sets, resulting into 12 application-data-set scenarios.

In the following, we validate the use of the proposed methods presenting a set of experimental results showing the average normalized error computed over a set of 15.000 randomly selected configurations of the design space. The results will be presented by varying a number of random configurations used as a training-set (known points) starting from 200 to 1800 configurations.

IX.2. Shepard Interpolation

Figure 7 shows the validation results for the Shepard Interpolations. In the figure, different lines represent different values of the power variable in the Shepard Interpolator and different Box-Cox transformations.

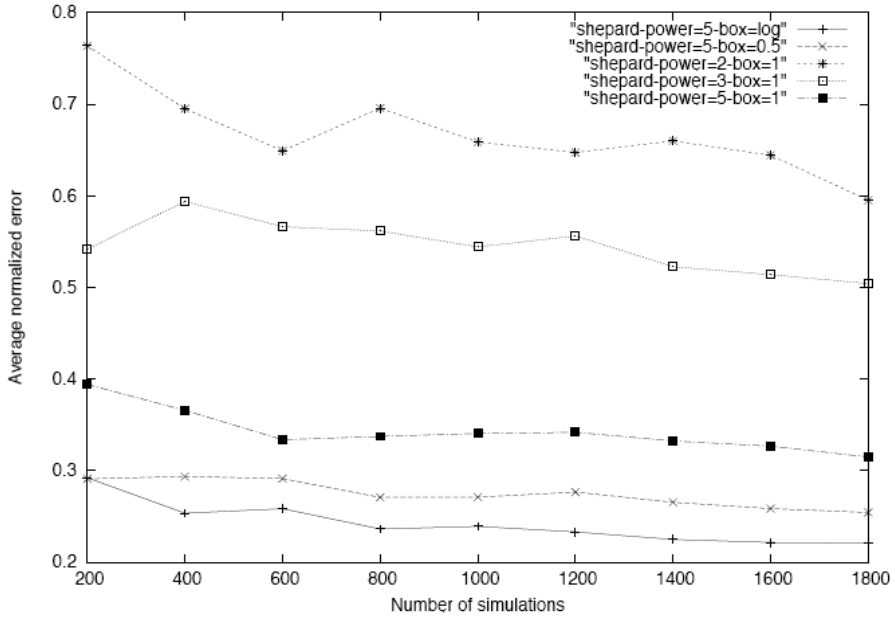

Figure 7 - Average normalized error for the Shepard Interpolation

In the figure, only the most significant power coefficients and preprocessing transformations have been shown. The prediction accuracy of all the selected models slightly increases (the average normalized error decreases) with the increment of the number of evaluated point in the training set. The model with a power value of 5 and $\lambda = 0$ provides the best performance in terms of mean error (less than 25%).

IX.3. Radial Basis Functions

Figure 8 shows the validation results for the Radial Basis Functions. In the figure, different lines represent different radial functions and different Box-Cox transformations.

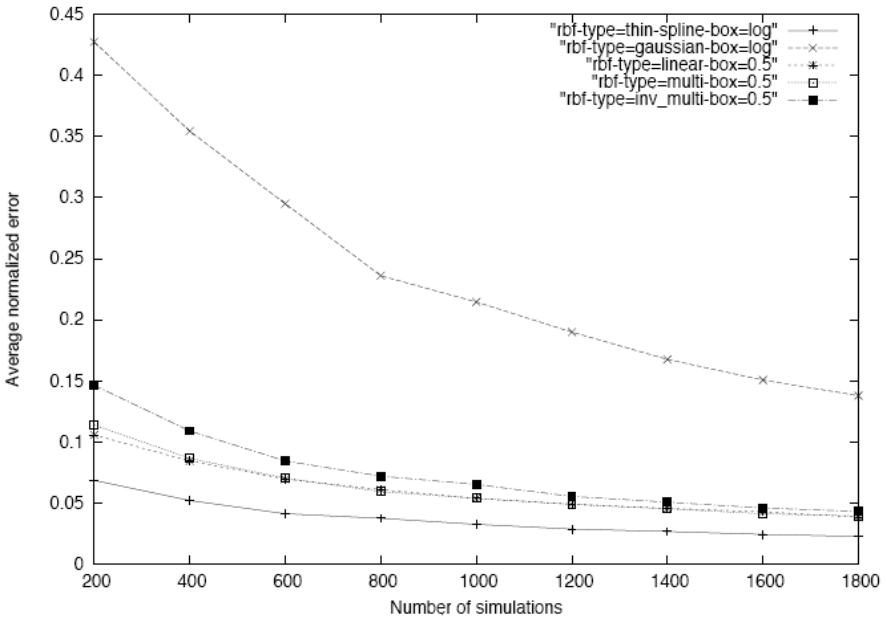

Figure 8 - Average normalized error for the Radial Basis Functions

The previous figure shows only the RBF functions and preprocessing transformations with the best performance. As for the previous case, the accuracy of the selected models increases with the increment of the number of training point. The RBF using a thin-plate spline radial function where the data have been pre-processed by using a logarithmic Box-Cox transformation present the best results reaching an average normalized error down to 3%.

IX.4. Linear Regressions

To validate the results in the use of the Linear Regression method we add also some results obtained for the "adjusted" R^2 (see Figure 9) instead of showing only the average error graph (see Figure 10). In figure different lines represent different linear models and different Box-Cox transformations.

