International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

# Process Simulation Framework Design and Validation with Grinding Systems

**S. Sridevi**

***Abstract:*** ***Process Industrial & their complex control operations require comprehensive simulation software systems for modeling plant dynamics and analyzing gaps and to achieve optimal control efficiency. These models support in training plant engineers on various process scenarios in controlled pseudo real time environment. Higher degree of model designing customization, flexibility, scalability, cost efficiency and domain agnostic solution features, are the desired characteristics of any process simulation framework. This paper formulates prototype design of an integrated generic process simulator platform and its components, enabling intuitive and interactive representation of intelligent model formats, facts, knowledge, rules & behaviors. The benefits range from safer process training, analysis / synthesis of controller models; control optimization and theoretical learning. The simulation performance of proposed framework is verified through material fineness control modeling of rotary vertical grinding mill. The adaptive leaning features, with hybrid prediction model validations results in the simulation accuracy and results are compared with prevalent systems.***

***Keywords : Process Modeling, Simulators, Rule Engine, Knowledge base, Process Training***

## I. INTRODUCTION

Simulation is a constructive tool in analyzing the dynamics of any domain. The success of a process modelling and simulation (M&S) software depends on the features it provides for modelling, domain expertise of subject matter expert (SME) to integrate these features and accuracy of the resultant model to mimic target system. Steady state simulation tools are commonly available and easier to configure. Dynamic systems combined with complex equipment designs, constant technology upgrades and variable production capacities demands more flexibility and customization features of simulation platform. Scalability and generic nature of the system, play key role in near real-time process training. Scalability is achieved through expandable custom library framework and integrated common modelling constructs proposed helps the system to be domain agnostic. Key objective of this paper is to build comprehensive process modelling and simulation software framework, enabling process engineers to design, analyses and learn process behavior in a risk free environment in customized simulation workbench. The proposed adaptable features of hybrid model and platform performance are validated through a rotary grinding unit for a use case of prediction material fineness. The resultant behavior is analyzed between actual plant behavior, simulated environment and other commercial framework.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
**Dr. S. Sridevi*,** Associate Professor, Department of Computer Science India.



## II. COMPARITIVE STUDY OF RELATED WORK

Ni Li, has compared few simulation frameworks and the individual frameworks like UML & Modelica[1] and elaborated by Garson et al[2-4]. In recent years, different frameworks and tools have been leveraged for process modeling & simulation objectives, such as MATLAB, Aspen etc. These platforms have strengths in individual areas with functions such as a tool box library, data-driven analytical systems, optimization algorithm functions etc.

**TABLE 1. Comparing Simulation frameworks**

| M&S Tools | Strengths | Constraints |
|---|---|---|
| Matlab/ Modellica/ GNU Octave | Numerical computing language/ environment | Though Simulink adds graphical simulation capabilities, it lacks as expert KB systems & comprehensive process modelling |
| Modelica / Open Modelica | Discrete equations & cross- domain/ component based modelling | More of a language; no inbuilt front end / mature graphical user interface (GUI) features/ frameworks |
| UML/ SRML systems | Standardised and widely used & generic. | lacking integrated environment for simulation systems |
| LISP/ CLIPS | Engineering modelling features | Not suitable for optimization constraints resolution |
| Open source solutions : DWSIM | Suitable for educational use, No Cost, Collaborative | Ownership / support issues. Not mature / proven like commercial products |
| Commercial: Aspen+/ Unisim etc. | Scalable / configurable/ generic features / libraries | Expensive, weaker user interface (UI) & customization, non-integrated platform, |

They are not complete enough to depict complex problem with knowledge-based intelligence and mature development/run-time environment, to model end to end simulation systems and be domain agnostic training platform. The different simulation frameworks available, are compared briefly in table. 1 above. Most of the M&S systems supports either the knowledgebase development or model optimization applied to a specific domain / control objective. This paper aims at integrating the merits of the above systems as highlighted by Uraikul et al.[5-7]. who proposed generic design of comprehensive process simulation environment. The novel concept of proposed framework which enables, intelligent system behavior modelling though defined language specifications, rule engine for model interpretation, development environment/ user interfaces to design, compile & execute logic, data exchange channels among components, scalable optimization algorithms in libraries, adaptive model tuning, process training features etc.

which all operate in synergy for effective process simulation & learning. Multiple nested algorithm blocks, operates in parallel (threading) and works collaboratively to meet optimization target.

The target platform to represent process relations and critical underlying behavior trends & events, providing insights to process trainees on the system dynamics.

The analysis of proposed system performance results is validated for a rotary system. The proposed control architecture for this scenario can separately maintain production throughput & quality, with hybrid NN/MPC layers for improved accuracy.

## III. ARCHITECTURE OF SIMULATION PLATFORM

As the objective is to provide design and runtime platform for process modelling & capable of simulating different scenarios and holding near real time data, the VC++ development environment is chosen, which offers system level programming constructs. The proposed system consists of trainer application connected with multiple trainee systems in a local network or across internet as in figure.1 and provides controlled environment to get trained on different scenarios.

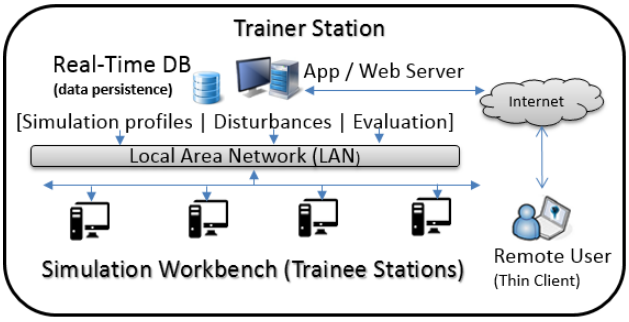


**Fig1. Process Simulator Architecture**

### 3.1 Trainer Components:

The proposed trainer station where the simulation application runs, maintains list of plant sections/scenarios to load to a specific trainee station, related to the training planned. Trainer could record / replay features of how an operator executed the control of the specific processes and grading systems to evaluate the trainees based on predefined criteria, like output produced, emissions / equipment trips controlled, critical alarms encountered and consistency etc. Trainer could introduce disturbances during a particular point of training session to check how it is handled by the operator & restore the process to steady state. The disturbance can be a simple change to a digital signal (run status of induction Fan), or a continuous ramp function over a process variable.

