論文93-30B-11-10

휴리스틱 및 기계 학습을 응용한 엔진 모델의 보정

(ICALIB: A Heuristic and Machine Learning Approach to Engine Model Calibration)

柳 光 烈*

(Kwang Ryel Ryu)

要 約

엔진 모델을 성공적으로 사용하기 위해서는 반드시 보정을 해야 하는데, 그 보정과정은 매우 어렵고 힘든 작업을 요한다. 따라서, 모델 보정의 효율 향상을 위해 휴리스틱 및 기계학습법을 채택하여 ICALIB 이라는 지능형 보정 프로그램을 개발하였다. 이 프로그램의 사용으로, 평균 수 시간 걸리던 모델보정이 평균 수 분이내로 단축되었다. 본 논문에서는, ICALIB에 사용되는 탐색제어 기법들 중, 상태간 거리 추정 함수에 근거한 힐 클라이밍 탐색법과 변동식 허용오차 윈도우를 이용하여 점진적으로 문제 해를 개선시키는 방법 및, 탐색제어에 필요한 대상 매개변수들의 보정 순서 결정법등이 상술되어 있다. 또한, 대상 매개변수들의 순서를 정하는데 필요한 휴리스틱룰들을 자동으로 획득하기 위해 GID3 * 라는 기계학습 프로그램이 어떻게 응용되는지를 보였다.

Abstract

Calibration of engine models is a painstaking process but very important for successful application to automotive industry problems. A combined heuristic and machine learning approach has therefore been adopted to improve the efficiency of model calibration. We developed an intelligent calibration program called ICALIB. It has been used on a daily basis for engine model applications, and has reduced the time required for model calibrations from many hours to a few minutes on average. In this paper, we describe the heuristic control strategies employed in ICALIB such as a hill-climbing search based on a state distance estimation function, incremental problem solution refinement by using a dynamic tolerance window, and calibration target parameter ordering for guiding the search. In addition, we present the application of a machine learning program called GID3 * for automatic acquisition of heuristic rules for ordering target parameters.

I. Introduction

Engine modeling and simulation is becoming more and more important to the automotive industry under ever-escalating pressure to reduce automobile air pollution,

*正會員, 釜山大學校 컴퓨터工學科
(Dept. of Computer Eng., Pusan Nat'l Univ.)
接受日字 : 1993年 4月 23日

to improve fuel economy, and to reduce vehicle design-to-production time. To meet the demand, many engine component models, analytical as well as empirical, have been developed for engine concept assessment, design change evaluation, and performance problem diagnosis. As hard as modelers try, many models cannot be used directly to accurately predict engine behavior, because the knowledge about engine components being modeled is often incomplete or imperfect. To be able to use a model to simulate an engine, one has to first calibrate the model to match the engine at one or more operating conditions. This calibrated model can then be applied to simulate the engine with an acceptable accuracy. The effects of typical model calibrations can be as simple as offsetting a target parameter curve or changing the slope of a target parameter curve; or it can be as complicated as changing the shape of a target parameter curve. In this paper we use a representative model, GESIM (General Engine SIMulation program) [6], to illustrate the calibration problem.

GESIM model calibration in the past has been a very tedious manual procedure. A typical model user has to spend at least several hours to complete one GESIM calibration session. This process has to be repeated whenever a different engine is to be simulated. Furthermore, for some special applications, engine models have to be calibrated at many operating points for the need of generating an engine performance map. The need for an automated and efficient calibration program is therefore obvious. This automation, however, is a very challenging task, because (1) the calibration process searches for a goal state in a huge, continuous state space, (2) calibration is often a lengthy and frustrating task because of complicated mutual interference among the target parameters, and (3) the calibration problem is heuristic by nature, and often heuristic knowledge for constraining a search cannot be easily acquired from domain experts. To avercome these difficulties , we propose a combined heuristic and machine learning approach to acquisition and incorporation of domain knowledge into a model calibration process. We identified three important heuristic strategies that can be used to reduce search during model calibration. The first is a hill-climbing search strategy based on a heuristic state distance estimation function. The second is an incremental problem solution refinement strategy based on a dynamic tolerance window scheme. The third, and most important, strategy is the ordering of calibration target parameters based on a set of heuristic rules automatically acquired. An automated intelligent calibration program, called ICALIB, has been developed which employs all the above three strategies. The program has dramatically reduced model calibration time from several hours to only a few minutes on average.