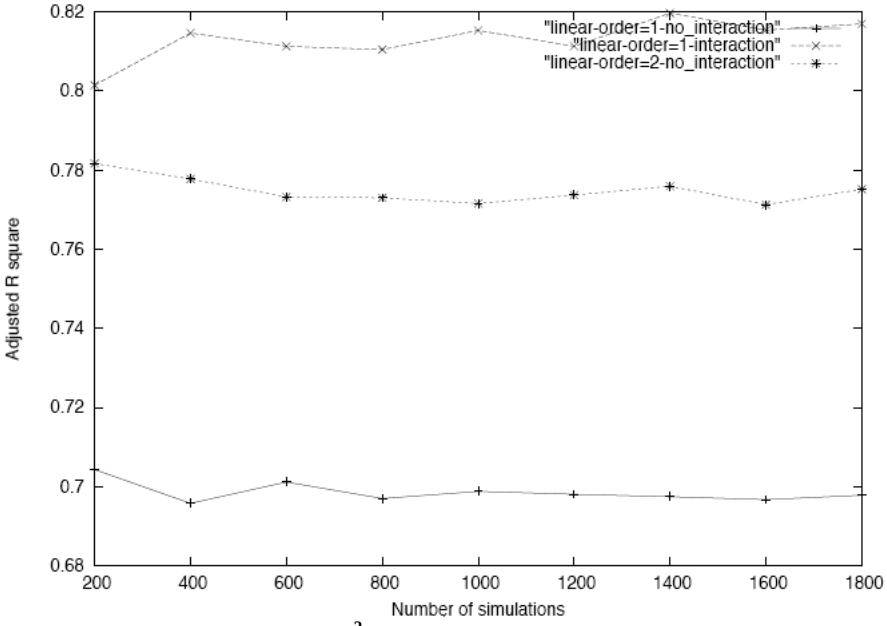

Figure 9 – Adjusted R^2 for the Linear Regression techniques

Figure 9 shows the average adjusted R^2 for the considered models by varying the number of simulations. The first order model with interaction between parameters presents the best results among all the possible model configurations in terms of ability to explain the linear dependence among all the parameters. The second order model without interaction between parameters is the second best, while the first order model without interaction is the worst one. The previous experimental considerations suggest to choose the first order model with interaction as the candidate model to be further analyzed introducing the Box-Cox power transformations.

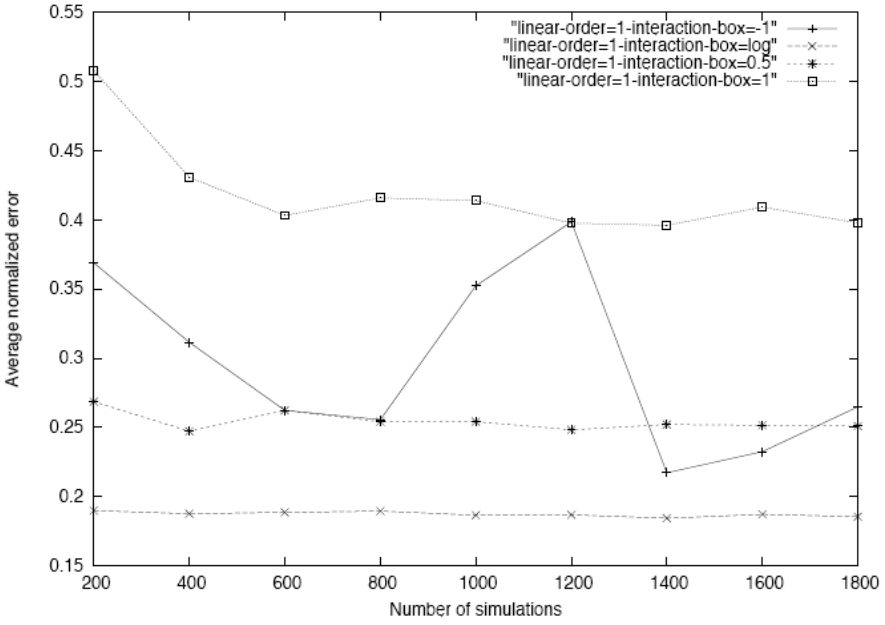

Figure 10 - Average normalized error for the Linear Regression techniques

Figure 10 shows that, among all the possible Box-Cox transformations, the logarithmic transformation is the most stable and provides the best approximation error for the considered model configuration. On the other hand, the $\lambda = -1$ transformation decreases the stability of the prediction model despite the trends is on decreasing the average error.

IX.5. Artificial Neural Networks

Figure 11 shows the validation results for the Artificial Neural Networks. In the figure the different lines represent different Box-Cox transformations applied to the Artificial Neural Network techniques.

