Innovation in Systems for Analyze and Estimation

Hristo Nenov[1] and Geo Kunev[2]

Abstract – **In this paper are shown different principals of systems for analysis and estimation. Also is made a preview of the evolution of paradigms in programming and different classificatory and approaches for classification.**

Keywords* – *Pattern recognition, Clasificatory, DataMining, kNN, KDD, MDA.

I. Introduction

The solving of every engineer task is preceded by presence of adequate model. In real human practice this type of model often misses. Main goal of the human (abstract) thinking is to search about regularities, which will help to understand mechanism of some phenomenon. Goal of the scientific searching is to be find orders, describing not only simple objects and processes, but classes of objects and processes. Once found, the model simplified the variations of orders, formulas or structures (images, schemas) and granted possibilities for explaining and/or forecasting.

II. Modelling of Systems

A. System approach

In System approach every object (process) is "System".

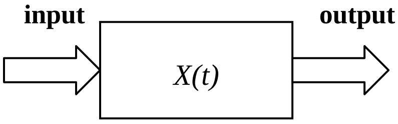
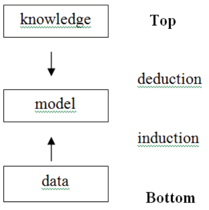


Fig.1.Managed system

Technical systems like general are managed systems (relatively closed physics systems) and can be described with terms input – output. (Fig.1)

Two main methods for system modeling are (Fig.2.):

- knowledge and understanding about system ("white box")
- experimental data for input and output ("black box")



Searching:	Knowledge based	Data based
synonyms:	top-down	bottom-up
approach:	deduction	induction
describe:	Internal structure	System behavior
method:	analyze	synthesis

Fig.1.Two main approaches for system modeling

B. Complex Systems

When we making preview /modeling of complex real systems, which normally are hierarchically, must note the fact that this kind of systems are parts (subsystems) of other systems and etc. (Fig.2).

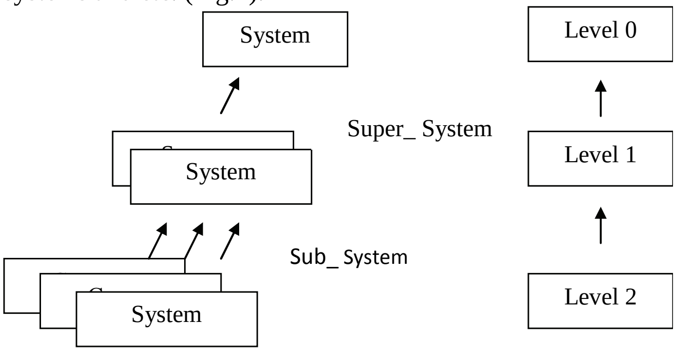


Fig.2. Complex Hierarchical System

This fact on the abstract level is the base of the object approach. It has been realized like class (object) hierarchy.

Searching of knowledge from bottom to top, based on data (induction) is most powerful tool for modeling in case of missing a prior knowledge.

"Black box" approach is inductive and including:

- collecting of (input-output) data from system
- synthesis of models
- application of modeling for describing/prediction

"White box" approach is deductive and including:

- knowing the mechanisms and behavior of the system
- possibility for deductive conclusions, based on knowledge
- base for expert systems

B. Modeling and coding

In computer programming was changed few paradigms.(Fig.3)

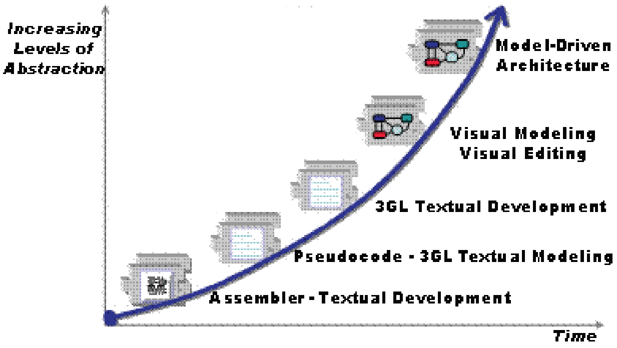

Increasing Levels of Abstraction
Model-Driven Architecture
Visual Modeling Visual Editing
3GL Textual Development
Pseudocode - 3GL Textual Modeling
Assembler - Textual Development
Time

Fig.3.Paradigms and level of abstractions

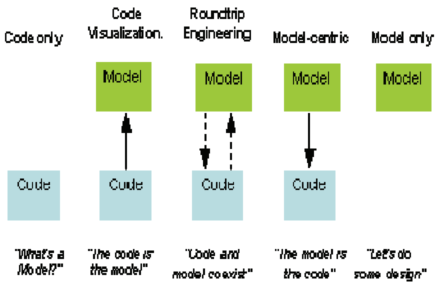


Fig.4. Modeling and Coding

Ratio coding – modeling is improving nonstop, together with evolutions of program paradigms (Fig.4.)[1].

