Java-Based Systems for Analysis and Estimation, with Application in Diagnostics of Complex Objects

Hristo Nenov

Abstract – **In this paper is described full process of design, realisation and testing of system for analysis and estimation. Like a tool for realisation of the system are used the possibilities of Java technologies, and advantages granted by it. Based on powerful mathematic apparatus, using supervised learning this system is very strong tool for observing of complex objects.**

Keywords – **System, Analysis, Estimation, Java, Object, Pattern recognition.**

I. INTRODUCTION

The concept – complex object or complex system is connected not only with their physical characteristic but with complexity of their condition recognition. Normally this is a fuzzy multitude of data and crossing classes of conditions. The systems used for solving of this type of tasks have different realizations, but common principal of using. The system described here has variants of realizations and different ways of using.

II. SYSTEM FOR ANALYSIS AND ESTIMATION

A. General Principals

Pattern in pattern recognition theory, we call the multitude of effects, united in common characteristic. Image is – every concrete effect which must be definite to one of the patterns, based on his characteristics. Follow this, it can be said, that pattern is formatted group like etalon, image is hypothetic member of a priory definite patterns [1].

After deep analysis over probabilistic and geometrical algorithms for pattern recognition, the conclusions are, that the main calculating procedure for pattern recognition described with normal distribution, using geometrical and probabilistic algorithms are common:

- calculating of vectors of mathematical meaning and co-variation matrix;
- transpose, multiplication and other manipulation over matrix;
- searching of own values and singular values;
- defining of border values, depending of chosen strategy.[1]

Like a common form for divide function can be accepted equation 1.1. First two members express square mahalanobis distance and the free member is the specific of strategy. (maximum of a posterior probability, maximum likelihood estimation MLE)(1).

$$g_i(X) = -\frac{1}{2}X^T\sum_i{}^{-1}X + (\sum_i{}^{-1}\mu_i)^T X - \frac{1}{2}\mu_i^T\sum_i{}^{-1}\mu_i - \\ -\frac{1}{2}\ln\left|\sum_i\right| + \ln P(w_i) + \ln|c_i| = X^T A_i X + a_i^T X + a_{i0} = \max \qquad (1)$$

With m classes of condition we searching maximum of dividing function for different classes (2):

$$\bar{y}_i \in X_1,\_if\_g_j(\bar{y}) = -\frac{1}{2}y^T\sum_j y + (\sum_j{}^{-1}\mu_j)^T - \\ -\frac{1}{2}\mu_j{}^T\sum_j{}^{-1}\mu_j - \frac{1}{2}\ln|\sum_j \mu_j| + \ln P(x_j) + \ln|c_j| = \qquad (2) \\ = Y^T A_j Y + a_j{}^T y + a_{j0} \Rightarrow \max$$

For visualization of classes is used principal component analyze (PCA).

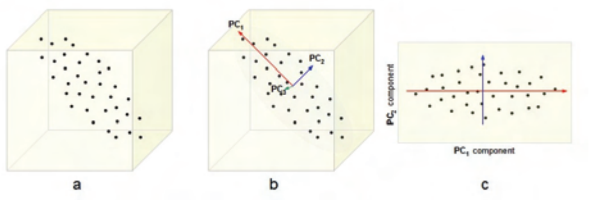

Fig.1.PCA

B. Program Realisation

Model–View–Controller (**MVC**) is a software architecture, currently considered as an architectural pattern used in software engineering. The pattern isolates "domain logic" (the application logic for the user) from input and presentation (GUI), permitting independent development, testing and maintenance of each[2].

The **model** is the domain-specific representation of the data upon which the application operates. Domain logic adds meaning to raw data (for example, calculating whether today is the user's birthday, or the totals, taxes, and shipping charges for shopping cart items). When a model changes its state, it notifies its associated views so they can be refreshed.

Many applications use a persistent storage mechanism such as a database to store data. MVC does not specifically mention the data access layer because it is understood to be underneath or encapsulated by the model. Models are not data access objects; however, in very simple apps that have little domain logic there is no real distinction to be made. Also, the ActiveRecord is an accepted design pattern which merges domain logic and data access code - a model which knows how to persist itself[2].

The **view** renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes.

The **controller** receives input and initiates a response by making calls on model objects.

An MVC application may be a collection of model/view/controller triplets, each responsible for a different UI element[2].

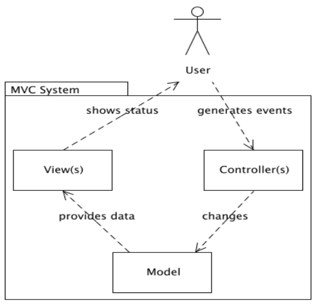

Fig.2. MVC

Model-driven architecture is focused on forward engineering, i.e. producing code from abstract, human-elaborated modelling diagrams. One of the main aims of the MDA is to separate design from architecture. As the concepts and technologies used to realize designs and the concepts and technologies used to realize architectures have changed at their own pace, decoupling them allows system developers to choose from the best and most fitting in both domains. The design addresses the functional (use case) requirements while architecture provides the infrastructure through which non-functional requirements like scalability, reliability and performance are realized. MDA envisages that the platform independent model (PIM), which represents a conceptual design realizing the functional requirements, will survive changes in realization technologies and software architectures.