### 3.2 Trainee stations

Each of the trainee stations has set of components (figure 2) and connected to the trainer data server through inter process communication methods (IPC). The trainee can also be from a remote location, as a web client to trainer's web server. Seamless remote operations require reliable connectivity. These processes can be remotely started / stopped by the trainer, through named IPC methods like 'named pipe' communication.

The system to facilitate loading set of pre-defined scenarios like startup/steady state of a section of a grinding mill, or a burning kiln of a cement plant process to trainee nodes. This can be achieved by storing a specific process rule profile dump with all the process variables, stored as a snapshot in the database, which can be loaded to the data server process.

## IV. SYSTEM COMPONENTS AND INTERFACES

The design approach of a complex simulation system to be modularized and interfacing logics to be laid out clearly before constructing the same. Below sections elaborate the typical components in of envisioned comprehensive generic process simulator system, as illustrated in figure 2.

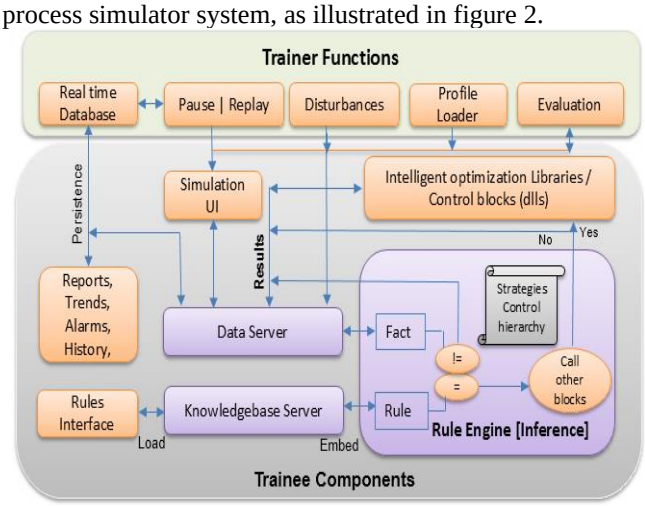


**Fig. 2. Process simulator Components and interactions**

A master spawning process initiates all these components and restarts the instance/connections in case of failover at runtime. These components running in unison, enables domain experts to design the controls, formulate and fine-tune rules of the model at design time and simulation at runtime.

### 4.1 Process Rules Configuration:

The domain expert configures the design rules required for simulating various scenarios of a production system through this interface. The rules are built using prebuilt library functions ranging from simple 'if-then' statements to complex fuzzy controllers, Math functions and prediction blocks like neural or MPC as needed to represent the model dynamics in action.

### 4.2 Knowledge Base (KB) Server:

The KB server serves as host for modelling logic in the form of computational blocks that can be independently built for specific need. The knowledge base is made up of a number of rules controlled by a rule engine. User intuitive configurators, as envisioned in below sections are planned for the design of different type of blocks.

The set of related combination of rules & blocks for a specific section of the plant (say burning operations of a 900 tons per day modern cement plant) can be grouped and serve as a prebuilt knowledge base which can be plugged into required scenarios. Such blocks are grouped to form a profile and saved in flat files for serialization / retrieval.

The knowledge base server is the running process (program in execution) that loads the specific profiles (rules/fuzzy/neural configurations persisted in files) along with related data loaded from data server process at run time and acts like a server to other client processes.

For example engine / UI (User interface) processes reads model strategy from KB server as in figure 2.
Process logic grouping is maintained in hierarchy as in figure 3, where the rules can be cascaded to call other special control blocks as put forth by Wayne[8]. Each of these blocks can be sequenced, delayed or disabled at run time using specific functions provided by KB server and gets executed by rule engine.

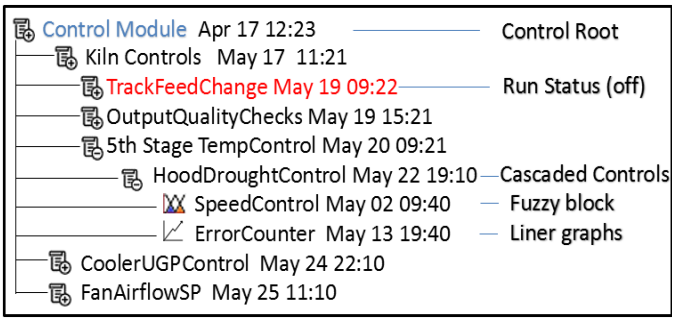

Fig. 3. Knowledgebase server - Rules hierarchy

KB provides set of in-built functions, for specific operations. For example a 'Call' function allow a strategy to run control or fuzzy blocks, 'Send' helps to communicate to other modules. Process logic block support a number of decisions making, logical and mathematical functions using predefined libraries (C++ dll - dynamic link libraries).

**4.3 Rule Engine:**

The rule strategy designed to produce the required output is executed by engine & continuous simulation is rendered at preset speed. This process parses / analyses the semantics of KB logic, interpret it based on values filled from data server. The proposed simple parser consists of a text scanning phase in principles of deterministic finite automaton (DFA) to recognize keyword / function patterns from character sequence and categorize to tokens. As detailed by Radu Vlas[9], these tokens are compared with set of allowed keywords in proposed lexical grammar and generate pre-defined actions as mapped.
In simulated environment, the pattern of the variables/value changes, to be generated by the rules written with dependency, sequencing, delays, ramps, digital/analog output generation techniques to model scenarios and react to user control actions and set points (figure 8). The whole logic hierarchy of the currently loaded profile, as per defined sequence / delays, is executed every second by a separate thread in Engine (figure3). The Rule engine is the heart of the simulation process, which executes control logic and enable the user to view the resultant process variables changes, through UI animations.

**4.4 Data Server:**

The data server holds number of plant parameters / variables of different types, and act as a soft PLC. The variables are persisted in data structures for each profile in flat files and loaded accordingly. All other simulator component processes get/set the current value of variables from data server through IPC methods.

**4.5 Real-time Database**:

To store & render simulated data history through trends/alarms/ replay functions etc. across profiles, we need robust database server. This should be capable of store/retrieve pseudo real time transactions, logged from trainee systems, as a snapshot of thousands of model variables, every second. As this needs stringent query performance/ backup strategy at DB level, SQL server 2012 was tested and used, as it is capable of addressing the needs of faster data capture (online & history) and retrieval. (figure 4).

