Cognitive-Compatible Human-Machine Interfaces by Combining Ecological Interface Design and Object-Oriented Programming

Salaheddin Odeh

Department of Computer Engineering, Faculty of Engineering, Al-Quds University, Abu Dies, Jerusalem, Palestine

***Abstract*—Most of human-machine interfaces (HMIs) for process control disseminated in the industry are mostly technique-oriented and don't reflect operators' needs. This approach tries to compensate this lack through offering user-oriented interfaces characterized as cognitive-compatible. If HMIs have to be cognitive-compatible, then designers should take various cognitive objects such as reasoning, memory and knowledge into account. The ecological interface design (EID) offers a well-founded methodology for designing HMIs because, on the one hand, the internal physical behavior of the technical system will be exposed through the interface; and, on the other, a harmonic and an effective interaction between operators and machines will be achieved. This kind of interaction is to see as the positive result of utilizing the physical and functional characteristics of the technical system as design inputs. Additionally, EID is oriented to the three human behavior levels: the skill-, rule- and knowledge-based levels known as the human-information processing. In order to obtain cognitive-compatible HMIs constructed of EID objects, a suitable means like the object-oriented programming (OOP) in combination with a powerful development tool such as the Visual Studio .NET Integrated Development Environment (IDE) is needed. As OOP is based on nouns and reflects how the real world is perceived, it disburdens developers in translating their design ideas and models into computer applications.**

***Index Terms*—Cognitive compatibility, human-machine interfaces (HMIs), ecological interface design (EID), object-oriented programming (OOP), C# and Visual Studio .NET**

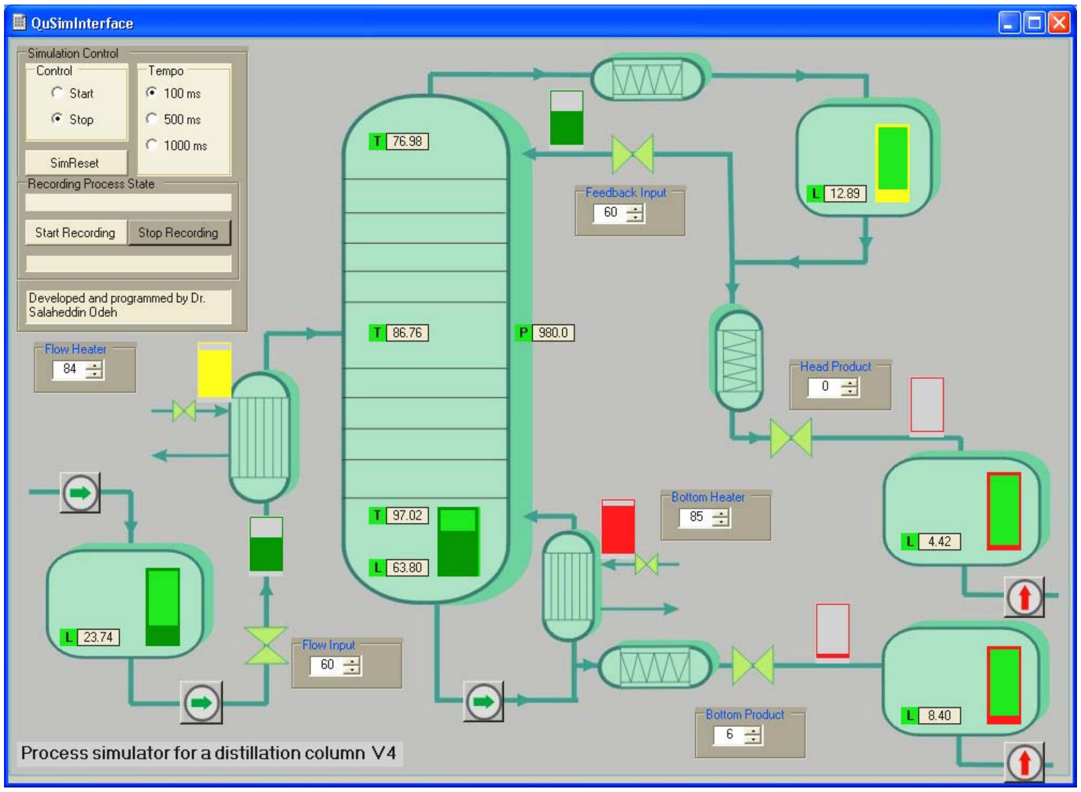
I. INTRODUCTION

DUE to the steadily growing requirements for product quality and economic viability, technical systems disseminated in the industry become complicated through strongly coupled and interconnected function modules and extensive automation equipments. The construction and the structure of technical systems, thereby, will be increasingly complicated. This complexity, which is the result of extensive strategies for process control and supervisory tasks, raises the cognitive stress of the human operators of these systems. The more adaptation of the graphical user-display mediating between the human operators and the technical system to the cognitive structures of the operators, the less cognitive stress and physical exertion will be invested by operators during their control and supervision tasks.

A. Conventional and User-Centered HMIs

Most of user-displays spreaded out in the industry are oriented towards topological or physical process views which mainly represent information about physical components and modules of technical systems [1]. This kind of user-displays guarantees an effective human-computer interaction only if regular system behaviors and situations are present, so they can easily be identified and analyzed by the human operator. If unexpected process situations and events occur, these user-displays will fail in representing and mediating the real system state that eventually leads to user failures. However, there are four critical points where user failures can occur during the interaction with an interactive software system [2]: users can form an inadequate goal; or they might not find the correct interface object because of an incomprehensible label or icon; or they may not know how to specify or execute a desired action; and may receive inappropriate or misleading feedback.

If human factors are considered in the design of both the management system and the corresponding user-interface, a user-oriented HMI is resulted, which is compatible to the cognitive structures of the operators. Such interfaces support operators in their control and supervisory tasks and take into account various human factors such cognitive elements, e.g. reasoning, memory and knowledge [3]. This contribution focuses not only on the part of the process-control management system including the different software structures, but on other aspects related to the management system such as user-centered graphics controls as well. These software modules interact with corresponding ergonomic-graphical structures displayed on the user-interface, e.g. EID graphical objects [4] and topological views [1].



Control: Start [✓ Stop]

Tempo: [✓ 100 ms] 500 ms 1000 ms

Developed and programmed by Dr. Salaheddin Odeh

Flow Heater 84

Feedback Input 60

Head Product 0

Bottom Heater 85

Flow Input 60

Bottom Product 6

T 76.98

T 86.76

P 980.0

T 97.02

L 63.80

L 12.89

L 4.42

L 23.74

L 8.40

Process simulator for a distillation column V4

Fig. 1. Graphical user-display with physical presentation of the distillation column

B. Case Study: A Distillation Column as a Complex Technical System

The technical system used here to test the design methodology and its appropriateness for complex technical systems exists in the form of a process simulator of a distillation column. The distillation column consists of different subsystems (parts) such as a column for separating a mixture, a mixture flow part, a light component and a heavy component section. This process separates an ideal mixture; and it contains several technical components such as a column, a reboiler at the bottom of the column, a condenser, a reflux drum, pumps and several control units managed by operators. Fig. 1 presents a graphical user-interface with physical presentation of the distillation column.

C. C# and the Visual Studio .NET Integrated Development Environment (IDE)

The system was programmed by means of the programming language 'C#' within the Microsoft development environment ASP .NET [5]. C# developed by Microsoft represents an event driven, object-oriented and visual programming language; and is incorporated into .NET platform characterized through its ability to allow the distribution of web-based applications on different devices and desktop computers; as well as communicating with different computer languages. To clarify the results achieved in this modest contribution, we strengthen our discussion using some concrete C# source code samples of the developed application. This will aid the reader with building an association between the developed methodology and its implementation. Since C# is based on C, C++ and Java as the most used and well-known programming languages worldwide, it will not be a hindrance for most readers in pursuing the C# code samples.

