

# DESIGN OF A PRODUCTION CONFLICT AND ERROR DETECTION MODEL WITH ACTIVE PROTOCOLS AND AGENTS

C.L. Yang, X. Chen, S.Y. Nof

PRISM Center

School of Industrial Engineering, Purdue University, 315 N. Grant Street, West Lafayette, IN, U.S.A.

## Abstract

A production conflict and error detection model has been developed to detect problems that occur in a collaborative environment. A case study was performed on three automobile company agencies (vehicle delivery center, shipping yard, and dealers) in a finished vehicle distribution/shipping environment. The vehicle shipping plan was investigated and possible errors and conflicts were identified. A proposed conceptual model, conflict/error detection model (CEDM) with active middleware, which is able to detect possible errors and conflicts and reduce the loss or damage in the car shipping plan, was developed. A computerized simulator, The Management Interactive Case Study Simulator (MICSS), was adopted to simulate and detect errors and conflicts in three agencies, which are represented by three MICSS views, production, purchasing, and marketing, respectively. Experiment results show that the proposed CEDM can detect all errors and conflicts as simulated by parameters in MICSS.

## Keywords:

Production conflict and error, Active middleware, Agent, Detection model, Detection protocol, Collaborative e-Work

## 1 INTRODUCTION

### 1.1 Finished vehicle shipping plan

For a modern automobile company, various kinds of car models and millions of cars are assembled in different locations. After car assembly is completed, shipping those finished cars to customers or dealers is an important planning issue. This planning is part of collaborative e-Work [8]. All personnel, schedules of shipping transportation, and resources such as truck and train, and even third-party shipping company have to be planned and coordinated well to fulfill the shipping requirements. In this complicated collaborative environment, however, conflicts and errors (CE) are unavoidable because of human mistakes, planning conflicts, and limited resources. Since detecting CE is an initial step for resolving them, a sophisticated detection model is needed. In order to detect all possible conflict and error problems, an agent-based detection model is proposed to perform a detection process in a shipping network.

A case study of finished vehicle shipping/delivery plan is investigated in this research. Three major agencies: shipping yard, vehicle delivery center (VDC), and dealers, coordinate with each other to deliver finished vehicles to customers. Figure 1 illustrates this finished vehicle shipping network.

#### Shipping yard

The shipping yard is a temporary storage close to assembly factory. Once car assembly operations are completed, the finished vehicles will be deposited at the shipping yard temporarily and wait for delivery to VDC. The finished vehicles will be delivered to VDC by truck and rail. In the shipping yard, moving vehicles to pre-assigned parking lot and delivering them to VDC according to the shipping plan/schedule are major tasks.

#### Vehicle delivery center (VDC)

VDC is a contracted partner that is in charge of delivering finished vehicles to different car dealers according to orders from dealers. VDC plays a "relay" role in vehicle delivery and manages the delivery plan. VDC also

maintains the vehicle trucks to ship finished vehicles to different dealers at different locations.

#### Car dealers (assumptions)

Car dealers can be grouped to several demand areas. In this study it is assumed that in each demand area, 10 to 25 dealers might be distributed. Each VDC will serve 10 to 50 demand areas on average. Dealers will send orders to VDC weekly based on their customers' orders. On average, each order may contain 10 to 30 vehicles.

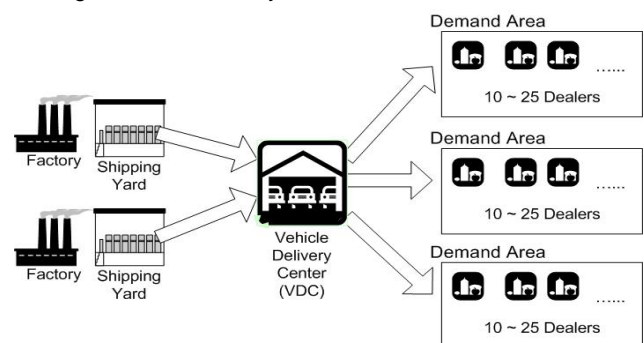


Figure 1: One VDC with two "parallel" shipping yards.