**4.6 Knowledge Base Serialization & Execution:**

The primary task of any simulate package is to have a configurable process knowledge/ scenarios builder (figure 8), which manipulates the same process variables to generate different process conditions like plant startup, reaching steady state & shutdown etc. Such standard scenarios can be stored in files and loaded.
Trainer can save a snapshot of a user scenario & load the same at later point of time to one or more users. This is achieved through storing all the process data to database with timestamp/user details etc. and load when needed along with the profiles (scenarios) to memory.

**4.7 Replay**:

As every process/internal variable is recorded in database per second, the trainer can replay a specific period as operated by the trainee from DB and loading it to data server module. User can stop this replay at a particular event/stage and take manual control to bring the process back to stability and repeat this sequence till desired control is achieved without outliers/alarms.

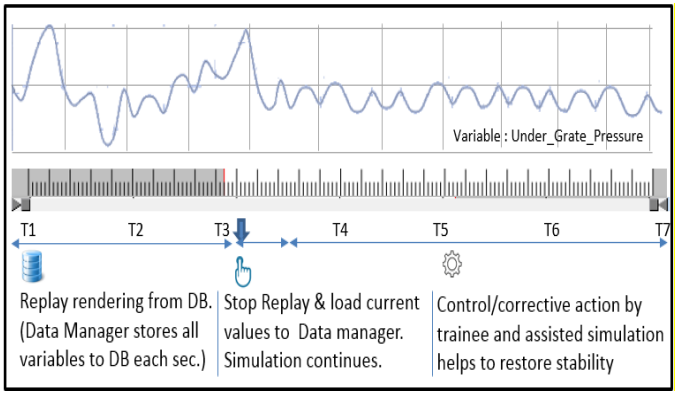

**Fig. 4: Replay function and control restoration**

**4.8 Communication Methods**

All of these components listed above are running as separate processes, in each of the trainee systems. These processes interface with each other and exchange data through different IPC (Inter process communication) methods like shared memory, named pipes, message queues, memory mapped files etc. Each component/process listed above, has a server and client component for data exchange as elaborated below.
We propose to use 'named pipe'- the 2 way data exchange IPC method, where the data is read/written character by character in a queue. Pipes can communicate between 2 running processes/ threads either in same or different computers and the latter is called 'Remote named pipes'. The server application in trainer, interface / exchange data with each of the trainee processes using this mechanism. Typically pipes are created in C/C++ like CreateNamedPipe(\\<server_name_or_ip\\pipe\\ <Pipename>). A client process connects to a server named pipe by using CreateFile function, with read or write attributes.

### 4.9 Simulation Interface [UI]

User views different sections of a simulation, in an interface which mimics a SCADA in a typical plant control room.

**Design mode**: A design/drawing interface is provided to process engineer, along with pluggable tools set, (figure 5) to create the visual model.

These tools are built through ActiveX controls, which are based on component object model (COM) & OLE functions to create animations. These control elements helps to add features like placing an image file, a motor which changing colors based on a status, a variable display text box for data monitoring, or configurable control loop etc., which when used in different combinations, depicts different scenarios. (figure 6)

The attributes (position, states and mapped variable etc.) of each control configured are serialized in standard structures. In design mode, visual simulation to be provided for tuning individual blocks for specific range of inputs, are developed.

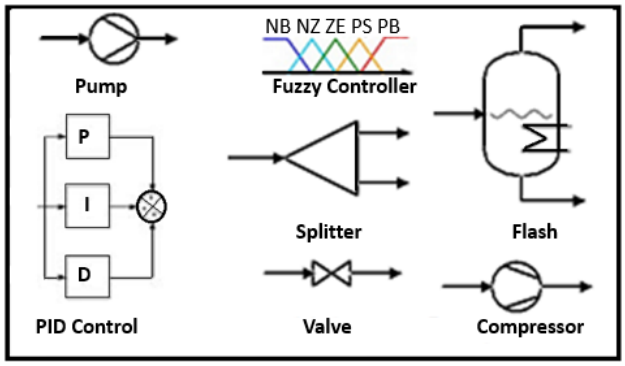


**Fig. 5. Sample Control / Tool box for model design**

**Run mode / simulation**: The simulation UI is implemented by an ActiveX container which gives an environment to host and run controls as per the attributes set for the controls.

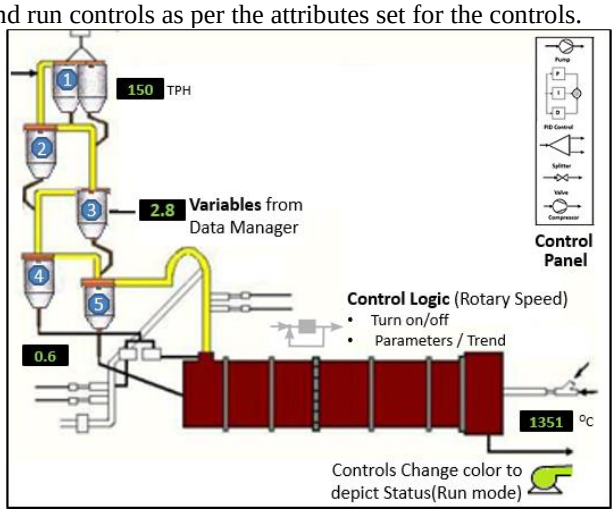


**Fig. 6. UI Building phase from control tool box**

The container framework provides support for multiple event handling during execution, interacting with in-proc server (dll) implementation of embedded objects. The ActiveX controls are packaged in an .ocx file & registered in windows registry. COM creates a unique ClasssID / GUID for each control for maintaining versions.

The container loads the controls using LoadLibrary (dll name) & uses a GetProcAddress function for dll entry point. An alternate for OCX controls is the Windows Presentation Foundation controls (WPF) in .Net or any Java equivalent.