II. COGNITIVE-ERGONOMICAL CRITERIA

A. Cognitive Elements

When designing HMIs ergonomically, it must be differentiated between the sciences of perception- and cognition-ergonomics [6]. In particular, the cognition ergonomics is related to the cognitive elements: reasoning, memory and knowledge [3]. The cognition ergonomics comprehends a variety of design aspects such as the human memory storage, attention allocation, abstraction level, and information form including serial or parallel presentation; whereas the perception ergonomics concerns about designing aspects such as color, shape form, dimension and allocation; and last, but not least, highlighting. It is to note that the software-technical implementation of elements obtained by the perception- and cognition-ergonomics is known as the software- and ergonomics structures including both graphical objects placed later on the user-display and its corresponding data structures located in the management system.

B. *Cognitive Compatibility*

The cognitive compatibility plays a central role for describing an interactive computer system; however it is mainly focused on the internal model of the human using that system. Users of interactive computer system conceive the system in the form of a 'natural' model, which can serve also as a mental model [7]. An internal model includes the user's imaginations of the HMI regarding performance and the ability of processing of task objects.

Streitz [8] defines the cognitive compatibility as a general property of a HMI. A HMI is ensured to be cognitive-compatible if the number of errors occurred is minimized while interacting with the whole system, the interface demands short learn time, the tasks execution will not be hindered in occasional situations; and the interface allows the user to react quickly on signals. In our considerations, compatibility is related to the human information processing supported by the cognitive structures; and the HMI consisting of software- and ergonomics structures. A cognitive-compatible HMI is accomplished if the software- and ergonomics structures of the HMI match the cognitive elements of the users, or are equivalent to them [9].

III. Software design and system architecture

A. *Iterative Design Approach*

The implementation of the system has followed the continual refinement design process [10], in which the development procedure was broken into two parts: a preliminary and a refinement. In the former, the requirements are verified and reviewed with close cooperation with the end users, i.e. the operators. Significant changes are often made to the requirements during this phase; whereas in the latter, the implemented system can be revised and tested iteratively. In order to obtain better understanding of the final system, a modest prototype has been developed. The prototype consists of a graphical user-interface, a simple management system for receiving the process data acquired by the technical dynamic system. The received data serves as inputs to the processing components located in the management system for calculating animation values for the graphical user-interface. The processing procedure uses a database including engineers' know-how and operators' experience about the technical system. Later, it will be shown that for enabling software reusability and for being able to handle process objects of different classes in a class hierarchy generically as objects of a common base class require several object-oriented programming (OOP) techniques such as inheritance and polymorphism. Furthermore, the system is provided with exception-handling to ensure robust and fault-tolerant programs.

B. *The Unified Modeling Language (UML)*

UML (the unified modeling language), which can be seen as a diagramming notation or a visual language [11], was used for modeling the software architecture and describing the

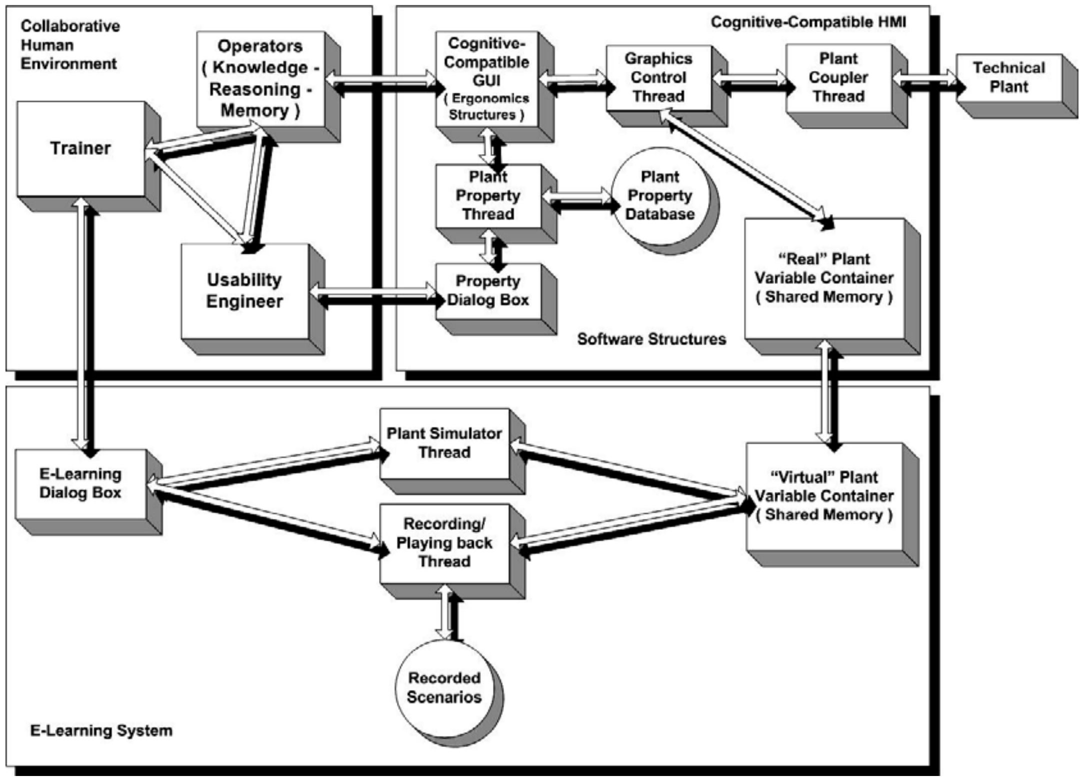


Fig. 2. A block diagram of the cognitive-compatible human-machine interface

development process. UML is a notation consisting of a set of diagrams with different elements that may be facilitated to describe the design of a software system [12]. It's not only a process or a method, but it comprises both a notation and a process. Although the UML offers several useful diagramming notations that can be very helpful during design software systems, this contribution utilizes only the class and object diagrams for modeling the objects of our interactive cognitive-compatible HMI since the design process itself will not be considered.

C. Architecture and Features of the Cognitive-Compatible HMI

In the following, we are is going to discuss the software aspects of both the management system and the graphics controls placed on the user-display, as they act on each other harmonically; and thus, can't be considered separately. The management system of the cognitive-compatible HMI includes different software- and ergonomics structures in the form of various classes for the arrangement of real process variables, graphics controls and virtual process variables. The real process variable classes are responsible for either reading or writing process values form or to the technical system; whereas, the virtual process variable classes that are mainly used for e-learning purposes, exchange information with a process simulator or receive recorded process situations by a playback component.

The graphics controls embrace and manage the views eventually visualized on the user-display. However, every metaphoric view [7] of a physical plant element or component managed by a graphics control class must have a corresponding real and virtual variable class in the management system. Properties of the user-display views reflecting characteristics of the technical system such as alerts, process variable units etc, are stored in a database which can be easily edited by means of a dialog box. Without taking the graphical user-display into account, the software part of the cognitive-compatible HMI, illustrated in Fig. 2, can be divided into two sections, one responsible for both the graphics management and obtaining values used to animate graphics on the user-display; and another for conducting the e-learning data.

Before discussing the software architecture of the cognitive-compatible HMI, it will be of great significance if we clarify some term definitions related to HMIs. This will not only simplify distinguishing between the terms used in this contribution, but it will mediate the concept as well. The following discussion leans on the structure shown in Fig. 2. As shown, the architecture of the cognitive-compatible HMI includes distinctive domains: collaborative human environment [2], a graphical user-interface, an e-learning tool discussed later, a management system and the technical system itself. Every domain is modeled by various links and components with autonomous functionalities.

In the collaborative environment, it is intended to integrate technology and user-centered aspects by taking interdisciplinary approach. The interaction among different types of users such as operators, engineers and trainers, their characteristics and how this interaction can be embedded together aimed at training, supporting and enhancing collaboration is a key part of this research and further investigations. Trainers, engineers and experienced operators collaborate also in building and maintaining the e-learning tool, which can be used for training novices and refreshing knowledge of experienced operators.