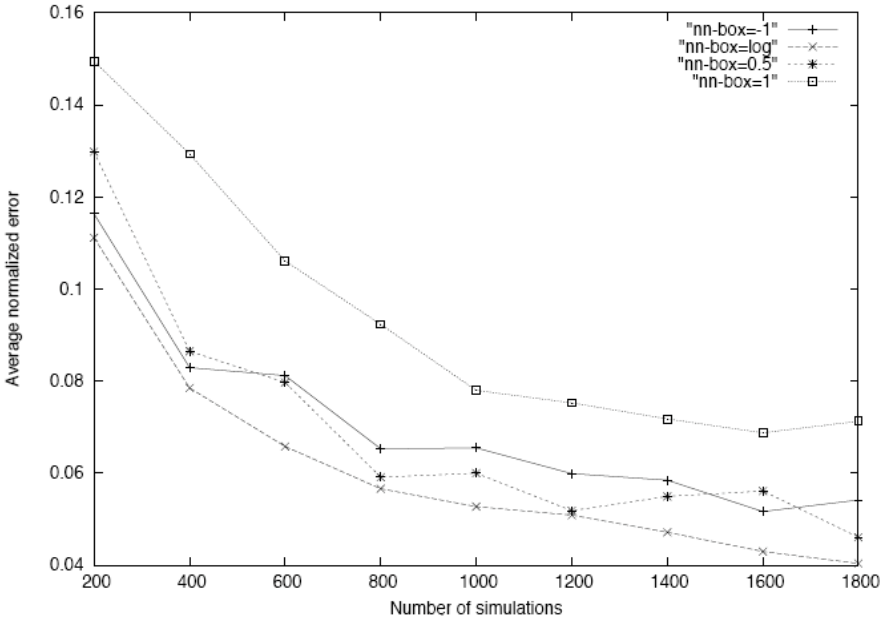

Figure 11 - Average normalized error for the Artificial Neural Networks

Although for every Box-Cox transform, the selected ANNs perform very well, with an error under 20%, the logarithmic Box-Cox transformation is the one that presents the most evident

advantages. The ANN performances are comparable to the performance presented before by RBFs. For sure ANN needs more points to be trained, in fact the left part of the graph presents higher values with respect to RBF, but on average are less sensible to the Box-Cox transformation adopted during the data preprocessing step.

IX.6. Spline-based Regression (***NEW***)

Figure 12 shows the validation results for the Spline-based regression. In the figure the different lines represent different Box-Cox transformations applied to the non-linear regression method based on spline.

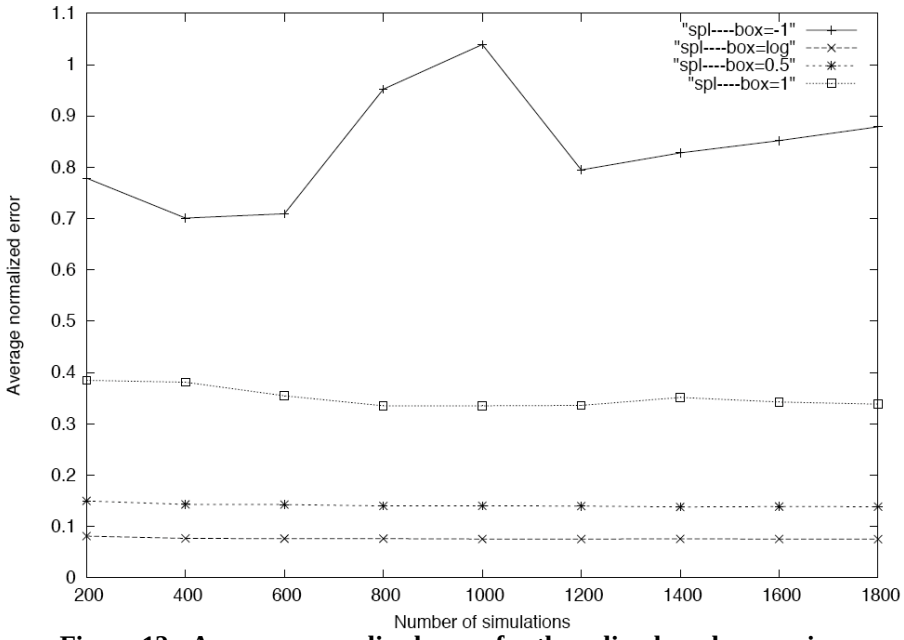

Figure 12 - Average normalized error for the spline-based regression

As for the linear regression, the prediction accuracy of the spline based regression models slightly increases (the average normalized error decreases) with the increment of the number of evaluated point in the training set. On the other side, the accuracy values obtained by this advanced non-linear regression mechanism results better than those obtained with the linear-one. In particular, by using the logarithmic Box-Cox, the model average normalized error is under 10%.

IX.7. Summary (***NEW***)

This section tries to summarize the validation results obtained by applying the analytical techniques to the prediction of system metrics of the multiprocessor architecture modeled on SESC. In particular, analyzing Figures 7, 8, 10, 11 and 12 it is possible to gather the following information:

- Prediction accuracy analysis
    - Radial basis functions and neural networks present the best average accuracy results, up to 5% and 4% respectively;

- Radial basis functions and Spline based Regressions present the best prediction accuracy with only 200 training configuration reaching values less then 10% of average prediction error;
- Radial basis functions and neural networks are also the two techniques that presents an average accuracy that is strongly dependent on the number of simulations used for the training;
- The prediction accuracy related to Shepard Interpolation, Linear-regression and Spline-based regression results to be less dependent on the number of simulated points used for the model training phase;

- Box-Cox parameter analysis
  - Neural networks model is the analytical technique that results to be less dependent on the value of the Box-Cox transformations
  - Spline-based regressions technique is the RSM that presents the maximum value of accuracy variation with respect to Box-Cox transformations, ranging from 8% to 90%;
  - On average the best values of the Box-cox transformation is the logarithmic one. Moreover it results the best transformation on all the presented techniques.

