PROGNOSTICS AND DIAGNOSTICS OF CONFLICTS AND ERRORS OVER e-WORK NETWORKS

X.W. Chen, S.Y. Nof
School of Industrial Engineering, Purdue University, 315 N Grant Street, West Lafayette, IN, USA

Abstract
This research studies two related functions over e-Work networks, prognostics with respect to conflict and error (CE) prediction, and detection with respect to CE diagnostics. Traditional prognostics and diagnostics approaches face two challenges: (1) How to model a system for effective conflict and error prediction and detection (CEPD); (2) Centralized CEPD algorithms are difficult to develop and implement for large systems. An agent-based approach is developed to model systems and corresponding decentralized CEPD logic is proposed for prognostics and diagnostics. A case study with four e-Work networks is applied to illustrate the proposed methodology.

Keywords:
Conflict detection, e-Work, error diagnostics, prognostics.

1 INTRODUCTION

Conflict and error prognostics and diagnostics are critical in distributed and decentralized operations where e-Business and e-Commerce activities require collaboration among multiple participants among whom conflicts and errors (CEs) are unavoidable. To predict, track, detect, trace, diagnose, and resolve CEs are fundamental challenges in collaboration. Various models, methodologies, algorithms, protocols, and tools have been developed to manage CEs.

In previous research, a *conflict* was defined as an inconsistency between cooperative units' (Co-Us) goals, dependent tasks, associated activities, or plans of sharing resources [1]. An *error* was defined as any input, output, or intermediate result that has occurred in a system and does not meet system specification, expectation, or comparison objective [2]. CE detection is extensively studied because of its unique importance to collaboration. If CEs in a network are not detected, (1) CEs cannot be diagnosed and resolved; (2) more errors and/or conflicts may occur due to propagation; (3) eventually collaboration among units in the network is damaged.

In this research, an agent-based approach is proposed to formally model a system for CE prognostics and diagnostics. A decentralized conflict and error prediction and detection (CEPD) logic is proposed and illustrated through a case study with four e-Work networks. The result shows the proposed CEPD logic performs better for networks with parallel activities. Future research will continue developing the CEPD logic for networks with different sizes (numbers of units) and topologies (collaboration methods). It is expected that (1) the agent-based modeling approach is more efficient than object-orientated approaches; and (2) the decentralized CEPD logic overcomes difficulties facing centralized algorithms.

2 RESEARCH BACKGROUND AND DEFINITION

2.1 Background

The prevalent method to detect conflicts was using layered constraints [3] [4] [5] [6]. Each unit must satisfy a set of predefined constraints which can be categorized into different layers, i.e., goal layer, plan layer, belief layer, task layer, and sub-task layer. Any violation of constraints that involves two or more units in a system is a conflict. Conflict detection was mostly studied in collaborative design [7] [8] [9]. A general conflict prediction and detection method was not developed yet.

Process monitoring, or fault detection and diagnostics in industrial systems, has become a new sub-discipline within the broad subject of control and signal processing [10]. Three approaches to manage faults for process monitoring and methods used in each approach are illustrated in Figure 1. The analytical approach generates features using detailed mathematical models. Faults can be detected and diagnosed by comparing the observed features with the features associated with normal operating conditions directly or after some transformation [11]. The date-driven approach applies statistical tools on large amount of data. The knowledge-based approach uses qualitative models to detect and analyze faults. It is especially suited for systems in which detailed mathematical models are not available.

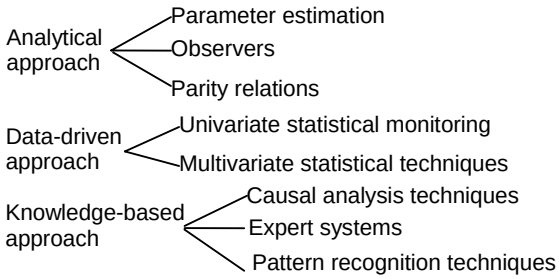


Figure 1: Fault management for process monitoring

Weaknesses of the above three approaches exist in terms of their ability to detect CEs: (1) All three approaches do not differentiate conflicts from errors. Better detection methods may be developed by recognizing differences between conflicts and errors; (2) They require domain-specific methods to be developed to detect CEs. Not only one of three approaches but also a method in the selected approach must be determined for a system, depending on the size, complexity, and modeling of the system. To find an appropriate method can be difficult and time-consuming; (3) All three approaches do not detect or diagnose typical CEs in industrial operations, e.g., scheduling conflicts. Moreover, little research so far has addressed the issue of error prediction.