Operators realize the graphical display, as part of the user-interface, visually, audibly or through other sensing channels. In addition to conventional process interaction elements, it includes other graphical modules such as menus, windows, dialog boxes and form fillins. However, technical systems could not be viewed as pure dynamic processes, as they might also include the automation equipments and could be provided with advanced supporting components like expert systems [13]. For simplicity, we will be using the term plant instead of the technical system in the following.

D. Multithreaded HMI

The different program components within these parts exchange their data via shared memory regions. For achieving more processor utilization and increasing the perceived speed of the whole application, the different program components have been arranged as multithreads [14]. The management system, which can be seen as the kernel of the whole cognitive-compatible HMI, consists of a primary and a secondary part. The former one includes the plant coupler thread, graphics control thread, a plant property thread, a plant property database, and various shared memories and a dialog box, which enables usability engineers to modify the plant property database. If this database is modified, the plant property thread writes the data in the shared graphics control container. The secondary part, the e-learning system, represents an advanced feature of the interactive process control system and includes several software components such as a plant simulator thread, a recording/playing back thread, a database for recorded scenarios, a shared memory for exchanging virtual plant values, and a dialog box for controlling the e-learning sessions by the trainer.

A multithreaded HMI leads to an efficient performance in execution of the applications running on the same system. This is achieved due to the fact that one thread can compute and present the received batch of data while another is reading the next batch from the controlled technical system, process simulator or recorded situations. Another benefit acquired is the achievement of modular program structure since the arrangement of programs in threads leads to simplicity in designing and implementing complex programs characterized not only by a variety of activities, but with several sources and destinations of data [15]. .NET framework class library makes concurrency primitives available to the application programmers, i.e. the programmer specifies the program portions that will be containing "threads of execution"; and then these threads proceed together concurrently.

Apart from the plant property thread, all other threads are managed asynchronously and treated in an alternating producer/consumer relationship. For instance, the plant coupler and graphics control threads are scheduled by a timer whose clock interval depends on the sample rate of the controlled plant. Interrupts inform the plant property thread as soon as the database for plant properties has been modified. The changed data will be written into the shared memory of the graphics control container, which will eventually be viewed on the graphical user-display. Visual Studio .NET (C#) enables programmers to create applications with producer/consumer relationship using the namespace *System.Thread*. Several methods and static classes that are available in this namespace can be used to schedule multithreaded applications. This enables mutual exclusion of concurrent threads such as *Thread.Sleep()*, *Thread.Pulse()*, *Monitor.Enter()*, *Monitor.Wait()* etc. Relational database programming is fully supported in .NET through Microsoft ActiveX Data Objects (ADO), ADO .NET, which disburdens programmer by their task of connecting, interacting and manipulating database tasks [16]. In the following, we are going to discuss some aspects of the interface design methodology, EID, adopted in this approach; and whose merger with OOP eventually conducts us to cognitive-compatible HMIs.

IV. An overview of Design philosophies of HMIs

On the user-display, various graphics controls, both input and output elements, are ordered and connected together according to a specific presentation technique, e.g. topological views or ecological interface design, so that information over the technical system can be mediated transparently and efficiently. As a result, several graphics controls different in their appearance and complexity are needed. Sequentially, this is reflected on the design and programming effort made. Later it will be shown that using the object-oriented programming approach will not only minimize the programming difficulties and efforts, but it will also increase the efficiency of the final result.

As mentioned, several interaction problems between operators and conventional HMIs might occur, particularly in occasional or abnormal system situations and while repairing malfunctions. In order to bypass these problems and to minimize human failures during the task of controlling complex technical systems, other design concepts and methods of user-interfaces have to be considered. Lind [17] emphasized that the quality of the human-computer interaction can be improved if system goals, functional characteristics and means are integrated and embedded in the user-interface. Such kind of user-interfaces that visualize the above mentioned information, are designated as functional-oriented. Unfortunately, user-interfaces designed using this method are very abstract, and therefore, not suitable for our purposes of the user-centered design.

A well founded alternative for the user-interface design method is the Ecological Interface Design (EID) developed by Rasmussen and Vicente [18]. This method serves as a basis for the design of graphical user-interfaces and should improve the interaction between human operators and technical systems due to the fact that the internal physical behavior of the technical system will be revealed through the user-interface [19]. In addition, a harmonic and an effective interaction between operators and machines will be obtained as a result of taking the physical and functional characteristics of the technical system into account. Furthermore, the human information processing represented by means of the three human behavior levels, the skill-, rule- and knowledge-based levels will be modeled in the interface as it will be shown in the next section [20]. The design of the software objects both on the graphical user-display and likewise within the information management system is oriented to the idea of decoupling of mass/energy balances within the system.

This contribution discusses some fundamentals of EID both theoretically and practically, and the design steps needed to apply EID to build a user-interface for the distillation column. At the end, we will conclude with a discussion highlighting the suitability of EID for designing graphical user-interfaces for complex systems characterized by having a huge number of process variables like the distillation column which consists of more than 30 process variables. In future work, it is intended to determine the differences between the cognitive-compatible HMI based on EID; and conventional presentations such as the topological presentations by means of usability engineering tests [21]. However, an evaluation of the effect of EID on designing graphical user-interfaces demonstrated on a noncomplex technical system named DURESS (DUal REservoir System Simulation) is investigated [19], [22].

V. Human information processing

As previously mentioned, the ecological interface design in addition to the functionality of the technical systems, takes into consideration the human information processing, which is described by means of different models: skill-, rule- and knowledge-based framework as well as an abstraction hierarchy as a structure of work domain semantics and as a reflection of the mental model of operators. Moreover, it implies a decision ladder, a way to describe human decision making. All these models were developed by Rasmussen [20], [23] and [24]. The decision ladder models the human decision making, especially during error situations and describes mainly the decision making in familiar and unfamiliar situations. The ecological interface design is founded on the two other models explicitly. In order to understand the interaction between humans and information technology systems and conceive the idea and meaning behind the ecological interface design, we shall briefly discuss these two models [25].

A. *Skill-, Rule- and Knowledge-Based Behavior*

Fig. 3 shows a schematic representation of the cognitive human behavior: skill-, rule- and knowledge-based framework while interacting with interactive technical systems. This model is formalized by means of analysis of operators' behavior in error situations during their supervision and control of technical systems. Each level, which describes a different behavior type, is stimulated by different types of information: signals, signs and symbols. The skill-based behavior occurs when human operators interact unconscientiously with technical system, and therefore, this interaction is called an automated form of the human information processing. If operators behave according to the rule-based level, their interaction is not only dominated with rules of the form "if-then", but known behavior patterns as well; whereas the knowledge-based behavior occurs only in new situations.

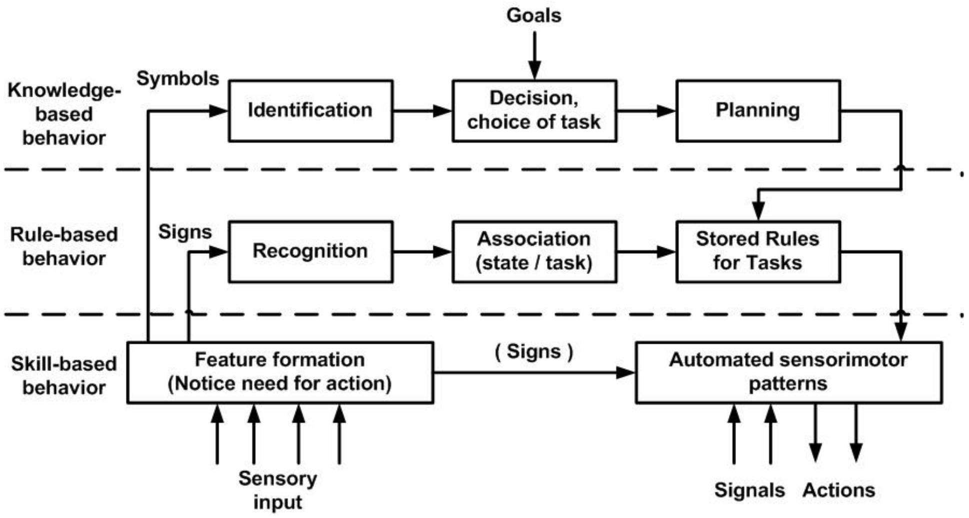


Fig. 3. Schematic representation of the cognitive human behavior: Skill-, Rule- and Knowledge-based framework [20]

B. *Part-Whole/Means-End Space*

The part-whole/means-end space or abstraction hierarchy, illustrated in Fig. 4, contains a framework of work domain semantics and reflects the mental model of operators regarding understanding and conceiving the operations for controlling and supervisioning of the technical system. The abstraction hierarchy space consists of two dimensions: part-whole of the technical system and means-end levels for the functionality of the technical system, indicating distinctive decomposition degrees.