X. Validation of the analytical techniques with MULTICUBE use cases (***NEW***)

This section of the deliverable shows the prediction capability of the previous presented analytical techniques on the MULTICUBE use-cases. The results will be presented showing the performance of the analytical techniques by varying as in the previous section:

- The model parameters as presented for each technique in the model selection section
- The Box-Cox transformations used to preprocess the training data

The performance metric of the analytical techniques used as prediction capability is the average normalized error by varying the number of point used as training set. Each prediction have been repeated 5 time for each method and the presented results are in terms of average values.

X.1. Experimental Setup (***NEW***)

For the final validation of the analytical techniques developed within Task 2.3 to support the fast and efficient exploration of the design space, we decided to use two of the four use cases developed in the project:

- MULTIMEDIA use case
- LOW-POWER PROCESSOR use case

This choice has been done in order to have two different types of design spaces: the first one characterized by a small design space, while the second one by a larger one.

Regarding the analytical techniques, we will present in the following subsections the results obtained when the Box-Cox transformations are equal to $\lambda = \{1, 0.5, 0, -1\}$. In particular, for the linear regression we will show results with a first order model with interaction and second order model without interaction, for the Shepard interpolation we used the power value fixed to 5, while for the Radial Basis Function we used only the Thin Plate Spline function that has been demonstrated in the previous section (Figure 8) to be the best one.

X.2. MULTIMEDIA USE CASE (***NEW***)

We applied the prediction techniques to the MULTIMEDIA use case (see Figure 18) composed of an MPEG4 decoder application running on a multiprocessor architecture developed by IMEC. A more detailed description of the use case can be found in deliverable D1.3.

The used design space was composed by the following parameters:

- Number of CPU: [2, 4, 5, 6, 7, 8]
- Instruction cache size: [4KB, 8KB, 16KB, 32KB]
- Frequency: [40MHz, 80MHz, 120MHz, 160MHz, 200MHz]

The design space size of the analyzed MULTIMEDIA use cases results in 120 configurations. The validation results have been obtained using a training set varying from 20 configurations up to 60 configurations (50% of the design space), while as validation set all the remaining configurations.

Table 2 summarizes the validation results obtained by applying the analytical techniques to the prediction of system metrics of the multimedia use cases. In particular, analyzing Figures from 13 to 17 it is possible to gather the following information:

- Prediction accuracy analysis
    - Radial basis functions and Shepard interpolation present the best average accuracy results, up to 5% and 8% respectively;
    - Radial basis functions and Shepard interpolation are also the two techniques presenting an average accuracy that is strongly dependent on the number of simulations used for the training;
    - The prediction accuracy related to Spline-based and Linear regressions results to be less dependent on the number of simulated points used for the model training phase;
    - Neural networks model presents the worst results. This is mainly due to the fact that the number of training points used is very limited.
    - The previous consideration (limited number of training points) should also be taken into account when we compare the prediction accuracy obtained for this use-case with respect the other results.
- Box-Cox parameter analysis
    - Shepard interpolation model is the analytical technique that results to be less dependent on the value of the Box-Cox transformations;
    - Spline-based and Linear regressions are the two techniques presenting the maximum value of accuracy variation with respect to Box-Cox transformations, ranging from 10% to 50% on average;
    - On average, also for the MULTIMEDIA use cases the best values of the Box-Cox transformation is the logarithmic one.

Table 2 – Validation results of the analytical techniques on the MULTIMEDIA use case