The rest of this paper is organized as follows: Section Ⅱ formulates the model calibration problem for the ease of explaining our heuristic approach; Section Ⅲ describes the detail of the three heuristic search control strategies; Section Ⅳ presents a machine learning approach to acquiring knowledge about ordering constraints among calibration target parameters; Section Ⅴ summarizes the result and discusses directions for future work.

Ⅱ. Model Calibration Problem

In order to explore heuristic strategies for the calibration problem, we need a good understanding of the calibration problem in general as well as a proper formulation of the problem. In the following, we first give an informal account of a GESIM model calibration problem. Then, we specify the calibration problem under a state-space

search problem framework.

GESIM model calibration has been described as follows [3] :

GESIM model calibration is a process whereby certain engine measurements, termed "target values", are selected from a dynamometer test of the chosen engine or combustion system. … The process of calibration then consists of executing the model for successive runs (or iterations), each with a different setting on a selected "adjustable parameter" until the "target value" is matched satisfactorily. … This process is repeated until all of the target values are satisfactorily matched.

We formulate the calibration problem as a state space search problem to facilitate the explanation and study of the heuristic strategies that we developed. Under the state space representation, problem solving is viewed as a search through the state space to find a sequence of operators that can transform a given initial state to a particular goal state.

Let n be the number of control parameters, and m be the number of target parameters. We use C_i for $1 \leq i \leq n$ to denote a control parameter, and T_j for $1 \leq j \leq$ m to denote a target parameter. We also use D_j for $1 \leq i \leq m$ to denote a data value to be matched by the target parameter T_j, and L_i for $1 \leq i \leq m$ to denote the error tolerance value for T_i. A state s in the calibration state space is determined by the values of all the control parameters. Each target parameter is a function of C_i's, and the value of a target parameter T_j in a state s is denoted by T_j^s. A state s is called a goal state if $|T_i^s - D_i| < L_i$ for $1 \leq i \leq m$. Given an arbitrary initial state, calibration is a search process for finding a sequence of operators that can transform the initial state to a goal state where all the target values are matched satisfactorily. For GESIM model calibration, we have two operators which are $SET(C_i, x)$ and $ADJUST(C_i,T_j,x,y)$. The operator SET assigns a value x to C_i, transforming a given state into a new state such that $C_i = x$. The $ADJUST$ operator changes the value of C_i iteratively until it finds a state s such that $|T_i^s - x| < y$.

The above problem formulation helps us to appreciate the degree of complexity of the problem and to understand the necessity of a heuristic approach. In addition, the formulation makes it easier to identify a set of heuristic search control strategies for reducing the search.

Ⅲ. Heuristic Search Control in Calibration

This section explains in detail the three heuristic control strategies, as well as measures taken to overcome the inherent limitations of the hill-climbing search strategy.

1. Hill-climbing Based on Heuristic Distance Estimation

Hill-climbing is a well-known heuristic search control strategy. [9] It is also often referred to as "steepest-ascent search" or "gradient search". According to the hill-climbing strategy, the selection of a state to move to among different alternatives during search is made by the estimate of how close a state is to a goal state. Hill-climbing is the best choice in calibration problem solving because we can easily provide an intuitive and efficient state evaluation function for estimating the distance from any state to a goal state. In fact, because of the complexity of the problem, we cannot afford a best-first search or any other systematic search algorithms. The distance estimation function is defined as follows:

$$d(s) = \sum_{i=0}^{m} d_i^s$$

where m is the number of the target parameters, and

$$d_i^s = \begin{cases} |T_i^s - D_i| / L_i & \text{if } |T_i^s - D_i| / L_i > 1 \\ 0 & \text{otherwise} \end{cases}$$

As can be seen from the above definition, the distance from a state toa goal state is the sum of distances of individual target parameters. This distance estimation is dimensionless and is measured as a multiple of individual tolerance interval lengths. The distance is larger than 1 if there exists at least one target parameter whose value does not fall into the tolerance interval associated with its target data value. The distance equals 0 otherwise. By applying *ADJUST* operator to a state, one can typically reduce the distance of at least one target parameter with respect to its target value to 0 by adjusting a control parameter within its valid range. With this estimation function, the hill-climbing strategy can be described as follows: At any state s, find i such that $d_i^s = mas_{k=1}^m\{d_k^s\}$. Then, select a control parameter C_j for the complete binding of operator $ADJUST(C_j,T_i,D_i,L_i)$. Some heuristic rules need to be applied to select the best control parameter to adjust. The selection heuristic in general can be different from model to model and will not be discussed in detail in this paper.