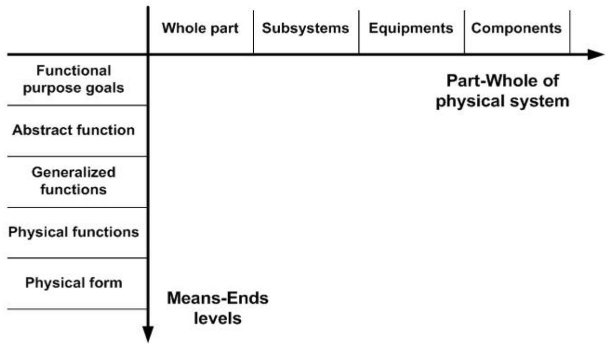


Fig. 4. Abstraction hierarchy [24]

The means-end/part-whole space serves as guideline for designers for creating user-interfaces. At a particular level within the means-end levels dimension, operators can specify WHAT should be done and, at higher level, WHAY it needs to be done. The corresponding part-whole dimension clarifies HOW the activities must be executed and serves as navigation mean within the abstraction hierarchy. Due to these facts, the user-interface with its graphical interface should give operators the opportunity to navigate themselves within the different levels. The part-whole dimension can be graphically implemented by means of zooming technique, which provides distinctive detailed information by a window hierarchy. Window hierarchies in graphical user-interfaces (GUIs) guarantee the visual momentum between the different process-views [26].

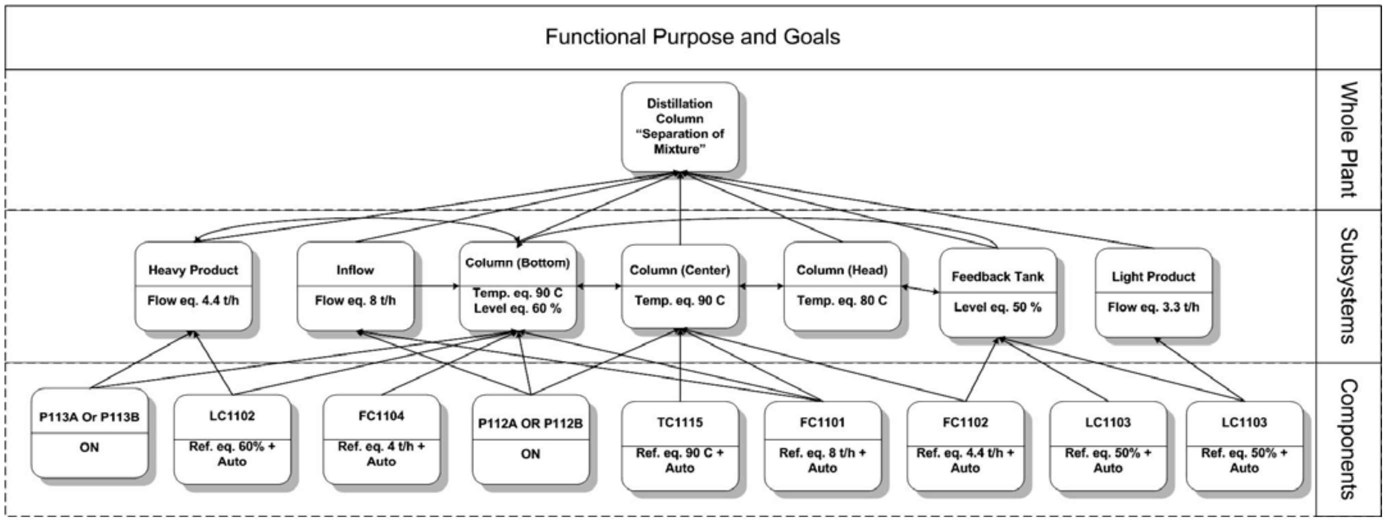

Functional Purpose and Goals
Whole Plant
Subsystems
Components
Distillation Column "Separation of Mixture"
Heavy Product Flow eq. 4.4 t/h
Inflow Flow eq. 8 t/h
Column (Bottom) Temp. eq. 90 C Level eq. 60 %
Column (Center) Temp. eq. 90 C
Column (Head) Temp. eq. 80 C
Feedback Tank Level eq. 50 %
Light Product Flow eq. 3.3 t/h
P113A Or P113B ON
LC1102 Ref. eq. 60% + Auto
FC1104 Ref. eq. 4 t/h + Auto
P112A OR P112B ON
TC1115 Ref. eq. 90 C + Auto
FC1101 Ref. eq. 8 t/h + Auto
FC1102 Ref. eq. 4.4 t/h + Auto
LC1103 Ref. eq. 50% + Auto
LC1103 Ref. eq. 50% + Auto

Fig. 5. Functional purposes and goals of the distillation columns on the part-whole dimension.

VI. Ecological interface design

A. EID Modeling of HMIs

In the previous section, we have shown how the abstraction hierarchy directs user-interface designers in understanding and analyzing technical plants, so that they have sufficient information for applying EID to design cognitive-compatible HMIs. This section will cover the three EID modeling diagrams: functional purposes and goals, abstract functions and generalized functions, although there are several models achieved by the levels of the abstraction hierarchy, part-whole/means-end space.

Fig. 5. illustrates functional purposes and goals of the distillation columns regarding various part-whole perspectives, subsystem and individual components. Every modeled part-whole element is assigned a goal: main, sub- or individual description. The links have two meanings: the first indicates the part-whole links; whereas the other means-end.

Another modeling result of the distillation column acquired by the means-end/part-whole space is the abstract functions. Fig. 6. shows the abstraction function level exposing the system dynamic from the mass- and energy viewpoint. The graphics controls used to describe the internal system behavior are energy/mass sources, sinks and storage. The links between these objects are flows of mass or energy. It should be noted that the structure in Fig. 6. is equivalent to the topological view in Fig. 1.