## V. BUILDING THE PROCESS LOGIC

### 5.1 Data Server & Variables

Any large simulation logic depends on numerous set of variables with unique ids, maintained by the system. It can be either used for internal calculations or mapped to process. In control systems, there are 3 key types of process variables like controlled variable (CV): e.g. - Temperature to be maintained, Set point (SP): e.g. - Desired temperature and Manipulated variable (MV): e.g. - Coal feed rate to adjust temperature.

| S.No | ID | Variable | Type | Attribute | Length | Trending | Description |
|---|---|---|---|---|---|---|---|
| 1 | 35 | ElecEarArray | Internal | Integer array | 4 | No | Electronic Ear Array |
| 2 | 44 | N-MotarKWArray | Internal | Float array | 120 | No | Power Array |
| 3 | 56 | RetrunFeedAvg | Internal | Float | 1 | Yes | Average Feed |
| 4 | 57 | VRMIDFanRun | Internal | Boolean | 1 | No | ID Fan Run Status |

Fig. 7. Simulation system variables & attributes

The system also needs to support different data types like integer, float, arrays and digital (Boolean) variables as shown in figure. 7. Arrays are often used to collect the process variable trend over a period of time and to take the average for smoothening, change in error and PID (Proportional–Integral–Derivative) functions as detailed by Undey[10]. These configured variables are created at run time and maintained by data server for data sharing across modules at real-time

### 5.2 Process Rule configuration:

To facilitate process engineers designing the system, a knowledgebase configuration tool is provided, to write and sequence rules. These rules are in simple English text coined through predefined syntax and keywords selectable from tool. These executable statements (rules) are put together in logically related groups to form process rule blocks. Components of a sample rule block is described in figure. 8

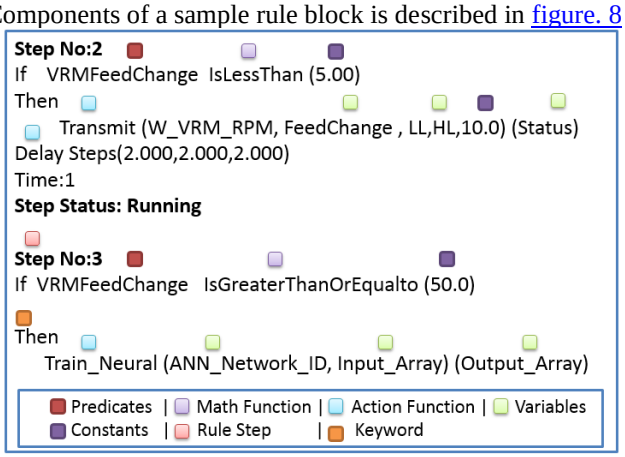


**Fig. 8. Control logic rules design with configurable functions**

The rules act as a supervisory layer over fuzzy blocks, correlation graphs or other sub-logics. Control objectives are defined in different rules and these can be executed collectively or on priorities, to model a process. A rule block is consists of a number of steps with no restriction on the step count within a rule. Each step groups a set of actions to be taken, conditionally or using other constructs. If the step is conditional, it contains a decision part and an action part. In the conditional part, the evaluation criteria is separated by connectors "And" / "Or". Non conditional steps contain only actions.

If the process logic is based on *If-then-Else constructs* only, the parser separate the predicates and fill with current values of variables from data server.

For example a predicate logic as in figure. 8 is reduced to propositions by interpreting value of the variables and branching happens as per the logic.

This is represented as, If a is constant, I and J are predicates, and x is a variable, a ∈ S then the formalization

$S_a^x(\mathrm{I(a)} \rightarrow \mathrm{J(x)})$ inferred as $\mathrm{I(a)} \rightarrow \mathrm{J(a)})$. (1)

A typical simulation model needs to represent set of continuous process state changes, as in below equations.

$$\frac{dx(t)}{dt} = \mathrm{f}\left(\mathrm{t, x(t), MV(t), d(t)}\right)$$

$$\mathrm{CV(t)} = \mathrm{g\ (t,\ x(t),\ MV(t),\ d(t))}$$

Where x is the state variable vector, d is disturbance vector. Process designers to first come up with mathematical model / expected responses of process being simulated.

For example, consider a material grinding unit's non-linear mass balance is modelled like below:

$$Tf\frac{dXf}{dt} = -Xf + \left(1 - \rho(s)\right)\beta(l,h) \quad (2)$$

$$Tr\frac{dXr}{dt} = -Xr + \left(\rho(s)\right)\beta(l,h) \quad (3)$$

$$\frac{dz}{dt} = \mathrm{Xr} - \beta(l,h) + v \quad (4)$$

$$\beta(l,h) = 20l\, e^{\left(-\frac{dl}{80}\right)} \quad (5)$$

$$\rho(s) = 9.3\left(\frac{s}{smax}\right)^2 - 12.9\left(\frac{s}{smax}\right)^3 + 5.1\left(\frac{s}{smax}\right)^4 \quad (6)$$

MV = $s$ : fine material separator rpm (speed); $v$ : Input feed ;

CV = $l$ : current load in mill ; h : material thickness ;

$Xr$ & Xf : reject / Input flow rate ; smax=250 ;

$Tr$ & $Tf$: reject/input time constants;

$\beta$ & $\rho$ : grinding and seperation functions ;

**Design time:** Each of the input/output/intermediate variables are configured in data server and also placed in UI interface over a grinding unit diagram (figure.12). Each of the above equations are written as rules with embedded math functions like differentiation, exponentiation etc., from math library or other blocks in 5.3, using the variables.

**Run time:** The rule engine, gets entire rule set from KB server, interprets these state models line by line, get/set current values of variables from data server, calls multi-level functions/libraries to process the logic and return results to the user through UI, every second. With the mass balance rules executed, user can view changes to mill load, inflow, outflow, and reject visually. The impact of MV on CV along with other process variables over time, can be configured/viewed in trends. (fig.4), by fetching historic values from database.

### 5.3 Computational block Libraries

Computational blocks enrich scalability of M&S framework. These pre-built, augmentable advanced control blocks perform specific functions like neural / mpc prediction etc. These parameterised configurable blocks are used by process rules as a component function, as depicted in figure. 9 below.

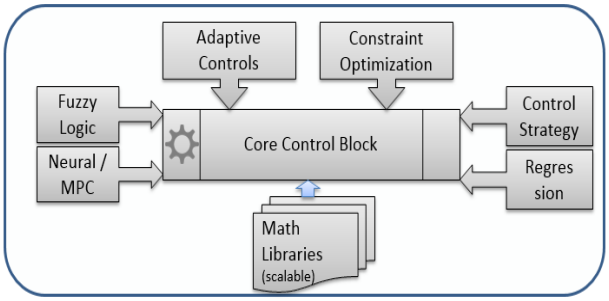


**Fig. 9 Scalable control libraries**

#### 5.3.1 Math Libraries

The math block caters to all the basic mathematical operations. A library of standard functions are provided for building process rules, which the user can use to write simple conditional statements to complex event sequencing logics.