### 1.2 Possible errors and conflicts in a finished vehicle shipping plan

Several kinds of errors and conflicts may occur in this collaborative delivery operation. For instance, if workers make a mistake in placing the wrong car at the shipping yard, or have a typo in paperwork, a wrong vehicle might be delivered to the wrong VDC. In this situation, re-transporting the vehicle to the correct destination is costly. This kind of error should be detected as soon as possible and be prevented in advance.

In addition, conflict problems may also occur in dispatching plans. For example, a schedule conflict might happen when ordered vehicles are out of stock in VDC or there is a delivery delay from a shipping yard to VDC. Further reschedule or modification of the delivery plan might also lead to other conflict situations, such as labor dispatching

conflicts. This kind of conflict situation is common since the prediction of orders is usually not precise. In order to increase service level, such as reducing customers' waiting time, conflict and error detection on the finished vehicle delivery plans is crucial for elimination or minimization of such damages. Typical possible conflict and error problems are listed below:

Table 1: Typical possible conflicts and errors.

Error	Conflict
<ul style="list-style-type: none"> <li>• Job missing</li> <li>• Task misunderstanding</li> <li>• Facility crash</li> <li>• Machine fatigue</li> <li>• Human mistake</li> <li>• Unaccepted quality</li> <li>• Abnormal event</li> <li>• Exceed resource capability</li> <li>• Insufficient ability/capacity</li> <li>• Error in specification</li> <li>• Information network break down</li> </ul>	<ul style="list-style-type: none"> <li>• Time (schedule) mismatch</li> <li>• Unexpected cost</li> <li>• Unsatisfied profit</li> <li>• Different processes or operations</li> <li>• Various task specifications</li> <li>• Resource overuse</li> <li>• Violation of common goal</li> <li>• Collision (path conflict)</li> <li>• Layout overlap</li> <li>• Different data format</li> <li>• Different units of measures</li> </ul>

In Section 2, the proposed conflict and error detection model is addressed. Section 3 describes the experiment design that is supported by MICSS, and Section 4 shows the results. Section 5 provides a short discussion and concludes the article.

## 2 CONFLICT AND ERROR DETECTION MODEL

### 2.1 Active middleware and detection model

Middleware is a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems [4] [6]. It is an enabling layer of software that resides between the business application and the networked layer of heterogeneous platforms and protocols [1]. According to Anussornnitisarn and Nof [2] [3], the major components of active middleware are: Multi-Agent-based Systems (MAS), Workflow Management Systems (WFMS), coordination protocols, Decision Support Systems (DSS), modeling language/tools, task/activity databases. Their conceptual model is shown in Figure 2.

In a distributed, collaborative environment, each participant has its own goals, tasks, and resources. The management system of each participant, e.g., information system, project planning system, database system, Enterprise Resource Planning (ERP) system, or manufacturing execution system plays an important role in managing the project, manipulating the operation of tasks, and monitoring the resources. Because of the heterogeneity of those distributed participants' systems, smooth collaboration between these systems is difficult to achieve. Active middleware can serve as a bridge between distributed systems and provides an integrated platform to communicate and cooperate. The term "Active" implies that in addition to normal middleware function, there are automated, e-Work supported decisions made for more effective integration.

Agent technology plays an important role in detecting CE problems [8]. In our proposed detection model, conflict and error detection model (CEDM) [9][10], each operator and participant should deploy a conflict and error detection agent (CEDA) to collect the needed information such as current shipping quantity, the updated shipping plan, or current status regarding shipping operations. After gathering relevant information, each agent will compare it

with the existing shipping guidelines and regulations to detect any error or conflict situation. Besides, the agent also transmits the obtained information and evaluation results to collaborating parties by applying a designed protocol, conflict and error detection protocol (CEDP). The protocol regulates and rationalizes agents in communicating with each other. Then, the on-line detection process is executed iteratively and continually among the collaborative networked parties. All activities performed by CEDA are supported by active middleware in CEDM model. More details about CEDA and CEDP are addressed in following sections. Figure 3 shows the basic architecture of CEDM with active middleware, based on the production shipping environment.