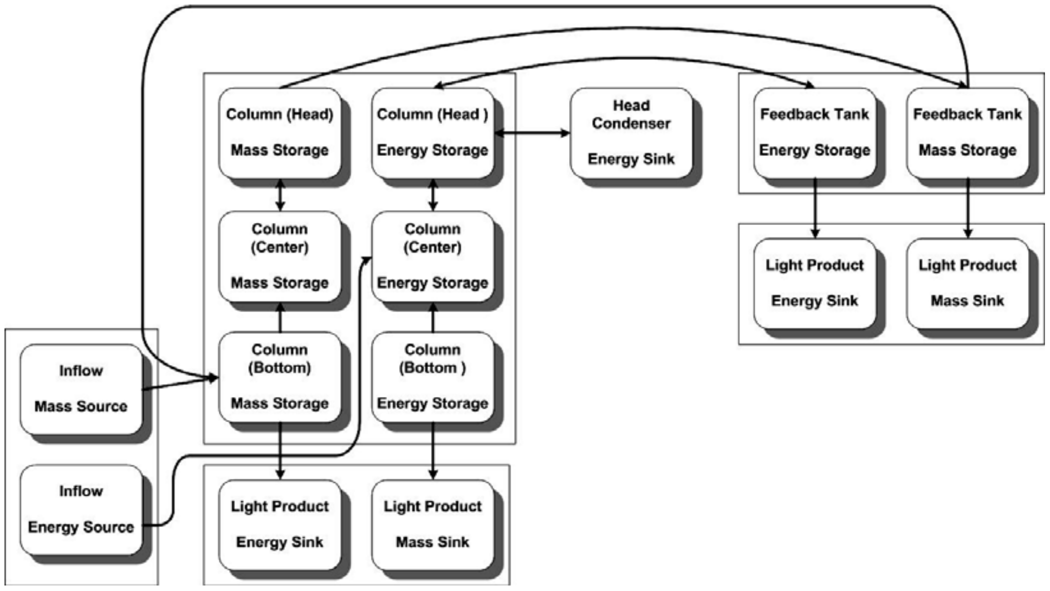

Column (Head) Mass Storage
Column (Head) Energy Storage
Head Condenser Energy Sink
Feedback Tank Energy Storage
Feedback Tank Mass Storage
Column (Center) Mass Storage
Column (Center) Energy Storage
Light Product Energy Sink
Light Product Mass Sink
Inflow Mass Source
Column (Bottom) Mass Storage
Column (Bottom) Energy Storage
Inflow Energy Source
Light Product Energy Sink
Light Product Mass Sink

Fig. 6. Simplified abstract functions representing mass- and energy relations.

The two models mentioned above offer coarse information about the distillation column and, therefore, they can be utilized in two ways. Firstly, they are beneficial for designers in analyzing and modeling the plant since they can be refined to obtain more detailed decomposition. Secondly, such models can be directly flowed in the graphical design of the user display, and thus, they serve as zooming tools for other process views provided with detailed information. If the designer begins to construct a cognitive-compatible user-interface based on EID, the generalized functions on the component level will be very helpful. These models contain more refined information of the technical system, so that that the relations between abstract objects such as mass storage, energy storage, transport elements for energy or mass, and links for mass and energy, and concrete plant objects such as pumps, heat exchangers, reservoir etc. can be established. Thus, this model determines which technical parts of the technical system are energy/mass coupled, and where to place their EID-based graphical presentation on the user-display.

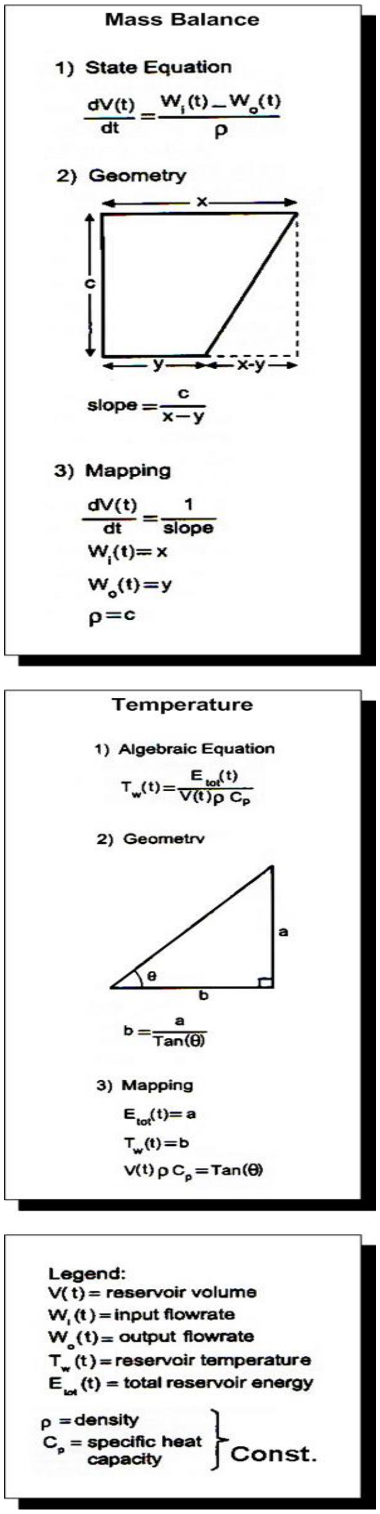

Mass Balance
1) State Equation
$\frac{dV(t)}{dt} = \frac{W_i(t) - W_o(t)}{\rho}$
2) Geometry
x
c
y
x-y
$\text{slope} = \frac{c}{x-y}$
3) Mapping
$\frac{dV(t)}{dt} = \frac{1}{\text{slope}}$
$W_i(t) = x$
$W_o(t) = y$
$\rho = c$
Temperature
1) Algebraic Equation
$T_w(t) = \frac{E_{tot}(t)}{V(t)\rho C_p}$
2) Geometry
a
θ
b
$b = \frac{a}{\text{Tan}(\theta)}$
3) Mapping
$E_{tot}(t) = a$
$T_w(t) = b$
$V(t)\rho C_p = \text{Tan}(\theta)$
Legend:
$V(t)$ = reservoir volume
$W_i(t)$ = input flowrate
$W_o(t)$ = output flowrate
$T_w(t)$ = reservoir temperature
$E_{tot}(t)$ = total reservoir energy
ρ = density
C_p = specific heat capacity
Const.

Fig. 5. Mapping between mathematical equations and geometrical shapes [27]

B. Advanced Graphical EID-Based Graphics Controls

When user-interface designers adopt the ecological interface design, they can follow these guidelines to manage their design procedure: In the first place, they should build a means-end/part-whole space containing different abstraction models as explained previously. Secondly, the design of graphic elements has to be oriented to natural objects existing in the real environment [4]. Finally, operators should be directly allowed to manipulate the technical system without hindrance caused by the graphical user-display. Vicente implemented the second guideline by mapping formalized mass/energy equations to geometrical shapes. To clarify the idea behind mapping mathematical equations to geometrical shapes, consider the following simple physical system shown in Fig. 7, which describes mass/energy balance and temperature relations. The management system of the HMI delivers calculated data used to animate mapped shapes on the user-display. The calculation of the animation data is carried out according to equations illustrated in Fig. 7.

Integrating the two shapes to a new object, leads us to the advanced graphics control shown in Fig. 8, which is not only couples the mass/energy balances and the temperature graphically, but it also predicts changes of process variables. This graphics control is cognitive-compatible since it fulfills both various requirements discussed in section II: "Cognitive-ergonomical criteria"; and it enables a transparent process control and supervision. Furthermore, this kind of graphical presentation has additional advantages. Firstly, it includes the desired reference value of the controlled process variable, e.g. the temperature as depicted in Fig. 8. Secondly, differences between the input-output flow of the energy or mass act as graphical predictors for energy or mass balance states.

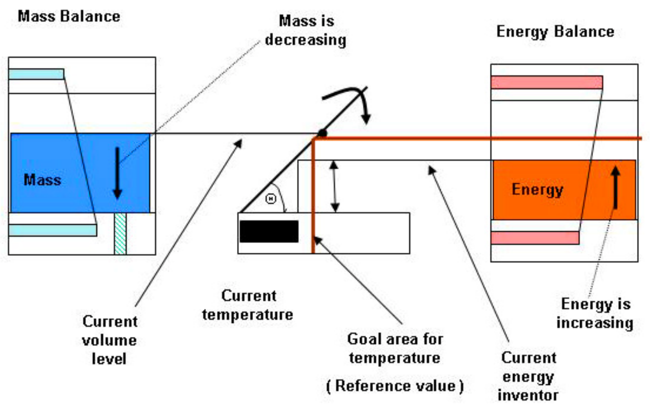


Fig. 6. Graphical mass/energy coupler

VII. EID-BASED USER-INTERFACE OF THE APPLIED TECHNICAL SYSTEM

This section discusses some aspects of the implemented user-interface of the distillation column. During the modeling and design of some parts of the technical system, we faced many difficulties, so we were forced to substitute their presentation with topological views. The conclusion clarifies these difficulties and problems encountered. Fig. 9 shows a cognitive-compatible presentation of the distillation column resulting from EID. This presentation exposes the dynamic of the thermal-hydraulic taking place in the bottom of the chemical plant, where the balances of the energy and mass inflows and outflows and their relations are visualized transparently.