Different functions are available to the user like Comparison (=, >, <), Mathematical (Average, Multiply etc.), Logarithmic, PID control and Array operations (store, retrieve, average etc.).

These functions are packaged in a C/C++ dll, which can be loaded at runtime by rule engine. Each function takes predefined set of inputs and gives the output which can be assigned to variable(s). For example, once parser identifies the key word as math function, say FindArrayIndex (array A, index I), it invokes the function id from math library, with the input array and index, as written by the user and returns the value at required array index. User can select these functions in process rule editor, as required & get executed by rule engine dynamically at runtime.

**5.3.2 Fuzzy Blocks Configuration:** Complex process loops, where multiple input parameters have impact on one or more output parameters are best modeled using fuzzy logic. The Fuzzy editor facilitates the building of fuzzy logic blocks for control. Any number of fuzzy blocks can be built and the first step in configuring a fuzzy block is to define the input and output parameters. For each parameter, user can define membership sets (Mamdani or Sugeno), range, shape and count of members. Fuzzy logic configurator provides the interfaces to perform heuristic modeling. Rules for each fuzzy logic block are framed as required and the fuzzy rules help to simulate the linguistic terms like 'slightly higher than' etc.

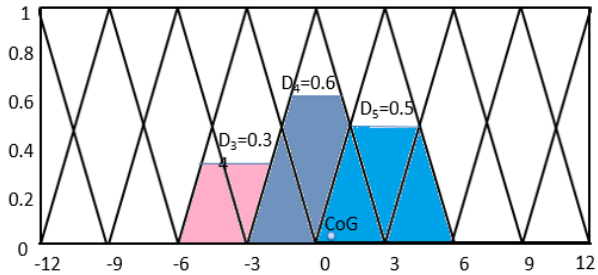


**Fig. 10. Fuzzy simulation - membership change vs output**

The process rules validate all inputs and outputs of a block before controlling the process. The input and output fuzzification/de-fuzzification can be achieved through group of membership functions and visually shown to the user, for fine-tuning as in figure. 10. The associated fuzzy rules help to simulate the linguistic terms like 'slightly higher than', 'warm liquid' etc.

**5.3.3 Linear Correlation modelling:**

The statistical relationship between 2 random variables can be represented by this tool. This can be useful to correlate single input/output system. Unlike fuzzy models, this lookup matrix is simpler tool, to enable the user to graphically represent the liner relationship between an independent & a dependent variable and derive the relationship.

**5.3.4 Neural Network (NN) prediction**

Neural networks adds adaptability and self-tuning to the knowledge base, depending on the environment of operation. Training phase of the neural network (with modified Back propagation algorithm 5.4.1) results in adapting to the relationship between input/output parameters through repeated process of adjusting weights and supervised learning.

NN configurator used to set training attributes for each problem in files, like required epochs, layers, optimization functions like Sigmoidal, Gaussian, exponential etc., which are processed by the dll at run time to produce prediction/output files.

Once the network is trained and works well for sample inputs, this block to be integrated to predict the output every second based on the current input parameter variations. Characteristics of error define the input parameter adjustments needed to achieve the desired output as suggested by Koffka[11]. The algorithm is written in a dll, which can be called in process rule block.

**5.3.5 Model Predictive Control (MPC) Integration:**

MPC works well to achieve nonlinear multivariable control objectives. MPC are useful controls over standard systems with long time constants, substantial time delays, and inverse responses, multivariate dynamic nature of the process having constraint limits (equality & non-equality constraints on CV/MV). Modre detailed survey on MPC products and their industrial application are detailed by Qin et al[12].

Optimizers adjust degrees of freedom (predicted MV changes) to achieve controlled process variable towards target trajectory by minimizing error function. This solves nonlinear programming problem at each sampling instance and the disturbance or error is estimated between actual CV from process and model prediction.

Typically 1st move from the total calculated moves of prediction horizon is implemented and next moves recalculated based on new error trend. This works on historic data of control actions and predict set of MV in target horizon as hinted by Holkar et al[13]. The dynamic model of the system is used to find the impact of future actions of the MV on the output and how the predicted error is minimized within the operating range.

**5.4 Integrated hybrid Models for simulation**

The details of individual control components are elaborated in previous sections, which users can configure as a black box with suitable parameters to represent a real dynamic models as in below figure.11:

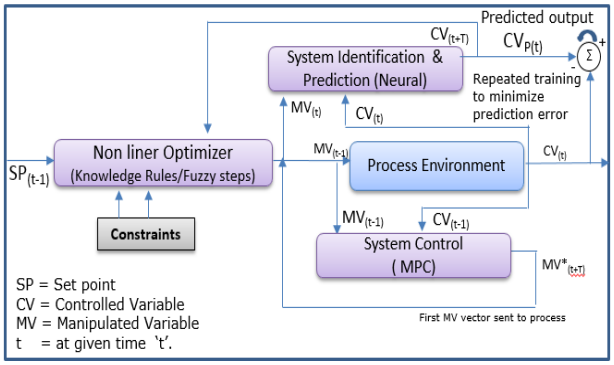


**Fig. 11. Neuro-MPC controller Schema with Fuzzy-**

Knowledge rules optimizer for simulation data generation

A typical simulation deign has a system identification phase, with continuous learning for fitting the model to process dynamics. The tuned model when looped as a predictor block, responds to input changes & the predicted output is fed to the optimizer module. Here the knowledge rules and fuzzy systems to decide MV steps required to transmit the control action back.

To overcome SME dependencies on fuzzy rules and to auto validate the suggested MV changes, a parallel MPC block is proposed to predict MV changes in a closed loop, and the most reliable changes are decided by statistical analysis, and fed to simulation environment as depicted in figure 11 & 13.

**5.4.1 System Identification:**

The C++ neural dll library implements below logic to minimise error between predicted output ($cv_p$) and corresponding historical output from plant cv(t), iterating through planned number of epochs. Thus the objective function to be minimized is given below:

$$J=\frac{1}{2}\sum_K(cv(t) - cv_p(t))^2 \quad (7)$$

The neural output represented as the function of MV & CV of previous cycles during supervised training:

$$cv_p(t) = af_N(mv(t-1), cv(t-1)) \quad (8)$$

Where $af_N()$ represents the activation function and we use sigmoidal function $\frac{1}{(1+e-x)}$ here. If the user configures faster learning rate factor, then we move to 'hard sigmoid' approximation method.