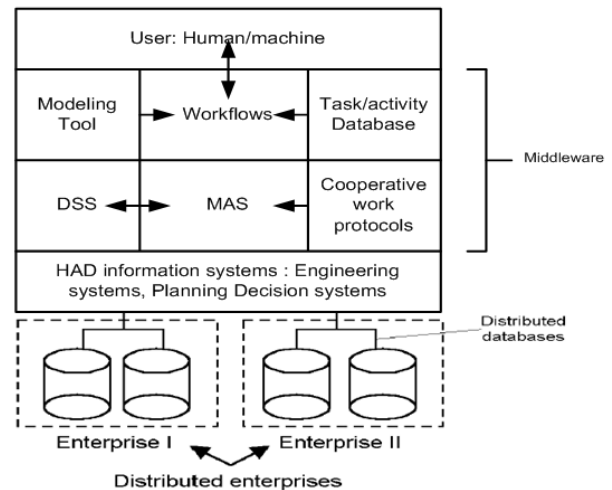


Figure 2: Active middleware architecture [2].

### 2.2 Conflict/error detection agent

CEDA is a software agent that is used to perform the CE detection process in a Collaborative Unit (Co-U), such as shipping yards, VDCs, and dealers of a shipping network. There are three components in the CEDA: 1. Detection Policy Evaluation Mechanism (DPEM), 2. Error Monitoring Mechanism (EMM), and 3. Conflict Evaluation Mechanism (CEM). These three components coordinate together to detect possible errors and conflicts and notify the correlated (potentially infected) Co-U's by CEDP.

#### Detection Policy Evaluation Mechanism (DPEM)

The Detection Policy Evaluation Mechanism (DPEM) is responsible for evaluating each detection method ( $dM$ ) regarding each possible CE problem and generating the detection policy ( $dP$ )<sub>T</sub> for detecting it. The detection policy is a guideline of how to detect a particular CE problem. Based on the information stored in the CE knowledge base, DPEM can evaluate all applicable  $dM$ s for a specific CE and select the best one. Then, DPEM will evaluate the cost-effectiveness of the selected  $dM$  and decide whether it should be applied, based on cost aspect. After evaluation, DPEM implements the effective detection policy ( $dP$ )<sub>T</sub> in executing the CE detection when the Co-U intends to perform particular task  $T$ .

#### Error Monitoring Mechanism (EMM)

The error monitoring mechanism (EMM) of the CEDA is in charge of continually monitoring the Co-U's activities. CEDA obtains the run-time current state ( $\Theta$ ) from the input of different monitoring devices. Based on the Co-U's plan of resource policy, shipping task specification, and current state, an error can be detected, recognized and managed. Until the error situation is recovered, EMM will continue to track this error.