In addition to physical (topological) views of the plant components such as valves, control units, heat exchangers etc., the topological view in Fig. 9 includes symbols through which EID views can be activated. These symbols are identified by mass/energy coupler icons and placed at locations where mass/energy relations are important. It's agreed that traditional conventional graphic elements, besides neoterical graphics controls on the user-display, have to be integrated into user-interfaces because of their importance both on the knowledge-based human behavior level and as mean to influence actuators' states of the distillation column, e.g. valves and pumps.

VIII. OBJECT-ORIENTED DESIGN APPROACH OF HMI

So far, we have discussed in depth the EID design method adopted in this research for accomplishing cognitive-compatible HMIs. For instance, the graphics control on the user-display such as EID objects represents some parts of the ergonomics structures. Operators interact directly with the ergonomics structures using the different sensing channels. In

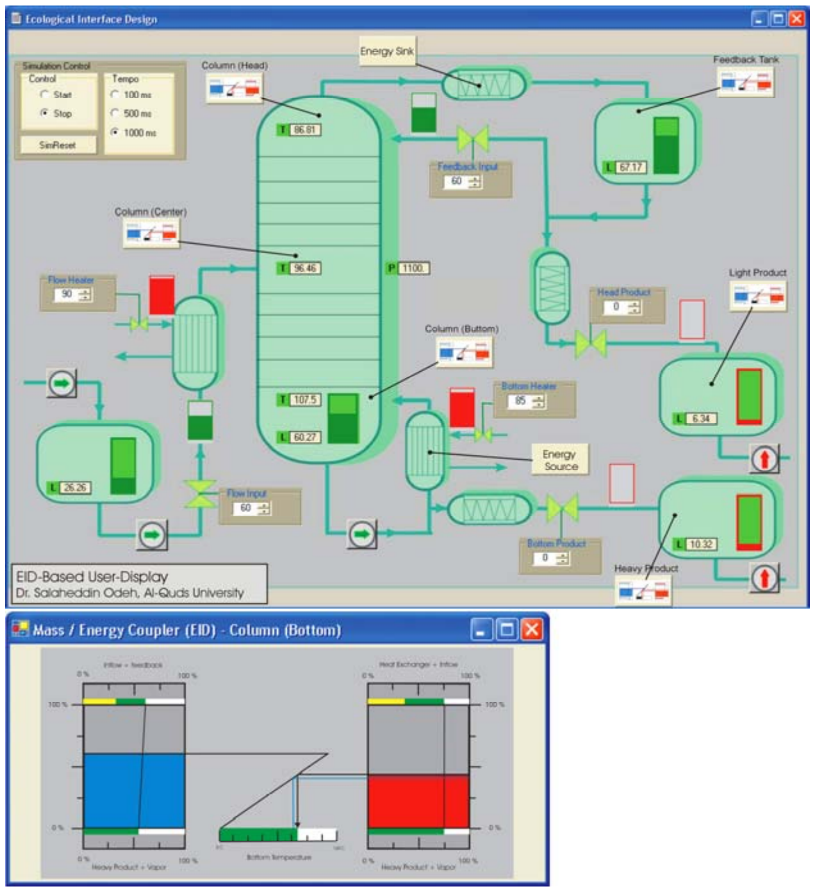


Fig. 7. EID-enhanced physical presentation of the technical system for achieving a cognitive-compatible

the following, we are going to cover the software structures. The ergonomics and software structures complement each other and, therefore, both must be thoroughly designed and properly integrated.

Fig. 10 depicts some of the graphics controls ordered in an inheritance hierarchy, which enables programmer to assign derived class objects to the base-class reference and cast explicitly between the different types in the hierarchy. All these graphics controls are derived from the base class *UserControl*, which is classified as an "abstract class". As abstract classes are characterized in such away that they cannot be instantiated, they must be only used as base classes. Their definitions are not complete, so that derived classes must complete define the missing chunks; and their derived classes must override inherited abstract methods.

All graphics controls on the graphical user-display are arranged in an array containing references to those controls. Despite their type differences, the controls can be treated together because they are derived form the same base class. In OOP, this feature is called polymorphism, which makes it possible for programs to deal with diverse related classes in a generic manner. Furthermore, programs are easily extensible. In this way, several methods for either updating the control values or redrawing the graphics can be individually developed, but equally arranged and handled.

In .NET framework, there are several preprogrammed classes and methods visible for application developers. In the following section, we shall highlight some of these classes and methods related to previous discussion. One of the graphics control created is the graphics control *LevelBar*, which is derived from the .NET class *UserControl*. The level bar graphics control can be used to display physical values like temperatures or pressures. Properties values of the physical process variable visualized by graphics control *LevelBar*, e.g. the maximum and minimum magnitudes, alert boundaries coded by the colors green, yellow and red, can be also configured. .NET enables application programmers to create a verity of custom controls.

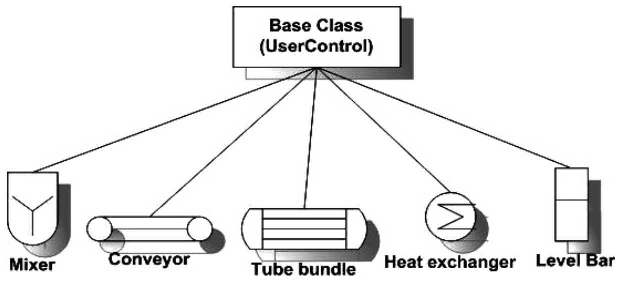
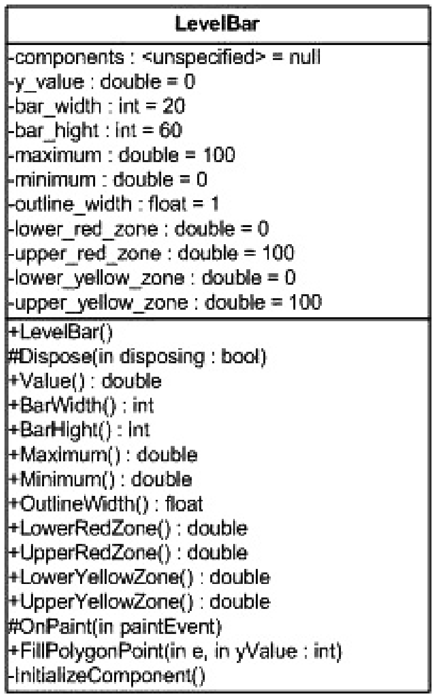