2. Incremental Solution Refinement using Dynamic Tolerance Window

Two problems emerged when we tried to apply the hill-climbing search control strategy alone. First, for a model like GESIM, it can be difficult for a target parameter to attain the corresponding target value if the other target parameters are too far away from their respective target values. Second, after a target parameter is perfectly matched with its target value, if another target parameter is far away from its target, then the subsequent step matching the second target parameter will very likely undo the matching of the first target parameter. This naturally led us to adopt an incremental problem solution refinement strategy as described in the following.

At the beginning of a calibration, relax the goal condition by enlarging the target parameter tolerance intervals. When the relaxed goal condition is satisfied, then tighten the goal condition to some degree and repeat the calibration process again until the very original goal condition is satisfied. This strategy is implemented by using a dynamic tolerance window associated with each target parameter. The window for target parameter T_i is defined as an interval $[D_i - L_i * \alpha, D_i + L_i * \alpha]$, where α is the knob used to control the window size. As can be seen, the value of α is the same for all the windows, whereas the actual window sizes can be all different due to the differences in the tolerance intervals. With the GESIM calibration, the initial value for α is set to 3, and is decreased by 1 each time the goal condition associated with the current window size is satisfied.

Initially, this dynamic tolerance window was adjusted in one direction only, namely, its size can only be reduced. It was found, however, when the calibration program reaches a "local maximum", increasing the tolerance window size can help to escape this trap. A state is classified as a local maximum if from that state, the *ADJUST* operator "favored" by the heuristic cannot successfully match a target parameter with its target value. Whenever such a "local maximum" is detected, the tolerance factor is increased until the tolerance window reaches the size such that any further increment makes all the goal conditions trivially satisfied. It is typical that, with this new window, a different target parameter, will be "favored" and selected.

Consequently, the calibration process is guided into a quite different direction. This strategy may still fail in some situations. A radical change of state is then necessary in order to escape the "local maximum". To achieve that, each control parameter is independently perturbed by a random factor, and then the search starts over again. The SET opertor is used to assign a new value to

each parameter. This combination of window change and random state perturbation have proved to be a very effective strategy in escaping from "local maxima" during a search.

3. Ordering Calibration Target Parameters

A serious problem affecting the efficiency of a calibration process is that the matching of a target parameter can often undo the matching of another parameter. This problem can cause the calibration process to "thrash", or go in cycles, repeatedly matching the same subset of target parameters.

The third search control strategy is to essentially identify the mutual interference among the target parameters and then use the information to override the target parameter selection based on the state distance estimation function. For example, if whenever the calibration program changes T_i, T_j gets affected by a significant amount, but not vice versa, then a heuristic rule is that T_i should be matched first. The reason is simply that if T_i is chosen first for the matching, then matching T_i at sometime later will very likely undo the matching of T_j; however matching T_j after T_i will less likely affect the value of T_i. To represent this kind of mutual interference in general, we define a relation, denoted by *ORDER*, over a set of target parameters such that *ORDER*(T_i, T_j) is true if and only if changing T_i will cause T_j to change a significant amount, while changing T_j may only affect T_i by a negligible amount.

To incorporate this ordering relation among target parameters into the procedure for selecting target parameters, the procedure is modified as follows: For a state s, first, a target parameter, say T_i, is selected as a reference based on the distance measure described above. Then, the ordering relation is checked to see whether or not there exists any other target parameter, say T_j, such that *ORDER*(T_j,T_i) is true, and $d_j' > 0$. If such a target parameter T_j is found, then select T_j to be the new reference parameter and repeat the step again. Otherwise, keep T_i as the reference parameter and continue the calibration process.

With the combined search heuristic for the calibration process, the efficiency of the calibration program is greatly improved. The remaining problem is how to acquire the knowledge about the mutual interference between the parameters and construct an *ORDER* relation based on the acquired information. This issue is addressed in the next section.