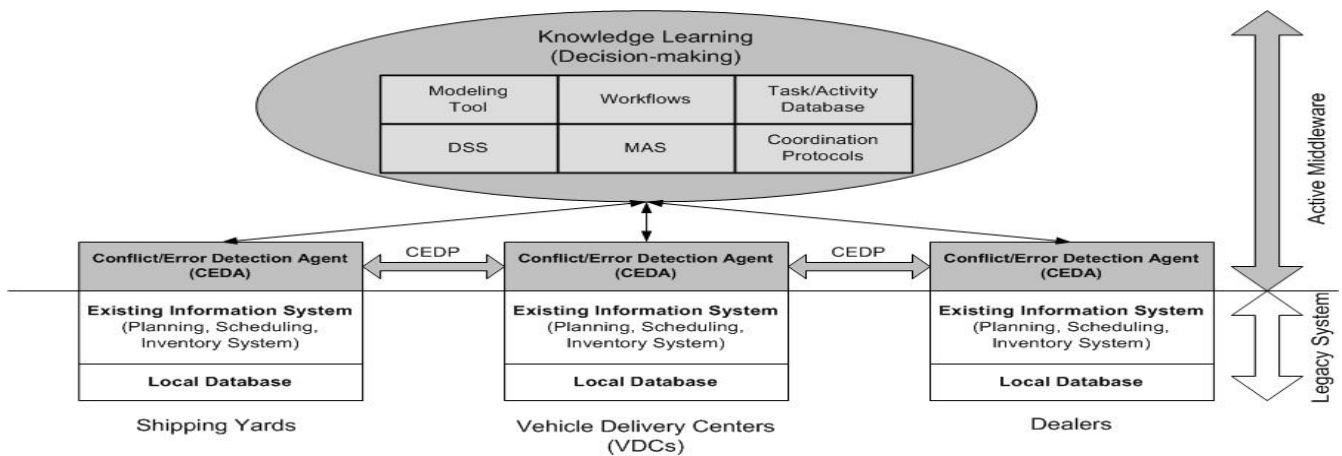


Figure 3: CEDM with active middleware

### Conflict Evaluation Mechanism (CEM)

The conflict evaluation mechanism (CEM) is the kernel of the CEDA with two functions, calculation and comparison, to detect possible conflicts when a Co-U performs a shipping task. Through these two functions, CEDA can standardize all current states ( $\Theta$ ), tasks constraints ( $\Omega$ ), and compare them to find a conflict when unsatisfied task constraints ( $\Omega$ ) exist. Figure 4 shows the components of CEDA with their input and output.

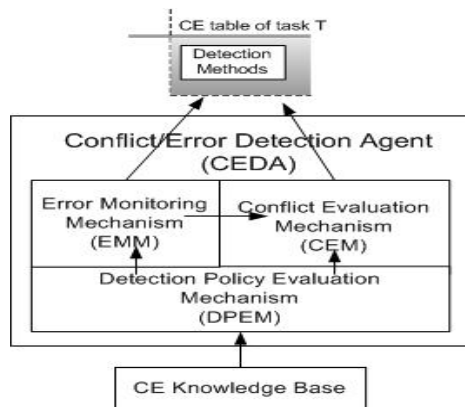


Figure 4: Inputs, outputs, and components of CEDA.

### 2.3 Conflict/error detection protocol

CEDP is an agent-based protocol that facilitates the exchange of detection information between organizations participating in the shipping network. CEDP enables CEDA to send or receive CE announcements, CE evaluation requests, and CE evaluation results. Through this protocol, not only CE events can be transmitted within a shipping network, but also the CE evaluation information can be shared when it is needed for the detection process.

#### Message Definition

CEDP handles the detection information exchange process among Co-Us, such as shipping yard, VDC, and dealers. Three kinds of messages are transmitted by this protocol: a CE announcement, a CE evaluation request, and a CE evaluation result.

1. CE announcement. Once a Co-U detects a CE inside its system, CEDA will broadcast the CE announcement to other correlated Co-Us through CEDP. When a CE announcement is received, the CEDA of a Co-U will evaluate this CE event.
2. CE evaluation request. A CE evaluation request is a message to ask a Co-U to evaluate its activity and return its evaluation result. The request receiver must examine its task processes and evaluate their potential influence.

3. CE evaluation result. After a CEDA receives a CE announcement or CE evaluation request, the CEM of the CEDA begins to evaluate its task activities and return evaluation results to the requesters.

#### CEDP Operation

Two kinds of CEDP operations are executed between Co-Us.

1. A CE is detected. Once a Co-U detects a CE problem that occurs within its boundary, this Co-U is responsible to inform other correlated Co-Us. Every Co-U that receives a CE announcement evaluates the potential influence of this CE and sends back the evaluation result. Then, any Co-U that detects a CE can also estimate the potential influence on other Co-Us.
2. A task specification is changed. If a Co-U wants to change the task specification, this change might affect other partners and cause conflicts. This change initiator should inform other cooperative participants. After evaluating the task activities regarding the new task specification, all participants will send back their evaluation result.

In the implementation perspective, CEDA is a software agent that can build upon existing shipping information system or managerial process. CEDA should access the needed data that are stored in the existing system to perform the CE detection. The detection logics and experiences can be accumulated and provided by CE Knowledge Base that is also part of the active middleware layer. In addition, by applying CEDP, each CEDA is able to transmit the detection information to CEDAs that represent other collaborative participants. Once a CE problem is detected by one CEDA, the corresponding CEDAs will continue to analyze and detect the possible CE problems that might propagate to other participants, until all detection processes are completed.

## 3 EXPERIMENTAL DESIGN

### 3.1 MICSS as a simulation tool

The Management Interactive Case Study Simulator (MICSS) by MBE-Simulations Ltd [7] is a computerized case study that simulates the realities of a manufacturing company. The objective of MICSS is to simulate complex systems, the rules that govern those systems, and the techniques needed to control the performance of such systems.