The final output prediction model thus can be represented as:

$$cv_p(t) = \sum_{i=1}^{N} W_i^o af_i (W_i\, mv(t-1) + W_i\, cv(t-1) + b_i) + b \quad (9)$$

$W_i$ is the input weight for the $i^{th}$ neuron from the input vector.

$W_i^0$ is the output layer weight coming from hidden layer's $j^{th}$ neuron. $b_i$ represents the bias (added to improve the flexibility to fix data), for the $i^{th}$ neuron from the hidden layer; b is the bias for the output layer. These weights are repeatedly adjusted in each cycle to minimise the cost function.

**Adaptability**: If the prediction range is continuously moving away from control objectives, online training is triggered automatically by the model, with recent data set collected from plant to reflect process dynamics as in figure. 13. During such triggers, a guiding strategy calls the fuzzy sets with inputs as learning rates and error / error change direction, to arrive at new learning rates, which is fed to the model parameters.

Similarly another set is configured to suggest momentum/gain factor adaptable to influence the magnitude / direction of error. These dynamic change of learning parameters aids in improving prediction accuracy.

Plant interfacing is established through communicating drivers like OPC (Object linking and embedding [OLE] for process control) or DDE (Dynamic data exchange) to get/set input / output data and to fine to the model, as in figure 13. These drivers usually interface with PLC (Programmable Logic Controller) and rendered in SCADA (Supervisory Control and Data Acquisition) systems.

**5.4.2 System Control:**

The hybrid model has a MPC layer which works in parallel to validate the Neuro-fuzzy MV predictions. MPC's ability to predict control values in future horizon is represented by below objective function:

$$J = W_{cv}^{i} \ (SP_i\text{-}CV_i)^2 \ + \ \sum_{i=1}^{N} W_{mv}^{i} \ (MV_i - MV_{(i-1)})^2 \quad (10)$$

Where $W^i$ is the weighing coefficient corresponding to MV or CV. We used 0.01 as the weightage given to MPC Cost function input. The user configures parameters like Control Horizon to confirm the number of future MV moves to be predicted (say 1), 'Dependency Size' to determine output pattern dependency on previous input patterns (say 3) and the model works on data size of 100 input/output records.

## VI. SIMULATION MODEL VALIDATION

The problem objective is to simulate controlling of a CV/outcome each second, which is otherwise measureable at higher intervals physically, and taking continuous proactive changes on inputs to maintain output stability.

The simulation model is tested for controlling a material grinding unit. The UI designed with I/O variables positioned appropriately over a process diagram (as in fig.12). The required rules/fuzzy/NN/MPC blocks configured as above. The simulation of output (fineness), in accordance with input control action steps from the model are described below.

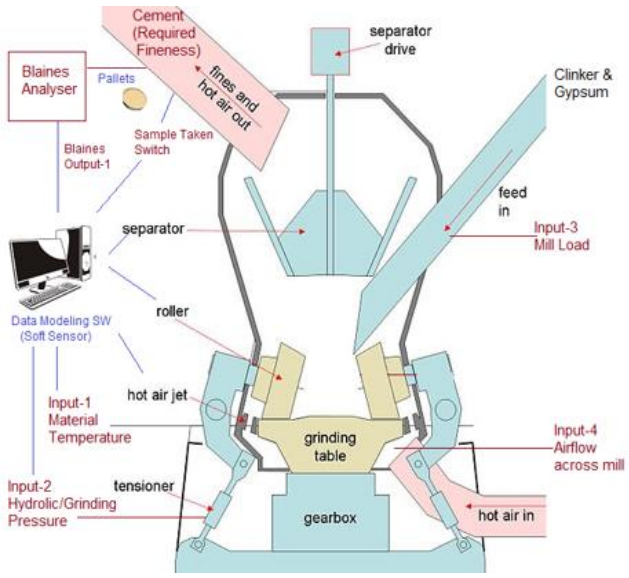


**Fig. 12. Predicting Cement fineness – input parameters**

The prediction model is applied in forecasting fineness of the material ground (output / controlled variable) in a continuous process industry, by adjusting input parameters (manipulated variables) like grinding pressure of the mill, moisture & temperature of the material being ground, load on the grinding table etc. figure 12 explains the process flow and the relation between input and output parameters.

The pattern of input parameters and corresponding output values are collected and fed to the Neuro module described above. The impact of each input parameter on the output is analyzed by principal component analysis (PCA) & most influential input parameters are fed to the training model as detailed by Russell (2001).

NN training file format and sample data collected from a grinding unit is given table.2 below:

**TABLE 2. Sample training file format with historical data**

| Manipulated Variable (MV) | | | | Control Variable (CV) |
|---|---|---|---|---|
| Input 1 | Input 2 | Input 3 | Input 4 | Output 1 |
| Feed (Tons per hour) | Pressure ($N/M^2$) | Airflow ($NM^3/s$) | Inlet-temp (C) | Fineness ($cm^2/g$) |
| 123.53 | 292.83 | 94.91 | 59.63 | 3214 |
| 123.71 | 302.74 | 112.44 | 62.17 | 3190 |
| 128.95 | 284.48 | 102.67 | 58.53 | 3322 |

### 6.1 Adaptive Training of prediction framework

Soft computing models such as Neural / MPC enhances non liner multivariate, probabilistic data and act as adaptive optimization tools for learning, predicting and classifying new patterns based on historical data as tried by Perros[15]

The input & output set is given to the neural network to enable supervised learning in back propagation algorithm with adaptable learning/momentum rates as described above. Once the model is trained, the final weights are saved in weights file. Similarly the MPC training stores the resultant coefficients in respective files.

To make the control more proactive & predict like real-time, the random functions generate the MV (input) in preset ranges every second & fed to the model as envisaged by Vinay & Vincent[16-17]. The rule strategies / fuzzy blocks are configured to suggest MV changes to reduce the predicted output (CV) & the set point variance, as in below figure.

The hybrid model with components blocks working in synergy, uses the trained MPC coefficients to suggest Next MV change in parallel. The output from Neuro-fuzzy and MPC predictions are compared through statistical analysis and valid changes are fed as environment inputs. These MV/input changes like mill feed-rates, grinding pressure, etc., are adjusted to maintain constant specific set point (CV). As a closed loop control system, the output error drives the input changes direction and magnitude for reaching process stability.