Of particular importance to model-driven architecture is the notion of model transformation.

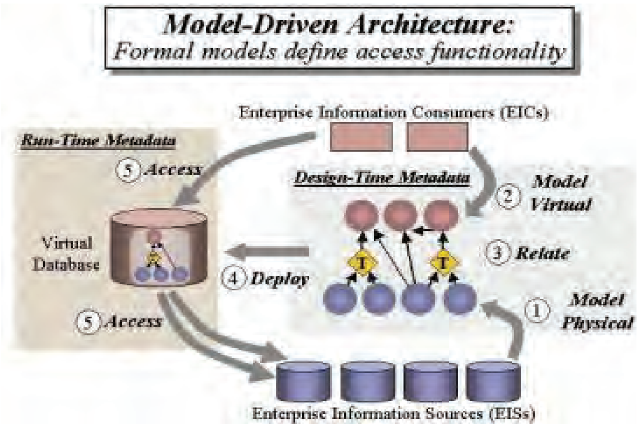
Model-Driven Architecture:
Formal models define access functionality
Enterprise Information Consumers (EICs)
Run-Time Metadata
5 Access
Design-Time Metadata
2 Model Virtual
Virtual Database
3 Relate
4 Deploy
5 Access
1 Model Physical
Enterprise Information Sources (EISs)

Fig.2.MDA

There are three variants of the system:

1.Desktop application based on PC

2.Client- server approach with connection with enterprise database.

3.Client- server approach, client part for mobile devises.

The cross platform (operating system independences) character of Java granted possibility of variants for realization and using of system.

III. System testing, with diagnostic of main ship engine MAK, from series M32C and M43C

The data for analysis is based on ship engine log-book. A priority are chosen twelve parameters which are marked like most informative (Z, RPM, Pz, Pc, λ, Ps, ε, Tbg°C, Ts, $\Delta P_{T.K.}$, ΔT^{air}_{cooler}, Cu)[1]. Thirteen classes are defined:

- Good working engine – class 1
- Heavy threadbare piston ring – class 2
- Dirty air filter – class 3
- Precipitation over turbine circle- class 4
- Threadbare fuel injector – class 5
- Threadbare nozzle – class 6
- Early fuel injection – class 7
- Late fuel injection – class 8
- Broken piston ring – class 9
- Air between piston and cylinder bush – class 10
- Precipitation over valves – class 11
- Heavy dirty air filter – class 12
- Dirty air cooler – class 13

The data is separated of two parts – for learning of the system and for recognition.

Results from recognition are shown in Table 1 and Fig.3.

TABLE I - CLASS RECOGNITION

	Real Classes													
Recogn. like:	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9	Class 10	Class 11	Class 12	Class 13	accuracy
Class 1	183	0	2	0	0	0	8	0	21	0	0	0	0	85.51%
Class 2	0	163	0	0	0	0	0	0	0	16	0	0	0	91.06%
Class 3	0	0	127	0	0	0	0	0	0	0	0	0	0	100.00%
Class 4	0	0	0	151	0	0	0	6	0	0	0	0	0	96.18%
Class 5	0	0	0	0	106	3	0	0	0	0	0	0	0	97.25%
Class 6	0	0	0	0	15	139	0	0	0	0	0	0	0	90.26%
Class 7	0	0	0	0	0	0	95	0	0	0	0	0	0	100.00%
Class 8	0	0	0	0	0	0	0	103	0	0	0	0	0	100.00%
Class 9	0	0	0	0	0	0	0	0	101	0	0	0	0	100.00%
Class 10	0	15	0	0	0	0	0	0	0	158	0	0	0	91.33%
Class 11	0	0	0	0	0	0	0	0	0	0	115	0	0	100.00%
Class 12	0	0	0	0	0	0	0	0	0	0	0	122	0	100.00%
Class 13	0	0	0	0	0	0	0	0	0	0	0	0	112	100.00%
Recogn.	100.00%	91.57%	98.45%	100.00%	87.60%	97.89%	92.23%	94.50%	82.79%	90.80%	100.00%	100.00%	100.00%	

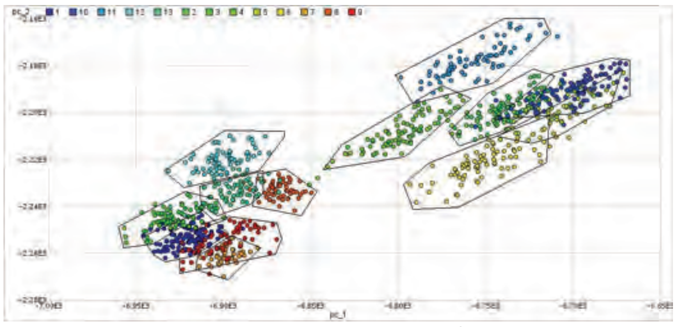

Fig.3. Principal-component analyze of condition classes

IV. Conclusion

The system is fully operational and the results from its work for this hard task are very accurate. This makes it, very powerful tool for diagnostic of complex objects.

Acknowledgement

The work presented on this paper was supported within the project BG 051PO001-3.3.04/13 of the HR Development OP of the European Social Fund 2007-20013.

References

[1] Nenov, Hr. “Algorithms and systems for estimation of condition and managing of complex objects” PhD thesis 2008.

[2] www.sun.com