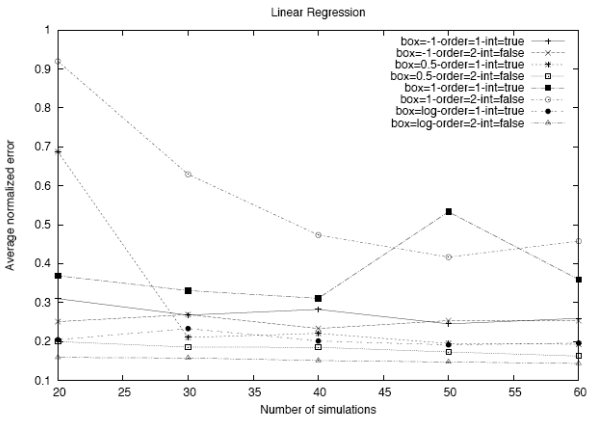

Figure 13 - Average normalized error for the linear regression technique

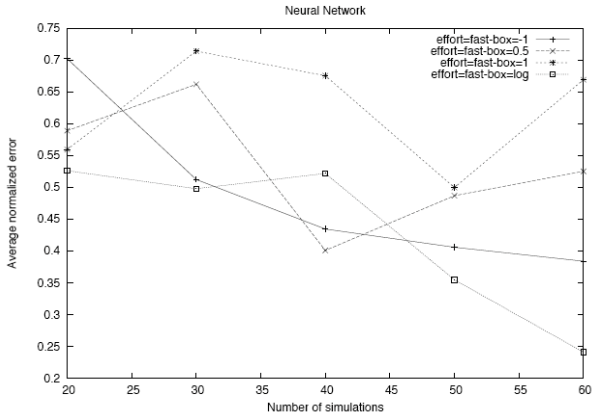

Figure 14 - Average normalized error for the neural networks technique

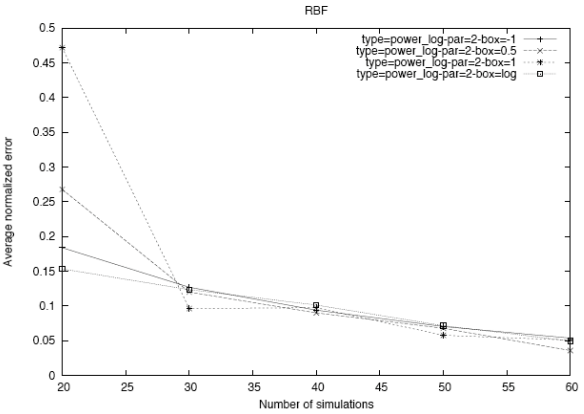

Figure 15 - Average normalized error for the Radial Basis Function technique

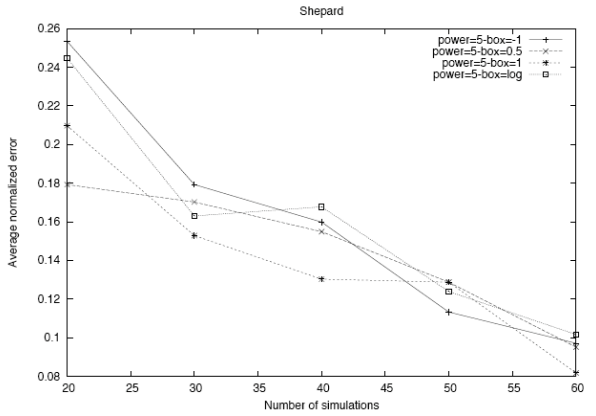

Figure 16 - Average normalized error for the Shepard Interpolation

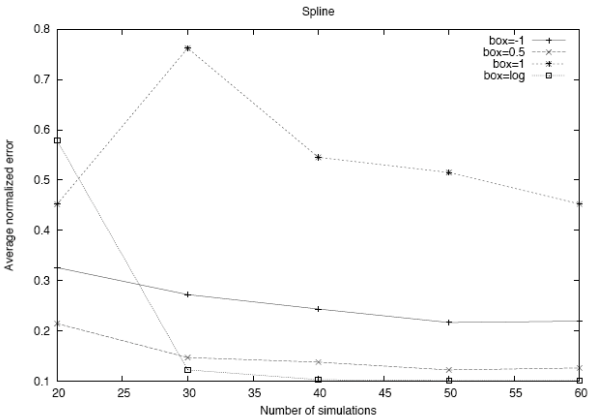

Figure 17 - Average normalized error for the Spline-based regression

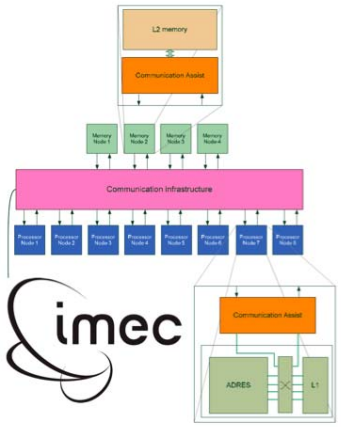

Figure 18 – MULTIMEDIA use case

X.3. LOW POWER PROCESSOR USE CASE (***NEW***)

In order to increase the validation results, we applied the prediction techniques to the LOW-POWER PROCESSOR use case (see Figure 24) composed by a low-power superscalar architecture developed by STMicroelectronics while it is running the CRAFTY application, one of the benchmark in the SPEC2000 suite. A more detailed description of the use case can be found in deliverable D1.3.

The used design space was composed by the following parameters:

- Reorder buffer depth: [32, 48, 64, 80, 96, 112, 128]
- Number of rename registers: [16, 32, 48, 64]
- Instruction window width: [8, 16, 24, 32]
- Instruction cache size: [16KB, 32KB, 64KB]
- Data cache size: [16KB, 32KB, 64KB]
- L2 cache size: [0KB, 256KB, 512KB, 1024KB]
- Load queue size: [16, 32]
- Store queue size: [16, 32]
- Number of miss holding registers: [4, 8]
- Branch history table size: [512, 1K, 2K, 4K]
- Branch target buffer size: [16, 32, 64, 128]

The design space size of the analyzed LOW-POWER use cases results in more than 500K different configurations that is bigger than both the previous analyzed validation cases. The validation results have been obtained using a training set varying from 200 configurations up to 1800 configurations (less than 1% of the design space). In this case the validation set is composed of 5.000 randomly selected configurations of the design space.

As in the previous section, Table 3 summarizes the validation results obtained by applying the analytical techniques to the prediction of system metrics of the low-power processor use cases. In particular, analyzing Figures from 19 to 23 it is possible to gather the following information:

- Prediction accuracy analysis
    - All the proposed analytical techniques presents a prediction accuracy with values under 2.5% of the average error with the minimum number of the used training sample.
    - Taking into account the limited variations of accuracy, Radial basis functions and Shepard interpolation are the techniques presenting an average accuracy that is more dependent on the number of simulations used for the training;
    - The best results have been obtained by using the Spline-based regression that reaches a prediction accuracy value close to 0.1%.
- Box-Cox parameter analysis
    - For the analyzed use case all the methods present a very limited prediction accuracy variation with respect to Box-Cox parameter values.