Significant research also has been conducted on fault detection and diagnostics algorithms [12]. The major

weakness of those algorithms is the centralized approach in which algorithms and needed system information are stored at and controlled by a centralized location. This has been proven to be difficult and inefficient.

2.2 Prognostics and diagnostics definitions

To develop the CEPD logic, five basic concepts are first defined to model an e-Work network:

Definition 1: Co-U

A cooperative unit, Co-U, is an autonomous working unit in a network that executes tasks to achieve its local goals, and collaborate with other Co-Us to accomplish the common goals of a set of Co-Us in the network.

$$u(i,t) = \{\pi(i,t), \vartheta(i,t)\} \quad (1)$$

$u(i,t)$ is Co-U i in the network at time t. i is the index of Co-Us and is a nonnegative integer. The value of i is unique for each Co-U.

$\pi(i,t)$ is a set of constraint(s) in the system at time t that needs to be satisfied by Co-U i without collaborating with other Co-Us. Let $con(r,t)$ denotes constraint r in the network at time t. r is a nonnegative integer and the index of constraints. $\pi(i,t)$ is a set of constraints ($con(r_1,t), con(r_2,t), \ldots, con(r_m,t)$) that needs to be satisfied by $u(i,t)$ without collaboration.

$\vartheta(i,t)$ is Co-U i state at time t that describes what has occurred with Co-U i by time t. It includes necessary and sufficient information to determine whether or not constraints in $\pi(i,t)$ are satisfied. Two Co-Us $u(i_1,t)$ and $u(i_2,t)$ ($i_1 \neq i_2$) may have the same set of constraints ($\pi(i_1,t) = \pi(i_2,t)$) and same current state ($\vartheta(i_1,t) = \vartheta(i_2,t)$).

Definition 2: Co-net

A Co-net is a coordination network that enables cooperation, collaboration, and coordination among a group of Co-Us in a network. A Co-net r is defined as:

$$n(r,t) = \{\Omega(r,t), con(r,t), \theta(r,t)\} \quad (2)$$

$n(r,t)$ is Co-net r in a network at time t, r is the index of Co-nets and r is a nonnegative integer.

$\Omega(r,t)$ is a set of Co-Us in $n(r,t)$ that must collaborate to satisfy the constraint $con(r,t)$. Let $N(\Omega(r,t))$ denote the number of Co-Us in Co-net r at time t. $N(\Omega(r,t))$ is a positive integer and $N(\Omega(r,t)) \geq 2$.

$con(r,t)$ is constraint r that needs to be satisfied by $n(r,t)$ at time t.

$\theta(r,t)$ is Co-net r state at time t that describes what has occurred with Co-net r by time t. $\theta(r,t)$ includes necessary and sufficient information to determine whether or not $con(r,t)$ is satisfied.

Constraints and Co-nets have one-to-one relationship. Each constraint has one and only one corresponding Co-net. Each Co-net has one and only one constraint that needs to be satisfied. Sometimes, however, a constraint does not have any corresponding Co-nets. This indicates collaboration among Co-Us is not required to satisfy the constraint. Any Co-net must include two or more Co-Us ($N(\Omega(r,t)) \geq 2$). Each Co-net is unique in a system at any time t and is identified by its index r. Two Co-nets r_1 and r_2 ($r_1 \neq r_2$) may include the same set of Co-Us ($\Omega(r_1,t) = \Omega(r_2,t)$), and are still different Co-nets because any two constraints in a system are different at any time t ($con(r_1,t) \neq con(r_2,t)$).