Fig. 8. Inheritance hierarchy of graphics controls

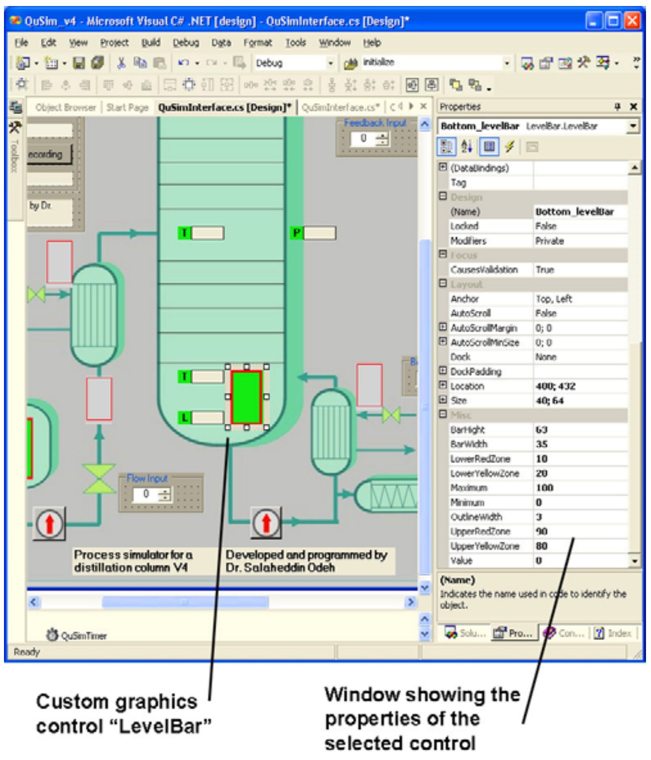
Fig. 11 illustrates the encapsulated graphics control *LevelBar* using the UML class view. This UML class view consists of two regions, one for the class instance variables and the other for the class methods. Most of these variables are either private or protected, so that they can only be accessed by the different class methods. .NET (C#) provides the *'set'* and *'get'* accessors through which the manipulation of the class's private instance variables can be easily controlled. As an example from Fig. 11, the method *Minimum()* is only allowed to get or set the private instance variable *minimum*.



```
LevelBar
-components : <unspecified> = null
-y_value : double = 0
-bar_width : int = 20
-bar_hight : int = 60
-maximum : double = 100
-minimum : double = 0
-outline_width : float = 1
-lower_red_zone : double = 0
-upper_red_zone : double = 100
-lower_yellow_zone : double = 0
-upper_yellow_zone : double = 100
+LevelBar()
#Dispose(in disposing : bool)
+Value() : double
+BarWidth() : int
+BarHight() : int
+Maximum() : double
+Minimum() : double
+OutlineWidth() : float
+LowerRedZone() : double
+UpperRedZone() : double
+LowerYellowZone() : double
+UpperYellowZone() : double
#OnPaint(in paintEvent)
+FillPolygonPoint(in e, in yValue : int)
-InitializeComponent()
```

Fig. 9. UML class description of the graphics control 'Level Bar'

An advantage of using class accessors is to ensure that writing or reading of new value is appropriate for the data of the private instance variable, e.g. variable interval. Using the *'set'* and *'get'* accessors for handling private member variables does not only make the .NET environment attractive from software engineering view point, but also it allows the graphical direct manipulation of these data through a dialog form. In the Visual Studio .NET Integrated Development Environment (IDE), the visualized data are called *properties* and, if desired, they can be viewed and changed through a dialog form called "properties window". Fig. 12 presents a snapshot of IDE while editing the properties of our custom graphics control *LevelBar*. It is to note, that the properties' names, e.g. *Minimum* and *Maximum*, are corresponding to the methods *Minimum()* and *Maximum()* illustrated in Fig. 11. The graphics control container is implemented as an array of the base class *UserControl*. In order to allow array style access to the data of classes, we can provide these classes with indexers. If we use indexers, we can use the bracket ([]) notation, as with arrays. Consequently, graphics controls of different types can be treated polymorphically, such as the case with the method *OnPaint()* for repainting the graphics and *Update()* for updating the values.


QuSim_v4 - Microsoft Visual C# .NET [design] - QuSimInterface.cs [Design]*
File Edit View Project Build Debug Data Format Tools Window Help
Object Browser | Start Page | QuSimInterface.cs [Design]* | QuSimInterface.cs*
Feedback Input
Flow Input
Process simulator for a distillation column V4
Developed and programmed by Dr. Salaheddin Odeh
QuSimTimer
Ready
Custom graphics control "LevelBar"
Window showing the properties of the selected control

Properties

Bottom_levelBar LevelBar.LevelBar

(DataBindings)	
Tag	
Design	
(Name)	Bottom_levelBar
Locked	False
Modifiers	Private
Focus	
CausesValidation	True
Layout	
Anchor	Top, Left
AutoScroll	False
AutoScrollMargin	0; 0
AutoScrollMinSize	0; 0
Dock	None
DockPadding	
Location	400; 432
Size	40; 64
Misc	
BarHight	63
BarWidth	35
LowerRedZone	10
LowerYellowZone	20
Maximum	100
Minimum	0
OutlineWidth	3
UpperRedZone	90
UpperYellowZone	80
Value	0

(Name)
Indicates the name used in code to identify the object.

Fig. 10. Properties window showing the properties for our custom graphics control 'LevelBar'

The object-oriented software design and technique of this modest investigation in cooperation with Visual Studio .NET Environment directs developers to either realize newer design philosophies like the EID, or carry out other exotic user-interface design ideas. As a result, user-interface design, which has been created with the help of this tool, is a mixed presentation of physical (topological) and EID-based views. Fig. 13 illustrates the so-called graphical mass/energy coupler forming an advanced graphics control, which integrates two simple geometrical shapes, triangle and trapezoid, to an advanced one. The trapezoid might be mapped to mass or energy balances, whereas the triangular to thermal-hydraulic equations like the physical phenomenon taking place in a liquid-filled tank with a heater. This advanced view is available as a graphics control of the class *eidView* derived from the base class *UserControl*. Such complex view demands not only comprehensive mapping from mathematical equations into geometrical shapes, but extravagant programming effort as well.

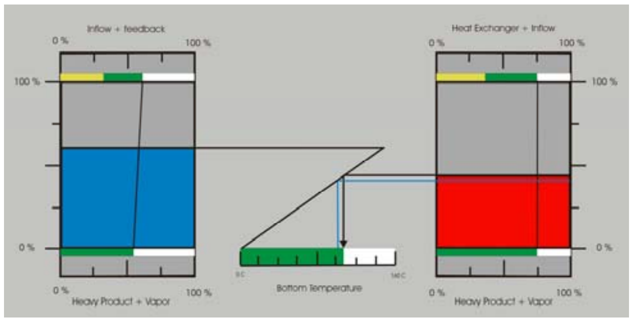


Fig. 11. Advanced graphics control implemented after the ecological interface design

IX. CONVENIENT EXPANSION TOWARDS E-LEARNING

As previously discussed, an e-learning system is integrated to the interactive process control system. E-learning has been praised by many as very useful tool for learning, and a way to minimize commuting and relevant problems. Despite the low cost possibility of e-learning against traditional learning, it might often require substantial investments in equipment. Strategically, the usage of e-learning in combination of process control systems will be profitable, since operators have the opportunity to both learn new process situations and refresh their knowledge and experience about occasional plant states. The e-learning system implemented here is mainly based on a multimedia simulation and recorded situations. However, an educational simulation can be defined as a model of some phenomenon or activity that users learn about through interaction with the simulation [28]. This means that the simulation definition embodies techniques such virtual learning and case-based scenarios. The latter is equivalent to recorded situations.

While a training session administrated by the trainer, the operators will be instructed in repairing malfunctions, so that this will be seen as they are controlling and super visioning the real plant. The e-learning container whose variables are identical to the plant variables is realized as a shared memory. It obtains its data form either the plant simulator or previously recorded situations. Software technically, the classes in this container, which is implemented as an array like the graphics control container, uses overloaded methods in order to get their data from several sources without changing the methods' names. Overloading in OOP enables using multiple methods with the same name. Overloaded methods must have different signatures, but needn't have the same return type or access level.

X. USABILITY TESTING OF THE WHOLE SYSTEM

Despite the fact that the qualitative impressions have been left while having experimented with EID to build cognitive-compatible HMIs that are should be suitable to control complex technical system, it is necessary to evaluate the appropriateness of EID as a design methodology to satisfy the cognitive compatibility property. This can be achieved by usability tests where subjects, end users, interact with the interface through solving several scenarios [21].

It is intended to evaluate the resulted cognitive-compatible HMI through a comparative usability test, through which it will be compared with another user-interface with a distinctive design philosophy such as physical (topological) interfaces. In this evaluation, the different HMIs serve as independent variables, whereas the criteria discussed in section II "Cognitive-ergonomical criteria" as dependent variables. Hereby, it will be possible to measure the differences between the HMIs regarding the cognitive compatibility property.