The simulated company is driven by four functions: Marketing, Production, Purchasing and Finance. Each function provides its own "view," including information, managerial actions and policies. To obtain better results (profit, sales, etc.), it is necessary to synchronize and

coordinate the decisions and actions of all four views. This synchronization is called "the global view of the system." Attaching a global view is part of the challenge of MICSS. Any errors or conflicts in the company will affect the results in a complicated, often counter-intuitive way. The simulation can be run for any time period less than or equal to one year.

#### Marketing View

The marketing view displays the products the simulated company sells. MICSS recognizes two types of markets: the customer market and the contracted market. The customer market consists of many casual customers who place orders for a small quantity of a particular product. They pay the list price and are supposed to get their order at the specified quoted lead time (QLT). The contracted market is based on contracts with large clients.

#### Production View

The Production View displays the work centers and their occupants on the production floor. The production floor is driven by work orders (WOs). There can be no production unless a WO is issued. Every WO specifies a product and the number of units to be produced. The raw materials are released from the stockroom by the WO. Every work center has a list of work orders to be fulfilled. The WOs are created automatically by the embedded information system that is part of MICSS. The parameters of the planning algorithm that creates the WOs are set by the users.

#### Purchasing View

The purchasing view deals with providing the necessary "materials" (vehicles) to production. In our experiments, vehicles are purchased from suppliers. Available suppliers are listed in the "Actions" menu. Users can also change the default supplier. Purchasing is usually done automatically according to the rules users set. The basic rule is the order-maximum level. When the stock at hand plus the open orders from the vendor is less than the order-level, an automatic order is issued to the default supplier for the quantity to replenish up to the maximum level. The alternative is to purchase according to the MRP (ERP) algorithm. In the parallel operation, vehicles are purchased from two suppliers. In the nonparallel operation, vehicles are purchased from one supplier.

#### Finance View

The finance view does not have any "Actions" or "Policies" entries. Its purpose is to provide financial information. The finance main screen displays the current profit and loss statement. This statement is updated every month, from the start of the year until the end of the last month.

### 3.2 Detect errors and conflicts with MICSS

MICSS is sufficient to simulate the errors and conflicts in a company [5]:

- Inputs (may include errors) to the system are determined by users (employees of the company).
- Different inputs produce same or different outputs (profits, sales, etc.)
- Different combinations of inputs (may include conflicts) have different outputs (results and impacts on the company).

In MICSS there are 20~40 parameters that are used as inputs to the system, depending on the scenarios used. Some inputs yield relatively better outputs (higher profits and sales, short production time) while some inputs yield worse outputs (losses, bankruptcy). However, it is recognized that better outputs (higher profits and sales) do not always indicate better performance of a company; they may be only short-termed. Therefore, a combination of

best inputs (baseline policy) can be determined by the company and errors and conflicts corresponding to various inputs are defined by comparing the outputs *with* errors or conflicts to the outputs of the baseline policy. The detection of errors and conflicts in MICSS can then be simulated by examining the outputs of the company with the help of CEDA and CEDP. The outputs to be examined include not only long term or final outputs, such as profits and sales (the identification of them may be meaningless because catastrophic results already happen), but also short term (midway) or sensitive outputs, such as customer order, resource idle time, sales, etc.

The procedure of detecting errors and conflicts in MICSS is shown in Figure 5. The outputs of the baseline and outputs with errors and conflicts, as well as their corresponding inputs, are learned in advance and stored in the knowledge base. The outputs associated with any new inputs are compared to the outputs of the baseline stored in the knowledge base. If there are errors or conflicts, further comparison is conducted to find out the type of error or conflict. There could be several possible errors/conflicts of which the knowledge is not stored in the knowledge base. In that case, new knowledge is learned with the help of the user.

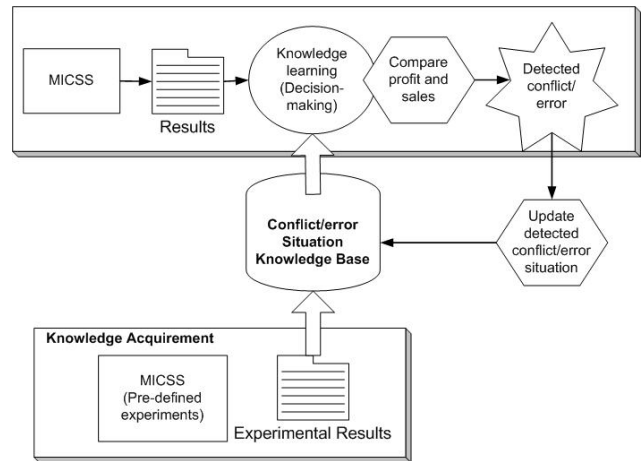


Figure 5: Conflict/error detection with MICSS evaluation.