Table 3 – Validation results of the analytical techniques on the LOW-POWER PROCESSOR use case

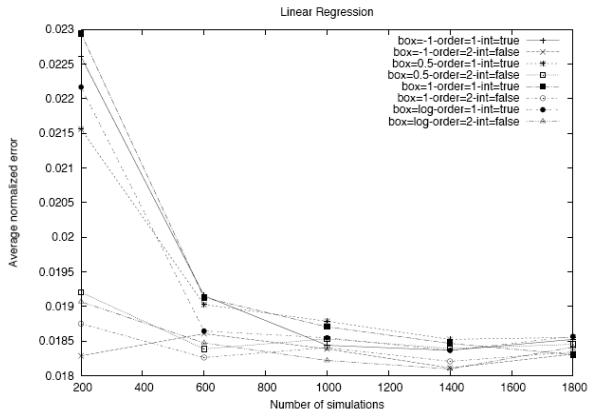

Figure 19 - Average normalized error for the linear regression technique

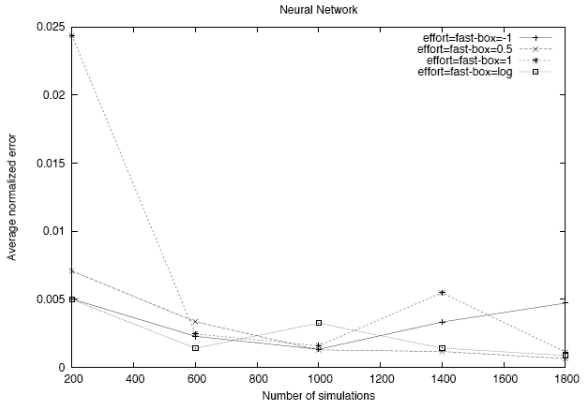

Figure 20 - Average normalized error for the neural networks technique

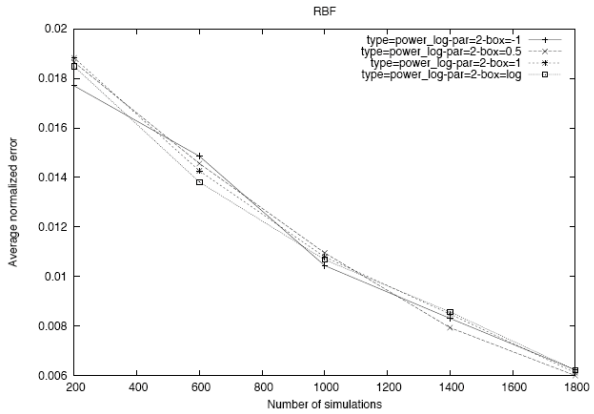

Figure 21 - Average normalized error for the Radial Basis Function technique

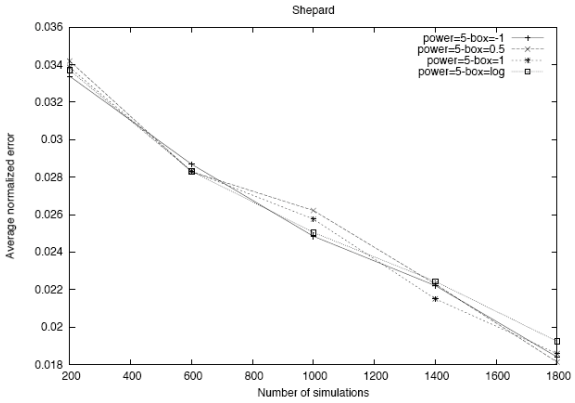

Figure 22 - Average normalized error for the Shepard Interpolation

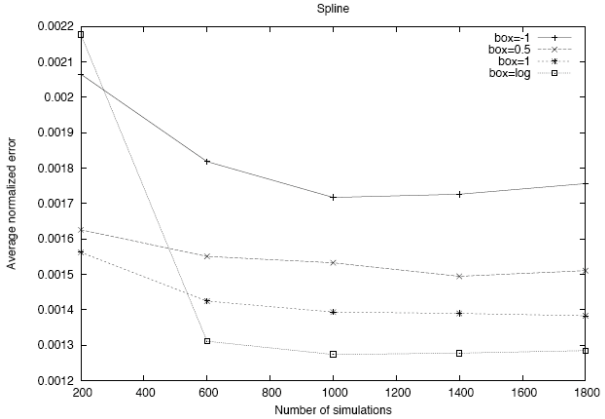

Figure 23 - Average normalized error for the Spline-based regression

Figure 24 - Low-Power use case

XI. Development of the software modules

The techniques described in Sections IV to IX have been implemented in software modules to be included in the M3Explorer tool, one binary for each analytical technique.

To interface the analytical models with the external world, we adopted a standard format for the data. The common interface used to exchange information among modules is what is called XDR standard format (eXternal Data Representation, derived from IETF standard [15]) and it simply consists of a text format where each row represents a design point. It has a first field that is a number which identifies how many data, separated by a space, will follow it and then, on the same row, can exist a new number to identify how many data will follow it an arbitrary number of times. When a row is finished a new line character is put so that is possible to identify a new row of the database. An example of a single line in the XDR format is given in figure 25.

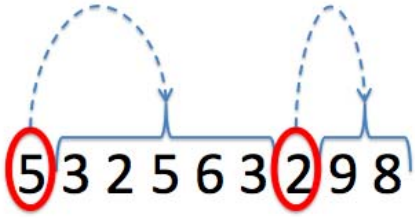

Figure 25 - Simple example of the XDR representation

This format is generic but it is also easy to be read. In fact, the meaning of the numbers in our specific case can be summarized in the following way: the size of the design vector (“5” in the example), the design vector (“3 2 5 6 3” in the example), the size of the response vector (“2” in the example) and the response vector (“9 8” in the example).