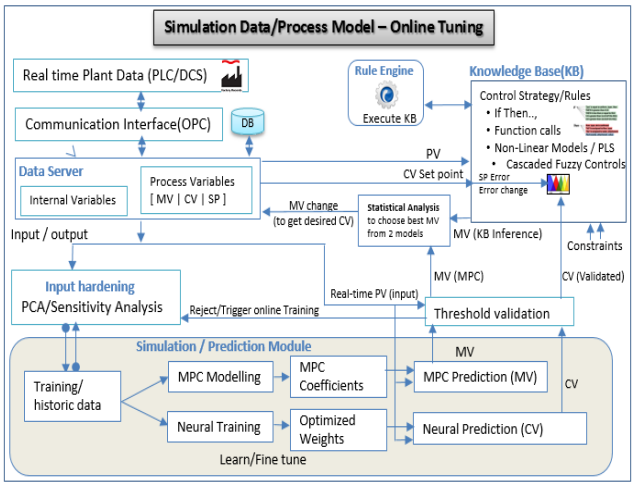


**Fig. 13. Online simulation model tuning**

The inputs/outputs are also fed to *sensitivity analysis* with major influencing variables to compare whether the predicted output has any major variance and to ensure the results are in the agreeable range.

As described in section 5.4.1 and in above figure, with the online training through plant communication interface and all the above validation process described, the model stays closer to real world processes with improved simulation accuracy.

### 6.2 Result Analysis:

The above grinding operations simulation unit helps trainee to observe the model and take appropriate control on MV, and to bring the CV closer to set point. The trainer could record/replay control action from different trainees, get reports/trends of critical parameters controlled and share evaluation reports. Figure.14, shows both model performance and trainee control actions during the course of simulation. The Input changes (MV) are simulated by applying ramp step-changes within the defined ranges as in the bottom 4 lines of the below trend.

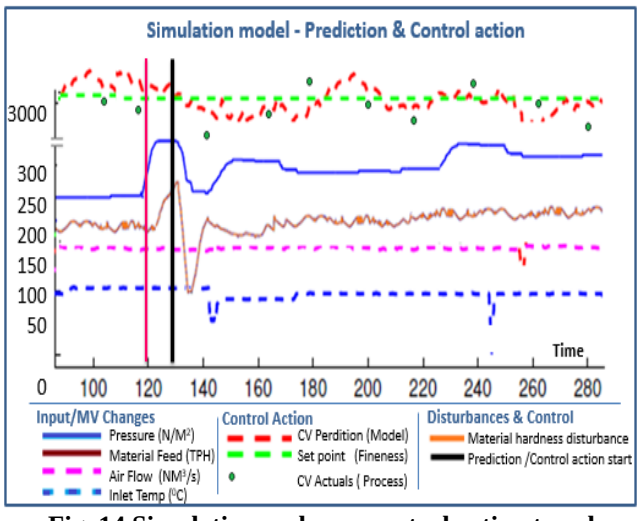


**Fig. 14 Simulation and user control action trend.**

As in figure.14, at specific intervals trainer could introduce disturbances (like change in input material moisture / hardness) by stepping the process variable, remotely around their typical values in data server, which is read by trainee process through IPC methods discussed (Orange line). The magnitude of the ramp step changes are randomly picked from the uniform distribution U (-d, d) where d is deviation. The trainee gets visualization on all impact to all related parameters & alarms and takes control to bring the process back to stability (black line). During the course of his training trainee could replay and learn in accordance to prediction system responses and guidance on MV and CV.

**Prediction model performance:**

The trained model (optimized weights & coefficients to near zero errors) could predict the output, with the simulated inputs (unsupervised). By linking the trained model to control engine, prediction process is triggered for each second. For NN/MPC training, we get optimal results for below parameters as configured in system, resulted in optimal outputs and faster convergence for the problem chosen.

**TABLE 3. Neural/MPC parameters for optimal results**

| Parameter | Neural | MPC |
|---|---|---|
| Activation function | Soft to Hard Sigmoidal | |
| Epoch | 1000 | |
| Momentum | 0.2 | |
| Hidden Layers | 3-5 | |
| Node/hidden layer | 6 | |
| Learning rate | 0.05 | |
| Error Tolerance | 0.0015 | |
| Dependency Size | | 3 |
| Weightage to cost function | | 0.01 |
| Training time (500 records) | 8 sec | 5 sec |
| Bias | | -1, 0, +1 |

With repeated adaptive tuning, the system could predict in the near real data ranges and converge faster. The exercise is done with commercial products with equivalent model setup (Neuro fuzzy & MPC blocks) and in same operating range. The results are compared against the actual results from plant. figure 15.

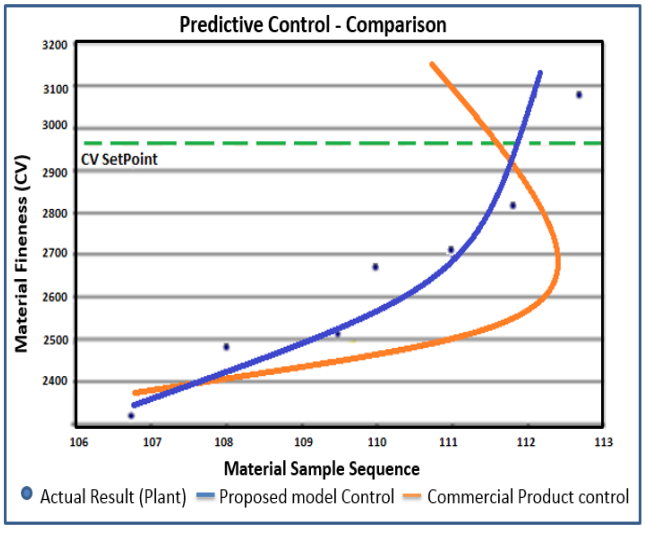


**Fig. 15 Comparison of simulation outputs**

The comparison results in table-4 below depict that the proposed closed loop model converges faster towards the set point (material fineness) to be maintained.