The experiments are conducted in order to study the application of CEDM for a specific complex enterprise with parallel workflow, and compare the differences between the parallel and nonparallel workflows. The basic concept of the experiment design is to change the values of four parameters in MICSS to simulate errors and conflicts. Single input change is viewed as errors, and more than one input variations are treated as conflicts. Outputs (profits and sales) of the company are recorded and compared with the outputs of the baseline policy predefined by the company. The difference between outputs, the economic impact in this designated experiment design, serves as the indicator of the possible impacts of uncontrolled errors and conflicts.

The experimental design is described as follows:

- Four parameters including Price and QLT in the marketing view, and Max. Level and Order Level in the purchasing view are chosen to represent simulated errors and conflicts.
- The baseline policy (Tables 2 and 3), which is the best ideal set of inputs is run to obtain the best performance outputs [5]. The baseline policy is chosen to represent the "ideal" situation of shipping operation. By running MICSS with the baseline policy, ideally, it is possible to calculate the best long-term profits and sales (not necessarily the highest, which would be typically short-term oriented).

- c. Two suppliers (shipping yards) are selected. Two “raw materials”, representing two vehicle models, are supplied by one supplier, and another vehicle model is supplied by the second supplier. These two suppliers have different prices (shipping cost, storage cost, etc.) and QLT (delivery time) for each type of vehicle model.
- d. To represent different levels of errors and conflicts, adjust the value of four parameters to half or double, producing a set of 80 different experiments (C indicates the calculation of combination):  $C^4_1 * 2^1 + C^4_2 * 2^2 + C^4_3 * 2^3 + C^4_4 * 2^4 = 80$  experiments.
- e. Record the monthly profits and sales for each experiment.

Table 2: Baseline policy of marketing

Para- meters	Marketing (vehicle models)					
	A1	A2	B1	B2	C1	C2
QLT (Day)	16	16	18	18	16	16
Price (\$ X100)	200	226	150	146	210	240

Table 3: Baseline policy of purchasing

Parameters	Purchasing (vehicle models)		
	W1	W2	W3
Max. Level (car)	5500	4500	4000
Order Level (car)	3500	3000	2500

Three vehicle models W1, W2, and W3 are further categorized into six models A1, A2, B1, B2, C1, and C2 for marketing purpose.

#### 4 EXPERIMENT RESULTS AND ANALYSIS

Based on the design of experiment, two outcomes of MICSS experiments can be obtained: profit and sales, for each experiment with a particular parameter combination. The results show the variation of profit and sales during one simulated year and differences between parallel and nonparallel workflows.

##### 4.1 Analysis of experiment results

The profits at the end of the year are compared between parameter modification scenarios and the baseline policy. The results clearly show that the profit decreases if one or more parameters have been changed, representing undetected errors and conflicts. Furthermore, the profit decreases relatively more in the two-supplier scenario than in the one-supplier scenario, compared to the baseline policy.

CEDM can be applied by detecting certain evaluation metric change in the system. In terms of the profit, the detection algorithm can be defined as:

$$\frac{|\alpha_c - \alpha_B|}{\alpha_B} \times 100 \geq \gamma_P \quad (1)$$

$\alpha_c$  is the current profit,  $\alpha_B$  is the baseline profit and  $\gamma_P$  is the organization's predefined tolerance for variation; or the benchmark from the knowledge base.

In the parallel environment (two vehicle suppliers), the evaluation metric could have more dramatic change than in the non-parallel (single supplier) environment. This indicates the possibility of detecting CE problems promptly at a lower cost if appropriate CEDM model is established and monitored. On the other hand, the experiments results show that if CE problems have not been detected promptly,

the potential losses could be more severe in the parallel case.