XI. Conclusion

It has been shown, how we can design cognitive-compatible HMIs for presenting complex technical systems using EID, a well-founded design methodology for designing user-interfaces, combined with OOP. In addition to OOP with its advanced programming features: encapsulation, inheritance, polymorphism and overloading, it was necessary to use powerful development tools like the Visual Studio .NET Integrated Development Environment (IDE) for accomplishing effective and transparent interactive systems. Using OOP and ASP .NET will not only simplify modeling the ergonomics- and software structures, but implementing them as well.

Before concluding the experiences and results achieved while investigating EID combined with OOP for designing cognitive-compatible HMIs, we should recall in mind that one goal of this investigation is to apply the approach on complex technical systems like chemical processes characterized by difficulties in measuring process variables as the sensors may be not available or are very expensive. As examples for such difficult measurable process values are concentrations and stream flow magnitudes of mixture substances existing in the different distillation column levels.

It is to note that, usually, experienced operators of such complex technical systems can deal with the control and supervision complexity of technical systems using traditional topological interfaces acquiring their animation process values by conventional sensors. However, these tasks can be achieved only with large effort, so that the operators are permanently under mental and physical exertion. Cognitive-compatible HMIs resulted through this approach must disburden the operators in stress situations where they have to repair malfunctions quickly and efficiently.

The object-oriented software design and programming makes it possible to implement powerful interactive systems for process control on the one hand; and enable system designers and user-interface designer to realize newer ideas on the other. Using Visual Studio .NET Integrated Development Environment (IDE) simplifies accomplishing the object-oriented cognitive-compatible HMI. Besides the management system with its various parts for graphics controls, real plant variables, plant properties and virtual plant variables, an investigation to applying and integrating e-learning methods and techniques to the interactive process control system has been carried out. The communication of these all parts is taking place in a multithreaded environment both with and without synchronization.

Prior sections highlighted in the first place the advantages of EID modeling and designing techniques. A discussion of the problems we came across while experiencing and applying EID for modeling complex technical systems is summarized in the following: In the first place, before creating any display element such as the graphical mass/energy coupler, EID required mathematical modeling to describe the technical system behavior. This represents a troublesome activity that is often impossible to accomplish since mathematical models of technical systems are often very complicated to be determined due to project duration and investment viewpoints. Secondly, some process variables are technically or costly immeasurable. Thirdly, delay times of influencing variables were difficult to implement technically and graphically. Finally, EID-based graphics controls demand valuable display place, so that operators are forced to carry out their tasks through several process views and user-display windows.

References

[1] "Guideline for Process Control Using Display Screens", *VDI/VDE Guideline of the Association of German Engineers*, Düsseldorf: VDI-Verlag, 2005.

[2] B. Shneiderman, and C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (4th Edition). Addison Wesley Longman, 2004.

[3] J. R. Anderson, J. R., *Cognitive psychology and its implications* (5th ed.), New York: Worth, 2000.

[4] K. Vicente, *The Human Factor: Revolutionizing the Way We Live with Technology*. Vintage, Canada, 2004.

[5] A. Duthie. *Microsoft ASP.NET Programming with Microsoft Visual C# .NET Version 2003 Step by Step*. Microsoft Press, 2003.

[6] G. Johannsen. *Mensch-Maschine-Systeme*. Berlin: Springer, 1993.

[7] N. A. Streitz, *Mental Models and Metaphors: Implications for the Design of Adaptive User-System Interfaces, Learning Issues for Intelligent Tutoring Systems*. New York: Springer-Verlag, 1988.

[8] N. A. Streitz, "Cognitive compatibility as a central issue in human-computer interaction: Theoretical framework and empirical findings", in G. Salvendy (Ed.), *Cognitive engineering in the design of human-computer interaction and expert systems*. Amsterdam: Elsevier, 1987, pp. 75-82.

[9] P. Fuchs-Frohnhofen, E. A. Hartmann, D. Brandt, D. Weydandt, "Designing human-machine interfaces to match the user's mental models," *Control Engineering Practice*, Volume 4, Number 1. Elsevier Science. 1996.

[10] T. Faison, *Component-Based Development with Visual C#*, John Wiley & Sons, 2002.

[11] S. Si Alhir, *Learning UML*, O'Reilly, 2003.

[12] C. Larman, *Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Prentice-Hall, 2004.

[13] J. E. Larsson, *Knowledge-based methods for control systems*, PhD Thesis from Lund Institute of Technology, Department of Automatic Control, Lund University, Sweden, 1992.

[14] K. Ardestani, F. C. Ferracchiati, S. Gopikrishna, T. Redkar, S. Sivakumar, T. Titus, *C# Threading Handbook*, Apress, 2004.

[15] W. Stallings, *Operating Systems: Internals and Design Principles*, Prentice Hall, 2005.

[16] J. Ferguson, B. Patterson, J. Beres, P. Boutquin, M. Gupta, *C# Bible*, John Wiley & Sons, 2002.
[17] M. Lind, "Multilevel flow modeling," *AAAI'93 Workshop on Functional Reasoning*, Washington, 1993.
[18] J. Rasmussen, K. Vicente, "Ecological interface design: Theoretical foundation," *IEEE Transactions on systems, man, cybernetics*, vol. 25, no. 4, 1995.
[19] K. Vicente, M. Bistanz, "Making the abstraction hierarchy concrete," *Human-Computer Studies*, 40, 83-117, 1994.
[20] J. Rasmussen, "Skills, rules and knowledge; signals signs, and symbols and other distinctions in human performance models," *IEEE transactions on systems, man and cybernetics*, Vol. SMC-13, No. 3, May/June 1983.
[21] M. B. Rosson, J. M. Carroll, *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*, Morgan Kaufmann Publishers, 2002.
[22] K. Christoffersen, C. N. Hunter, K. J. Vicente, "A longitudinal study of the effects of ecological interface design on skill acquisition," *Human factors*, vol. 38, pp. 523-541, 1996.
[23] J. Rasmussen, *Information processing and human interaction, an approach to cognitive engineering*, New York, 1986.
[24] K. Vicente, J. Rasmussen, *A theoretical framework for ecological interface design*. Risϕ report M-2736, Risϕ national laboratory, DK-4000 Roskilde, Denemark, August 1988.
[25] L. P. Jensen, P. Koch, "An ecological man-machine interface for temporal visualization," in: *Proceedings of the 1st international conference on intelligent user interfaces*, Florida, 1993.
[26] D. D. Woods, "Visual momentum: A concept to improve the cognitive coupling of person and computer," *Int. J. Man-Machine Studies*, Vol. 21, pp. 229-244, 1984.
[27] K. Vicente, *Cognitive Work Analysis: Toward Safe, Productive, and Healthey Computer-Based Work*, Lawrence Erlbaum Associates, New Jersy, 1999.
[28] S. M. Alessi, S. R. Trollip. *Multimedia for Learning: Methods and Development*. Allyn & Bacon, Massachusetts, 2001.

AUTHOR

Salaheddin Odeh received his master degree in electrical engineering, specialization area computer and control engineering, from the University of Stuttgart and the PhD degree from the University of Kassel; both universities are in Germany. He is currently an assistant professor in the Department of Computer Engineering, head of the Department of Computer Engineering and coordinator of the master program of the Faculty of Engineering at Al-Quds University in Jerusalem. His interests include control engineering, robotics, advanced programming, operating systems, human-computer interaction, and multimedia. In 1999 and for his doctoral thesis, he was the recipient of the first prize of the Association of German Engineers (VDI) in the state Hessen in Germany for the best technical-scientific research.

Salaheddin Odeh, Department of Computer Engineering, Faculty of Engineering, Al-Quds University.

P.O. Box 20002, Abu Dies, Jerusalem

Email: sodeh@eng.alquds.edu

Manuscript sent on October 02, 2006.