Ⅳ. Learning the Ordering Heuristics

Although the ordering heuristic is very important to the efficiency of the calibration process, it is very difficult to derive even for the domain experts. Even harder is the generation of heuristic rules that are dependent on state conditions. This naturally led us to look for a machine learning technique as a means of acquiring such heuristics. Based on the observation of model output after changing control parameters to various different states, a learning system is expected to derive general ordering rules. The rules should reveal a plausible ordering of parameter adjustment and the condition under which the ordering holds.

1. Appropriate Approach to Learning in Calibration Domain

There have been significant research results reported under the topic of deriving subgoal ordering rules for efficient search control. The reasoning approach by Cheng and Irani [1, 2] and the problem space compilation method by Etzioni [4] analyze the preconditions and postconditions of problem operators and derive a partial ordering on the set of subgoals to avoid goal interferences. Minton's EBL (Explanation-Based Learning) [7] and the goal stack analysis method by Ryu and Irani [10] learn goal ordering rules from analysis of search traces which led to either successes or

failures. Common to all these analytical approaches is the assumption that a clear model of the domain problem is available in the form of a well-defined set of problem operators. The effect of making a move in search space under a certain condition is clearly predictable and thus the overall system behavior can be analyzed.

Unfortunately, for the engine model calibration, our knowledge of underlying physics of the model is often far from sufficient to make possible accurate prediction of the model's behavior. In other words, we can hardly predict effects of matching a certain target parameter on the rest of the target parameters in various different states. Even after observing these effects, we cannot explain exactly how they happened. Therefore, none of the above mentioned analytical methods directly applies to our problem of deriving ordering rules for search control. We could only take a data-driven approach in which we rely on our collection of data obtained through many runs of the model to empirically derive or learn ordering rules. These rules are supposed to be useful for determining the order that minimizes mutual interference under any state condition.

Each data point resulting from running the model is described by the model input and output which are the state information and the effect on target parameters, respectively. A lot of such data serve as training examples from which an empirical learning algorithm captures underlying regularities in the form of a set of rules. The state information is represented by attribute value pairs where each attribute corresponds to a control parameter. The effect on the target parameters is interpreted in such a way that the ordering relation between each pair of parameters becomes evident. This interpretation is essentially an assignment of class membership to each of the data point. Overall, our learning problem can be characterized as an empirical and classification type of learning for which the ID3 algorithm [8] is appropriate. We picked up one of the extensions of the ID3, called GID3 * [5] for our rule learner. In the following, we give a step-by-step illustration of the whole learning procedure from the generation of the training data to the derivation of ordering rules for GESIM calibration.

2. Learning Procedure

First, a set of raw data is collected by running GESIM under different combinations of the four control parameters. We form a group consisting of five runs by first running one base case and then changing each control parameter by a certain amount as shown in Table 1.

Table 1. A group of five GESIM runs: The first row is theresult of the base run and the rest show the effect afterchanging each control parameters to the indicated values.

C_1	C_2	C_3	C_4	T_1	T_2	T_3	T_4
Bnum	*Celin*	*Hfact*	*Hfactb*	*Burn0-10*	*Ten-pmax*	*Airflow*	*Isfc*
2.0	2.2	0.7	1.0	19.74	17.66	24.59	230.82
4.0	2.2	0.7	1.0	21.21	17.58	24.61	230.49
2.0	2.5	0.7	1.0	19.53	17.18	24.47	231.19
2.0	2.2	1.1	1.0	21.43	18.40	22.78	232.77
2.0	2.2	0.7	1.4	19.82	17.65	24.33	241.44

To see the effect of changing each control parameter, percent changes of target parameter values are calculated for each run result with respect to the base run result. The base run result can then be removed from the data table but the state condition of the base run is remembered as the state condition of all the other runs within the group. Each row in the data table is indexed such that each index indicates the target parameter intended to be changed. For example, in Table 2, the row with index "2" shows the percent changes in the target parameters when the second one (*Ten-pmax*) is meant to be changed. More generally, the ith number in the row with index i indicates

the intended effect and the rest in the same row reflect the side effects.

Table 2. Percent changes of target parameter values after eachintended change.