In a practical CEDM model, CE problems should be detected at many different time points with different combinations of evaluation, metrics. In the experiments conducted, profit and sales are considered as metrics. Only the ends of the year profits are shown in Tables 4 and 5. The  $\gamma_P$  in Equation (1) can be set to 0.537%, which is the smallest profit loss in both single and parallel scenarios. However, profits might slightly decrease even when there is no error or conflict, because of various uncertainties: markets' fluctuation, competitors' products, etc. It is not practical to determine a value for  $\gamma_P$  for all different scenarios. Instead,  $\gamma_P$  can be set to 4.102% (the second smallest profit loss), which yields a more robust measure to detect errors and conflicts. Nevertheless, other metrics are needed to detect CE problems together with the profit.

Table 4: Maximum, minimum, and average profit loss with four kinds of conflicts/errors (Single)

	One-para.	Two-para.	Three-para.	Four-para.
# of experiments	8	24	32	16
# of bankruptcies	4	18	28	16
Max. profit loss (%)	212.786%	876.455%	891.298%	Bankrupted
Min. profit loss (%)	4.102%	14.515%	247.189%	Bankrupted
Avg. profit loss (%)	81.262%	295.406%	571.165%	Bankrupted

Note: The profit loss can exceed 100% while comparing to baseline.

Table 5: Maximum, minimum, and average profit loss with four kinds of conflicts/errors (Parallel)

	One-para.	Two-para.	Three-para.	Four-para.
# of experiments	8	24	32	16
# of bankruptcies	4	18	28	16
Max. profit loss (%)	388.148%	1451.11%	1472.71%	Bankrupted
Min. profit loss (%)	19.179%	0.537%	407.026%	Bankrupted
Avg. profit loss (%)	147.181%	493.135%	945.213%	Bankrupted

Note: The profit loss can exceed 100% while comparing to baseline.

It is observed that sales decrease in most scenarios when CE problems occur. The comparison between sales in the parallel and nonparallel environments does not provide meaningful conclusion. Hence, the evaluation metrics need to be carefully chosen when applying CEDM. However, the sales give a good example of detecting CE problems at different time points. The data collected at the end of January do not provide any insight of possible CE problems, while the data from February to December can be used to detect if CE problems occur. A simplified CEDM algorithm for both parallel and nonparallel environments can be written as:

$$\frac{|\beta_a - \beta_B|}{\beta_B} \times 100 \leq \gamma_S \quad (2)$$

$\beta_a$  is the magnitude of current sales,  $\beta_B$  is the baseline sales and  $\gamma_S$  is the organization's predefined tolerance magnitude, for variation or the benchmark from the knowledge base.

The data collected show that  $\gamma_S = -5.16\%$  (the smallest sales drop, from February to December) can be used to

detect errors and conflicts. The effectiveness of this algorithm, or the Type I error (CE problems exist but not detected) can be calculated:  $4 / 80 = 5\%$  (sales increased in four experiments). In other words, there is a 5% probability that an existing CE problem is missed if a random time point is picked to detect errors and conflicts. On the other hand, data collected are the average sales from one kind of parameter change in one month, instead of the actual sales, which also affects the effectiveness of the algorithm using sales as the evaluation metric. Therefore, a combined profit-sales metric is proposed next.

## 4.2 Combined evaluation metric

As explained above, profit as the detection metric is not robust. Meanwhile, using sales leads to a relatively larger Type I error. Furthermore, the detection of CE problems separately using profits and sales may give inconsistent or even contradictory results. Thus, there is a need to find the relationship among metrics and combine them in the algorithm. A suggested algorithm using profits and sales is as follows:

$$\lambda = \frac{\alpha^2}{\beta}, \lambda > \gamma_{PS} \quad (3)$$

$\lambda$  is the profit-sales ratio,  $\alpha$  is the profit,  $\beta$  is the sales and  $\gamma_{PS}$  is the organization's predefined tolerance to variation, or the benchmark from the knowledge base.

The square of the profit ( $\alpha^2$ ) eliminates negative values of profits. Equation (3) does not seem to be 'reasonable' as fewer sales but more profits are actually preferred, which indeed indicate there are errors or conflicts. Equation (3) defines a range within which the system has no errors or conflicts. Because profits are more sensitive to the CE problems, the square of the profit assigns it more weight and the algorithm is relatively less sensitive. Figure 6 illustrates results of the impact of CE on  $\lambda$ . The operation with CE problems has much bigger  $\lambda$  than that of the operation defined by the baseline (without CE problems).