Considering not only a single line/point but also more points to be represented (as in the Figure 26) the design and response matrix can be easily identified.

```
9 0 1 2 0 3 3 3 0 2 2 0.286008 -0.552515
9 0 1 2 0 3 3 3 0 3 2 0.316722 -0.631457
9 0 1 2 1 1 1 0 0 0 2 -0.664549 0.801651
9 0 1 2 1 1 1 0 0 1 2 -0.654137 0.0934367
9 0 1 2 1 1 1 0 0 2 2 -0.649368 -0.310935
9 0 1 2 1 1 1 0 0 3 2 -0.630934 -0.499138
9 0 1 2 1 1 1 1 0 0 2 -0.59098 0.313031
9 0 1 2 1 1 1 1 0 1 2 -0.581711 -0.167422
9 0 1 2 1 1 1 1 0 2 2 -0.577237 -0.444602
9 0 1 2 1 1 1 1 0 3 2 -0.562213 -0.575119
9 0 1 2 1 1 1 2 0 0 2 -0.265124 0.0571041
9 0 1 2 1 1 1 2 0 1 2 -0.25507 -0.306059
9 0 1 2 1 1 1 2 0 2 2 -0.250622 -0.517272
9 0 1 2 1 1 1 2 0 3 2 -0.236253 -0.617707
9 0 1 2 1 1 1 3 0 0 2 1.32059 -0.0700946
9 0 1 2 1 1 1 3 0 1 2 1.33391 -0.37716
```

Design Matrix Response Matrix

Figure 26 - Example XDR file with several design instances

The software modules implementing the analytical model will use that format to take as input the points to use for the training and to know what are the points to predict the system response, and to give as output the required prediction.

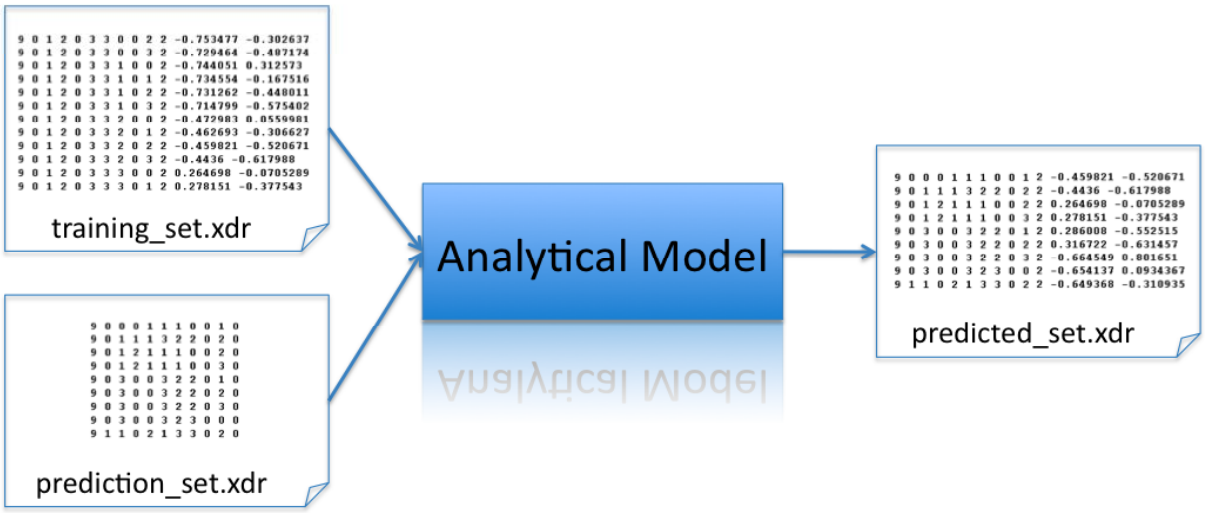

Figure 27 - Input and output files for the analytical models

Figure 27 shows graphically how the analytical model interact with the three different type of data represented by using the XDR format.

XII. Release in M3Explorer v1.0 (***NEW***)

Figure 28 presents the modular structure of the version 1.0 of M3Explorer. M3Explorer is the open source exploration framework developed within the MULTICUBE project in Task 3.1 up to M24. In the release v1.0, the exploration tool have been extended by adding the response surface modeling techniques developed in task T2.3 and presented in the previous sections.

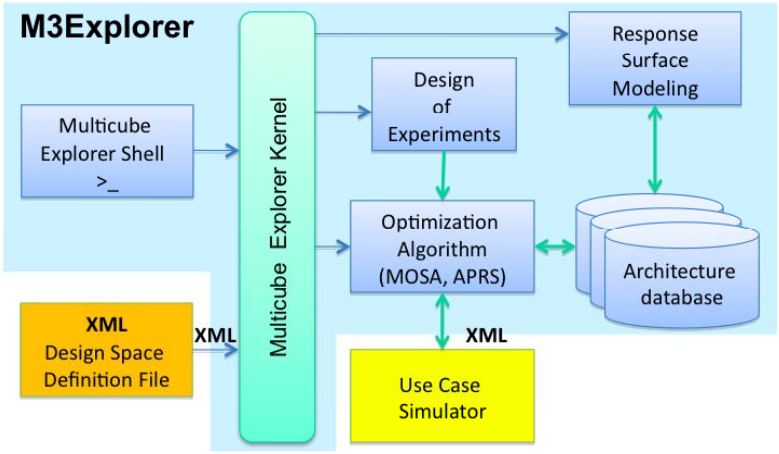

Figure 28 – M3Explorer internal structure

The joint releases of the analytical techniques together with the automatic exploration tool enabled the M3Explorer to be formally equal to the structure required by the MULTICUBE design flow presented in Figure 2.