State:
Bnum= 2.0, *Celin*= 2.2
Hfact= 0.7, *Hfactb*= 1.0

	T_1 *Burn0-10*	T_2 *Ten-pmax*	T_3 *Airflow*	T_4 *Isfc*
1	7.45	0.47	0.10	0.14
2	1.06	2.74	0.46	0.16
3	8.54	4.21	7.36	0.84
4	0.41	0.06	1.04	4.60

We can see from Table 2 that changing *Burn0-10* has almost no effect on the others (row 1), while changing *Airflow* has significant side effects both on *Burn0-10* and on *Ten-pmax* (row 3). To have this relative level of significance interpreted into a quantitative measure, we introduce a threshold denoted by θ. Let v_{ij} be the percent change of the value of T_j (jth target parameter) when T_i was intentionally changed. v_{ij} appears as the jth value in the ith row of GESIM run group. If $v_{ij}/v_{ii} < \theta$ for a small non-negative θ, v_{ij} is turned to zero. This allows us to ignore relatively small side effects. Table 3 shows the result of applying $\theta = 0.1$ to the values of Table 2.

Table 3. Result of applying $\theta = 0.1$ to the percent changes oftarget parameter values shown in Table 2.

State:
Bnum= 2.0, *Celin*= 2.2
Hfact= 0.7, *Hfactb*= 1.0

	T_1 *Burn0-10*	T_2 *Ten-pmax*	T_3 *Airflow*	T_4 *Isfc*
1	7.45	0.00	0.00	0.00
2	1.06	2.74	0.46	0.00
3	8.54	4.21	7.36	0.84
4	0.00	0.00	1.04	4.60

Table 4. Training data represented by attribute value pairs.

Data Category	State *Bnum*	*Celin*	*Hfact*	*Hfactb*	Class Assignment
P12	2.00	2.20	0.70	1.00	$ORDER(T_2, T_1)$
P13	2.00	2.20	0.70	1.00	$ORDER(T_3, T_1)$
P14	2.00	2.20	0.70	1.00	$NO\text{-}ORDER(T_1, T_4)$
P23	2.00	2.20	0.70	1.00	$ORDER(T_3, T_2)$
P24	2.00	2.20	0.70	1.00	$NO\text{-}ORDER(T_2, T_4)$
P34	2.00	2.20	0.70	1.00	$ORDER(T_4, T_3)$

Now we are ready to derive ordering relation between the parameters. To simplify the problem, we pay attention to a single pair of parameters at a time. For example, by comparing the second and the third rows of Table 3, we notice that changing *Ten-pmax* does not affect *Airflow* as much as changing *Airflow* does *Ten-pmax*. This observation leads us to the conclusion that *Airflow* should be changed before *Ten-pmax* in this state. We can make a similar comparison for every pair of parameters to assign ordering. Again, we need a quantitative measure of the relative significance of mutual interferences. Let v'_{ij} be the percent change of the value of T_j upon change of T_i after applying the threshold θ. If $v'_{ij}/v'_{ii} > \eta$ v'_{ji}/v'_{jj} for some $\eta \geq 1$, we say T_i interferes with T_j significantly more than T_j does with T_i, thus assigning the ordering $ORDER(T_i,T_j)$ to this parameter pair (note that $v'_{ii} = v_{ii}$). Similarly if $v'_{ji}/v'_{jj} > \eta$ v'_{ij}/v'_{ii} for some $\eta \geq 1$, we can say T_j interferes with T_i more than T_i does T_j, thus assigning $ORDER(T_j,T_i)$. If it is not any of the two cases, then we conclude $NO\text{-}ORDER(T_i,T_j)$ because the interference is either severe in both directions or not significant at all. Table 4 shows the result of assigning ordering relationship to every pair of target parameters with $\eta = 1.5$. Note that the state condition is attached to each pairwise comparison result. The pairwise comparison results serve as training data for the learning program GID3*. The data belonging to the same category are collected together and fed to GID3*, to derive a set of ordering rules for the given parameter pair. Figure 1 shows one such rule which recommends an ordering under the specified condition. A total of 105 rules were derived across all six rule sets with 480 data points provided per category. These rules can be used in the following way to minimize interferences between the target parameters. When given a state condition in terms of the values of *Bnum*, *Celin*, *Hfact*, and *Hfactb*, each rule set returns the ordering between a particular pair of target parameters. The pairwise ordering recommendations from all

the rule sets are collected together to derive a partial ordering of all the target parameters. This ordering relation is used for selecting the target parameter to be matched as explained in Section Ⅲ-3. The *ADJUST* operator is then applied by changing a control parameter's value iteratively until reaching a specified state. If the state reached is a goal state, calibration is completed. Otherwise, the rules are searched again in this new state and the same process repeats.