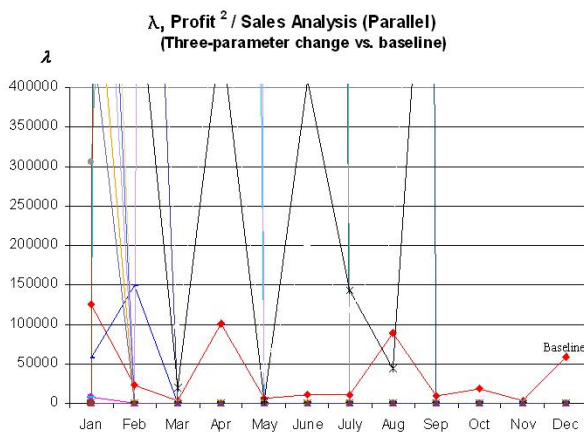


Figure 6: Impact of CE on  $\lambda$

## 5 CONCLUSIONS

The proposed conflict/error detection model is developed by the fundamental concept – "CE detection can be collaboratively performed by distributed agents." The ability to communicate with each other among participants of CE detection is important. Through designed CEDA interaction and CEDP channel, each of the participants not only connects with other participants, but also shares the detection information and support of active middleware components. This ability also extends the detection ability

from local point of view to global (entire shipping network) consideration.

The experimental analysis shows that the impact of errors and conflicts on the delivery operation is significantly larger when more CE problems occur concurrently (average profit loss is from 81% to 571% in the single shipping yard scenario and from 147% to 945% if CE is not detected in the parallel environment). The damage in the parallel environment is relatively higher than the damage in the non-parallel. Also, sales fluctuate dramatically when there are CE problems and in most cases sales decrease (-79% to 24% in the single shipping yard scenario and from -80% to 19% in the parallel environment, compared to the baseline policy). By and large, damages on both profits and sales are higher if CE problems are not detected earlier in time. The impact of CE problems is larger in the parallel environment.

The next step of this research will focus on developing the knowledge base of conflict/error detection and detection tools such as decision trees and neural networks.

## 6 ACKNOWLEDGMENTS

This research has been developed with the PRISM Center support, and support from GMR project on "Design of Active Middleware for Error and Conflict Detection (2004)".

## 7 REFERENCES

- [1] Anussornnitisarn, P., 2003, Design of Active Middleware Protocols for Coordination of Distributed Resources, Dissertation for Ph.D., Purdue University, West Lafayette, IN, U.S.A.
- [2] Anussornnitisarn, P., Nof, S. Y., 2001, The Design of Active Middleware for e-Work Interactions, Research Memorandum No. 2001-10, School of Industrial Engineering, Purdue University, West Lafayette, IN, U.S.A.
- [3] Anussornnitisarn, P., Nof, S. Y., 2003, e-Work: The Challenge of the Next Generation ERP Systems, *Production Planning and Control*, v 14, n 8, 753-765.
- [4] Bakken, D. E., 2003, Middleware, Chapter in *Encyclopedia of Distributed Computing*, Urban J. and Dasgupta P. (editors), Kluwer Academic Publishers.
- [5] Bellocci, T., Lehto, M. R., Nof, S. Y., 2003, Assuring Information Quality in Industrial Enterprises: Experiments in an ERP Environment, *Proceedings of the 10th International Conference on Human-Computer Interaction*, June 22-26, Crete, Greece, 654-658.
- [6] Bishop, T. A., Karne, R. K., 2003, A Survey of Middleware, *18th International Conference on Computers and Their Applications*, March 26-28, Honolulu, Hawaii, U.S.A.
- [7] MICSS (Management Interactive Case Study Simulator), October 2004, <http://www.mbe-simulations.com>.
- [8] Nof, S. Y., 2003, Design of Effective e-Work: Review of Models, Tools, and Emerging Challenges, *Production Planning and Control*, v 14, n 8, 681-704.
- [9] Yang, C. L., 2004, Conflict and Error Detection Protocol with Active Middleware, M.S.I.E. Thesis, Purdue University, West Lafayette, IN, U.S.A.
- [10] Yang, C. L., Nof, S. Y., 2005, Design of a task planning conflict and error detection system with active protocols and agents, *submitted to IJPR*.