Definition 3: Conflict/Error (CE); Definition 4: Error; Definition 5: Conflict

Let $\xrightarrow{Satisfy}$ and $\xrightarrow[Dissatisfy]{}$ denote that Co-Us/Co-nets satisfy and dissatisfy constraints, respectively. Let $CE(r,t)$ represent any CE(s), which is (are) defined as:

$$\exists CE(r,t), \; iff \; con(r,t) \; is \; not \; satisfied, \quad \forall r, t \quad (3)$$

Let $E(u(r,i,t))$ and $C(n(r,t))$ represent an error and a conflict, respectively. They are defined in (4) and (5).

$$\exists E(u(r,i,t)), \; iff \; \vartheta(i,t) \xrightarrow[Dissatisfy]{} con(r,t), \quad con(r,t) \in \pi(i,t), \quad \forall r, i, t \quad (4)$$

$$\exists C(n(r,t)), \; iff \; \theta(r,t) \xrightarrow[Dissatisfy]{} con(r,t), \quad \forall r, t \quad (5)$$

With these definitions, states of CEs and how Co-Us in a network collaborate are defined for the purpose of CEPD

3 CEPD LOGIC

CEs are unavoidable in large networks with collaborative Co-Us. Figure 2 shows how to apply prediction and detection logic for Co-nets and Co-Us.

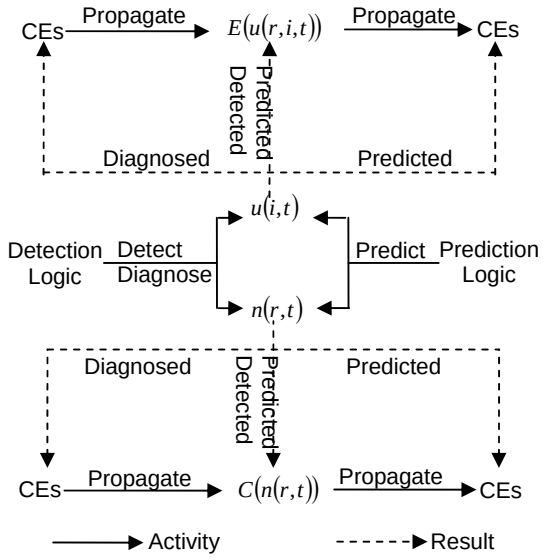


Figure 2: Apply the CEPD logic over a Co-U/Co-net

In Figure 2, CEs are predicted and detected at a Co-U or a Co-net by prediction logic and detection logic, respectively. Prediction logic is applied to analyze CE propagation to predict CEs that may occur at other Co-Us/Co-nets. Detection logic is applied to analyze CE propagation to identify causes (CEs at other Co-Us/Co-nets) of CEs.

3.1 Possible states of CEs

There are 16 possible states in which a CE is either detected (two states), not detected (two states), predicted (two states), not predicted (two states), identified as the cause of another CE (four states), or not identified as the cause of another CE (four states).

State 1: An error has been detected at time t.

$$\exists E(u(r,i,t_1)),\ \textit{such that } r,\ i,\ \textit{and } t_1 \textit{ are known at time } t_2,\quad \forall r, i, t_1, t_2, t, t_1 \le t_2 \le t \tag{6}$$

$t_1 \le t_2 \le t$ implies that t_1 is before or at t_2 and t_2 is before or at t. r, i, and t_1 are known at t because $t_2 \le t$.

State 2: An error has not been detected at time t.

$$\exists E(u(r,i,t_1)),\ \textit{such that at least one of } r \cup i \cup t_1 \textit{ is unknown at time } t_2,\quad \forall r, i, t_1, t_2, t, t_1 \le t \le t_2 \tag{7}$$

$t_1 \le t \le t_2$ implies that t_1 is before or at t and t is before or at t_2. At lease one of r, i, and t_1 is unknown at t because $t \le t_2$.