If (*Bnum*〈7) ∧ (*Celin* 2.65) ∧ (0.9≤*Hfact* 〈 1.3)
Then *ORDER*(*Airflow*, *Ten-pmax*)

Fig. 1. An ordering rule derived from the data category P23.

Ⅴ. Summary and Future Work

ICALIB has been used on a daily basis for engine model applications, and has reduced the time required for model calibrations from many hours to a few minutes on average. Because of the dramatic speed-up gained for the calibration process, we found several calibration cases finished successfully which were not possible before due to the computing resource limit.

Although the three heuristics that we use now work successfully, we have not sufficiently studied the individual contribution of each heuristic. More work is needed to identify the conditions under which each heuristic plays a major role, so that a more strategic use of them is possible. We also need more study of the sensitivity of learned result to the values of the two thresholds θ and η. They are used for judging the relative significance of side effects and are currently set to values subjectively determined.

In our current study, interference between parameters has simply been considered as harmful. As a matter of fact, there are cases where the side effect of a certain adjustment is helpful in a sense that the affected parameter's value is moved towards its target value. Therefore, we need to investigate the possibility of incorporating this specific information into the calibration program to improve its efficiency.

One final item to look into in the future is the problem of operator selection. For the GESIM model, we have the right control parameter, predetermined from the domain knowledge, to adjust to match each of the target parameters. In general, more than one control parameters might need to be adjusted simultaneously to match a single target parameter. Even worse, the set may have to vary from state to state for a calibration to finish successfully. Again, we believe an empirical machine learning approach will be appropriate.

References

[1] J. Cheng and K.B. Irani, "Ordering Problem Subgoals," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 931-936, Detroit, MI. 1989.

[2] J. Cheng, "Deriving Subgoal Ordering Constraints Prior to Problem Solving and Planning," *Ph.D dissertation in* EECS Dept. The University of Michigan, 1991.

[3] H.A. Cikanek, J. Cheng, C.E. Newman and G.C. Davis, *Automation and Analysis of GESIM Calibration*, Ford Research Technical Report No. SR-91-11.

[4] O. Etzioni, "STATIC: a Problem-Space Compiler for Prodigy," *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 533-540, Anaheim, CA. 1991.

[5] U.M. Fayyad, "On the Induction of Decision Trees for Multiple Concept Learning," *Ph.D dissertation* in EECS

Dept. The University of Michigan, 1991.

[6] J.R. Janzen, C.E. Newman and G.C. Davis, *General Engine Simulation Program GESIM)*, Ford Research Technical Report No. SR-88-08, February 1988.

[7] S. Minton, "Qualitative Results Concerning the Utility of Explanation-Based Learning," *Proceedings of the Seventh National Conference on Artificial Intelligence*, St. Paul, MN. 1988.

[8] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning 1*, pp. 81-106, 1986.

[9] E. Rich and K. Knight, *Artificial Intelligence*, 2nd edition, McGraw-Hill, 1991.

[10] K.R. Ryu and K.B. Irani, "Learning from Goal Interactions in Planning: Goal Stack Analysis and Generalization," *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 401-408, San Jose, CA. 1992.

著 者 紹 介

柳 光 烈(正會員)

1956年 10月 27日生. 1979年 서울대학교 전자공학과 학사. 1981年 서울대학교 전자공학과 석사. 1983年 3月 ~ 1984年 8月 충북대학교 공대 전자계산기공학과 전임강사. 1992年 미시간 대학교 컴퓨터공학 박사. 1992年 3月 ~ 1993年 2月 포드자동차사 과학연구소 연구원. 1993年 3月 ~ 현재 부산대학교 공대 컴퓨터공학과 전임강사. 주관심분야는 인공지능, 기계학습, 지식기반 시스템, 시스템 통합 등임.