- Current object oriented paradigm is connected with the idea that programming must be oriented not to procedures (describing computer actions) but to declaration languages (describing key abstracts of problem). Nevertheless higher level of abstraction (Fig.3.), its mass using and supporting with different programming environments is focused on the coding.[1]
- Future, model oriented paradigm supposes presence of tools for advanced describing modeling. (XML, UML), and approaches for automotive coding like MDA (Model Driven Architecture), but require considerable change in attitude.

C. KDD and Data Mining

KDD (Knowledge Discovery in Databases) is one relatively new approach for data analysis, which by definition must be *nontrivial extracting of non implicit and a priory unknown data, potentially useful, relatively simple non predefined knowledge from databases*. Knowledge extracting (Data Mining) is one of the steps in KDD technology, but they overlap each other often [1].

Data Mining methods are borrowed from statistic and including: classification, regression, association, clustering. Concept for managed (supervised) learning is borrowed from machine learning (ML), and artificial intelligence (AI).

KDD is used in finances (fraud searching, market prediction, credits), market (user "basket" analyze, target markets, sell predictions),quality control, medicine, astronomy microbiology"text mining", "web mining", "image mining", and etc. [1].

D. Unification of data in data warehouse

One "warehouse" for data (data warehouse) unifies data of different sources and integrates them in significant way. Operational data can be stored in different databases with different formats: relational, transactional, object-oriented, object-relational, active, temporary, text, multimedia, heterogenic and etc. To be unified such kind of data is not very trivial task.

Data warehouse granted data for three levels of data analyze:

- report generation
- OLAP (On Line Analytical Processing),
- KDD

Report generation is one of the functions of working databases (OLTP) and is programmed by SQL clauses, including appropriate functions.

OLAP is advanced techniques for analysis, by which the data is transformed in multidimensional arrays(hyper-cubs).

KDD is the most powerful technology, based on inductive approach, which granted model and knowledge searching in databases.

C. Trained systems and KDD

Approach which is used in KDD, for model searching in data is a case of inductive searching (learning).

There are many classification methods:

- Statistical pattern recognition:
  - Bayes rule
  - Discriminant analyze
  - KNN
  - Probability disperse estimation
  - Tree clasificators
- Neuron networks ("network approach")
- ML - methods
  - Concept training and rules
  - Case – based training
  - Genetic algorithms

E.Classificatory

Concept training.

The searched meaning (concept) c, is a Boolean function

$c: X \rightarrow \{0,1\}$ (1)

where X is multitude of units, which defined the meaning.

Training extract D is consisted from one copy x of X, and his target value $c(x)$ се състои от един екземпляр x от X, заедно с неговата целева стойност $c(x)$. Positive examples for copy are these for which $c(x) = 1$, negative $c(x) = 0$. Current D and examples of c, estimated c or searching hypothesis h in hypothesis space H like:

$(\forall x \notin X)(h(x)=c(x))$ (2)

Linear classificatory

Linear classificatory allowed, that one class can be representing with linear combination of attribute values.

In linear discriminant analysis is allowed that all $f_j(x)$ are multivariate normal density in common covariance matrix and different average vectors.

Relational training models and rules

They are based on logic from first row. The logic expression can work with digit and nondigit data. Models presented by logic languages are readable and understandable.

The following rules are possible:

- Condition rules :

if A then B

- Associative rules:

when A then B

Nonlinear classificatory

This classificatory adjust linear and nonlinear combinations of base functions to combinations of input variables. Examples for this are neuron networks, adaptive splines, nonlinear discriminant analysis and etc.

Genetic algorithms are based on evolutionally model in which survive only most fit members.

Example based methods.

They are based on parameter similarity. Most known method is k^{th} nearest neighbor (kNN).

Validation of classificatory

Based on training and test examples

Definition:

Space of unit X are define like including all metrics vectors of

$x=(x_1 \ldots x_n)$,

where x_i is metric variable(property,attribute).

Frequency and price of errors.

Price for wrong classification of class j is:

$cost(j)=\Sigma_{i=1,n} C(i|j)E(i|j)$ (3)

Where $E(i|j)$ is the count of classifications, that the class i *is classified like class j.*

Training – test paradigm

Data are separated on two (independent) multitude, L_1 – for training, and L_2.- for test.

F. Software realizations and evolutions

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].

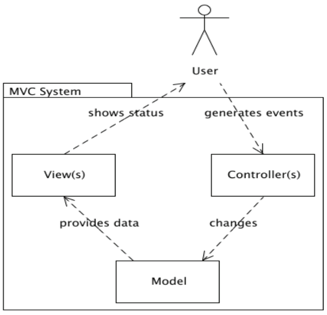


Fig.5

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.

III. Conclusion

The evolution of program paradigms is always on move. In close future maybe will be found more abstractive and more advanced tools and approaches for analysis and estimation.

References

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

[2] www.sun.com