**TABLE 4. Prediction model results comparison**

| Prediction model accuracy | Time (t) | SP | Other Products | | | | Proposed model | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Mean | Std.Dev | Normalised RMSE | Run Time (Mean) | Mean | Std.Dev | Normalised RMSE | Run Time (Mean) |
| Input feed with Disturbances to material hardness) | 0-8 | 120 | 114.3 | 0.0032 | 1.41 e-2 | | 114 | 0.0032 | 1.41 e-2 | |
| Output prediction over PV/MV changes by user control actions | 0-1 | 2950 | 2720 | 5.42 e-4 | 9.56 e-4 | 0.65 | 2793 | 3.62 e-4 | 6.94 e-4 | 0.49 |
| | 1-3 | 2950 | 2810 | 3.42 e-4 | 6.89 e-4 | 0.55 | 2855 | 1.52 e-4 | 4.32 e-4 | 0.51 |
| | 3-5 | 2950 | 3100 | 2.42 e-4 | 5.43 e-4 | 0.45 | 2892 | 1.12 e-4 | 1.94 e-4 | 0.34 |
| | 5-8 | 2950 | 2845 | 2.52 e-4 | 5.98 e-4 | 0.41 | 2970 | 0.92 e-4 | 1.42 e-4 | 0.29 |

Though we maintain same input conditions, the resultant data sets of compared products might be in different scales, for which normalized RMSE is used. This is calculated as,

$$\text{Normalized RMSE} = \frac{1}{\mu SP}\sqrt{\frac{\sum_1^N (PV-SP)^2}{N}} \quad (11)$$

Where: the mean set point is over time, SP: the set-point to be maintained, PV: is the actual set-point through simulation, N: is the data points considered.

Several prebuilt algorithms available in commercial products and as proposed in different literatures. ANFIS (Adaptive Neuro Fuzzy Inference System) as analyzed by Bhagyamma[18] has its own limitations like need for time consuming knowledge representation & SME dependencies. NARMA, NNMPC as detailed by Kittisupakorn[19], the training effort is higher.

To overcome this, the parallel MPC validation block in action, ensures the prediction quality in given range, as over and above Neuro-fuzzy suggested MV steps. Faster convergence is achieved through adaptive learning and momentum rates (5.4.1). These help in achieving global asymptotical stable (GAS) and improves prediction accuracy ~90% to near actual values. The momentum factor chosen helps to avoid early local minimum and achieve GAS of this closed loop system. This factor also determines the impact on weights from previous pass to current cycle. The PCA and sensitivity analysis decide thresholds for model input/output parameters and decides on adaptive training needs, which in turn improves accuracy.

## VII. CONCLUSION AND FUTURE RESEARCH

This concept paper proposes a comprehensive, integrated modelling and simulation framework for process training. The design aspects of its components / interfaces are elaborated along with compilation & execution approaches. The proposed system equips model designer to represent real-time system behavior better, for improving process performance, proactivity, and minimise cost, risks of equipment trips & material wastage etc. Also the system is scalable & domain agnostic, through control library building blocks, distinct set of interrelated knowledge base scenarios, profiles of data and controls configurations to address training needs of different industrial process segments. With adaptability and validation features suggested to hybrid model helps to improve simulation accuracy. Fine-tuned the simulation/ knowledge rules with support from SME of cement plant grinding unit and tested with sufficient data collected. Each of the approaches discussed here, has good research potential in developing a fully functional, cost effective & multipurpose process/training simulator.

## REFERENCES

1. Ni Li, Wenqing, Minghui Sun. (2012) 'Development & application of intelligent system modeling", Elsevier, SIMPAT, Vol. 29; pp.149–162.
2. Gerson, Wai-Ming. (2001) 'Using UML semantics for modeling & beyond, Advanced Information Systems Engineering", Springer; Vol. 2068 pp.433–448.
3. C. Dimian, S.Bildea and A.Kiss. (2014) 'Integrated design and simulation of chemical process", Elsevier, second edition; pp.127-156.
4. Victorino Sanz. (2012) 'Modeling of hybrid control systems with Modelica", Control Eng. vol. 10; pp.24–34.
5. Uraikul, Chan. (2007) 'Artificial Intelligence for monitoring and supervisory control of process systems", Science Direct, EAAI, vol 20; pp.115-131.
6. Weiming shen, Daizhong Su. (2008) 'Enabling technologies for intelligent Manufacturing". Taylor & Francis, vol. 46; pp.2329-2331.
7. Rengaswamy, R.,Venkatasubramanian. (1992) 'Integrated Framework for Process Monitoring", Diagnosis IFAC, Delaware; pp.49–54.
8. Wayne.B, Bequette. (2003) 'Modeling, Design, and Simulation, Prentice Hall Professional"; 3319_341.
9. Radu Vlas. (2011) 'Rule - Based Natural Language Techniques", 44th Hawaii International Conference on System Sciences; pp.1-10.
10. Undey, Cinar, A. (2002) 'Statistical monitoring of multistage, multiphase batch processes". IEEE Control Systems Magazine, Vol 22; pp. 40–52.
11. Koffka, Ashok. (2012) 'A Comparison of algorithms for training feed forward neural networks", Intelligent Systems and Applications; pp. 23-29.
12. S. Qin and T. Badgwell. (2003) 'A survey of industrial model predictive control technology," Control Eng. Practice, vol. 11, no. 7 ; pp. pp.733–764.
13. Holkar KS and Waghmare. (2010) 'Overview of model predictive control", International Journal of Control & Automation, Vol. 3 ; pp.47-63.
14. Chiang, Russell, L.E., Braatz (2001) 'Fault Detection & Diagnosis in Industrial Systems". Springer, London; pp.35-42.
15. Harry Perros (2009) 'Computer Simulation Techniques: Definitive introduction", NC State University. Raleigh, NC; pp. 25-45.
16. Vinay Chandwani, Agarwal. (2014) 'Applications of ANN in strength of Concrete", International Journal of Current Engineering & Technology; pp.2949-2956.
17. Vincent A. Akpan, George D. Hassapis. (2011) 'Non-linear model identification", ISA Transactions, vol 50 Apr; pp.177-194.
18. B.Bhagyamma and Dr.P.Sujatha. (2015) 'Predictive Current Control Strategy Using ANFIS", IRJET, Volume- 02 Issue- 08;
19. Konakom and Kittisupakorn. (2010) 'Neural Network-based MPC Strategy", - World Congress on Engineering and computer science, Vol 2; pp.770-774.

## AUTHORS PROFILE

**Dr.S. Sridevi,** Associate Professor, Computer Science, has done her doctorate in machine learning applications in process control and has interests in image processing and data science.