XII.1. Website (***NEW***)

The software modules implementing the analytical techniques have been included within the public distribution of M3Explorer v1.0. To download the new package or to know more about M3Explorer, the public website can be reached by following the appropriate link on the MULTICUBE official website under the open-source tools page:

http://www.multicube.eu/open_source.html

Alternatively, the website can be reached directly with the following links:

http://home.dei.polimi.it/zaccaria/multicube_explorer

http://m3explorer.sourceforge.net

XII.2. Installation Procedures (***NEW***)

The installation procedure of the RSM modules needs to specify some additional options during the configuration phase. The list of options to be enabled for the usage of the presented analytical techniques together with the required software libraries have been described into detail in section 5 of the M3Explorer user guide downloadable from the documentation page within the MULTICUBE explorer webpage. Moreover, it is included also in the M3Explorer SW package. In the same documents for each of the analytical techniques have been described how to set the parameters related to each RSM and a simple example of usage.

XIII. Conclusions

This deliverable presented the set of analytical techniques (also called Response Surface Methods) developed in the task 2.3. The developed technique will be used in order to reduce the exploration time in the evaluation of the system metrics defined in deliverable D.1.1. Those analytical techniques will be trained through the results obtained by an initial simulation campaign and can be substituted to simulation-based system evaluation during the exploration phase.

In particular, the deliverable presents the description of both interpolative techniques (Shepard Interpolation and Radial Basis Function) and regressive techniques (Linear Regression, Artificial Neural Networks and Spline-based Regression) showing prediction results on a set of case studies coming also from the MULTICUBE use cases.

The validation results have been used to outline the characteristic of each analytical technique in case studies with different design spaces and design space size.

Moreover the deliverable describes also the integration phase of the software modules implementing the RSM techniques within the MULTICUBE Exploration framework. The interfaces used to integrate the developed RSM modules into the exploration framework and how download the exploration framework are also outlined in the deliverable.

XIV. References

[1] A. A. Jerraya and W. Wolf, Multiprocessor Systems-on-Chips, Morgan Kaufman, San Francisco, Calif, USA, 2004.

[2] Erez Perelman, Greg Hamerly, Michael Van Biesbrouck, Timothy Sherwood, and Brad Calder. Using simpoint for accurate and efficient simulation. In ACM SIGMETRICS Performance Evaluation Review, pages 318–319, 2003.

[3] T. J. Santner, Williams B., and Notz W. The Design and Analysis of Computer Experiments. Springer-Verlag, 2003.

[4] Douglas C. Montgomery. Design and Analysis of Experiments. John Wiley and Sons, 2005.

[5] P.J Joseph, Kapil Vaswani, and M.J Thazhuthaveetil. Construction and use of linear regression models for processor performance analysis. High-Performance Computer Architecture, 2006. The Twelfth International Symposium on, pages 99– 108, 2006.

[6] Karen Basso , Paulo Ricardo , De Ávila Zingano , Carla Maria , Dal Sasso Freitas. Modified Shepard Method, Investigating Alternatives for the Interpolation of Scattered Data. IEEE Symposium on Computer Graphics and Image Processing, 1999.

[7] Buhmann, Martin D. (Radial Basis Functions: Theory and Implementations, Cambridge University Press, 2003.

[8] C. Bishop. Neural Networks for Pattern Recognition. Oxford University Press, 2002.

[9] S. E. Fahlman and C. Lebiere, The Cascade-Correlation Learning Architecture, 1991.

[10] D. Culler, J. P. Singh, and A. Gupta, Parallel Computer Architecture: A Hardware/Software Approach (The Morgan Kaufmann Series in Computer Architecture and Design). Morgan Kaufmann, August 1998

[11] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. Splash-2 programs: characterization and methodological considerations. Proceedings of the 22th International Symposium on Computer Architecture, page 2436, 1995.

[12] Jose Renau, Basilio Fraguela, James Tuck, Wei Liu, Milos Prvulovic, Luis Ceze, Smruti Sarangi, Paul Sack, Karin Strauss, and Pablo Montesinos. SESC simulator, January 2005. http://sesc.sourceforge.net.

[13] S. Wilton and N. Jouppi. CACTI:An Enhanced Cache Access and Cycle Time Model. IEEE Journal of Solid-State Circuits, volume 31, pages 677–688, 1996.

[14] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In Proceedings ISCA 2000: International Symposium on Computer Architecture, pages 83–94, 2000.

[15] RFC 4506 - XDR: External Data Representation Standard. May 2006. http://www.ietf.org/rfc/rfc4506.txt

[16] Benjamin C. Lee, David M. Brooks. "A tutorial in spatial sampling and regression strategies for microarchitectural analysis." IEEE Micro Special Issue: Hot Tutorials. May/June 2007

[17] Benjamin C. Lee, David M. Brooks. "Accurate and efficient regression modeling for microarchitectural performance and power prediction." ASPLOS: 12th International

Conference on Architectural Support for Programming Languages and Operating Systems. San Jose, CA, October 2006.

[18] C. Stone. Comment: Generalized additive models. Statistical Science, 1:312–314, 1986.

[19] F. Harrell. Regression modeling strategies. Springer, New York, NY, 2001.