State 3: A conflict has been detected at time t.

$$\exists C(n(r,t_1)),\ \textit{such that } r \textit{ and } t_1 \textit{ are known at time } t_2,\quad \forall r, t_1, t_2, t, t_1 \le t_2 \le t \tag{8}$$

State 4: A conflict has not been detected at time t.

$$\exists C(n(r,t_1)),\ \textit{such that at least one of } r \cup t_1 \textit{ is unknown at time } t_2,\quad \forall r, t_1, t_2, t, t_1 \le t \le t_2 \tag{9}$$

State 5: An error has been predicted at time t.

$$\exists E(u(r,i,t_1)),\ \textit{such that } r, i \textit{ and } t_1 \textit{ are known at time } t_2,\quad \forall r, i, t_1, t_2, t, t_1 > t \ge t_2 \tag{10}$$

$t_1 > t \ge t_2$ implies that t_1 is after t and t is after or at t_2. r, i, and t_1 are known at t because $t \ge t_2$.

State 6: An error has not been predicted at time t.

$$\exists E(u(r,i,t_1)),\ \textit{such that at least one of } r \cup i \cup t_1 \textit{ is unknown at time } t_2,\quad \forall r, i, t_1, t_2, t, t_1 > t_2 \ge t \tag{11}$$

$t_1 > t_2 \ge t$ implies that t_1 is after t_2 and t_2 is after or at t. At lease one of r, i, and t_1 is unknown at t because $t_2 \ge t$.

State 7: A conflict has been predicted at time t.

$$\exists C(n(r,t_1)),\ \textit{such that } r \textit{ and } t_1 \textit{ are known at time } t_2,\quad \forall r, t_1, t_2, t, t_1 > t \ge t_2 \tag{12}$$

State 8: A conflict has not been predicted at time t.

$$\exists C(n(r,t_1)),\ \textit{such that at least one of } r \cup t_1 \textit{ is unknown at time } t_2,\quad \forall r, t_1, t_2, t, t_1 > t_2 \ge t \tag{13}$$

State 9: An error has been identified at time t as the cause of another error.

$$\begin{aligned}&\exists E(u(r_1,i_1,t_1)),\ E(u(r_2,i_2,t_2)),\ \textit{such that}\\ &E(u(r_1,i_1,t_1)) \xrightarrow{Cause} E(u(r_2,i_2,t_2)) \cap\\ &r_1, i_1, t_1, r_2, i_2, \textit{and } t_2 \textit{ are known at time } t_3\\ &\cap E(u(r_1,i_1,t_1)) \xrightarrow{Cause} E(u(r_2,i_2,t_2))\\ &\textit{is known at time } t_3,\quad \forall r_1, i_1, t_1, r_2, i_2, t_2, t_3, t,\\ &t_1 \le t_2 \le t_3 \le t,\ r_1 \ne r_2 \textit{ if } i_1 = i_2 \cap t_1 = t_2\end{aligned} \tag{14}$$

The item on the left side of $\xrightarrow{Cause}$ is the cause of the item on the right side. For instance, if $E(u(r_1,i_1,t_1))$ causes $E(u(r_2,i_2,t_2))$ directly or indirectly, $E(u(r_1,i_1,t_1))$ is the cause of $E(u(r_2,i_2,t_2))$ and this relationship is expressed as $E(u(r_1,i_1,t_1)) \xrightarrow{Cause} E(u(r_2,i_2,t_2))$.

State 10: An error has not been identified at time t as the cause of another error.

$$\begin{aligned}&\exists E(u(r_1,i_1,t_1)),\ E(u(r_2,i_2,t_2)),\ \textit{such that}\\ &E(u(r_1,i_1,t_1)) \xrightarrow{Cause} E(u(r_2,i_2,t_2)) \cap\\ &r_2, i_2, \textit{and } t_2 \textit{ are known at time } t_3 \cap\\ &\begin{pmatrix}\textit{at least one of } r_1 \cup i_1 \cup t_1 \textit{ is unknown at time } t_3 \cup\\ E(u(r_1,i_1,t_1)) \xrightarrow{Cause} E(u(r_2,i_2,t_2))\\ \textit{is unknown at time } t_3\end{pmatrix},\\ &\forall r_1, i_1, t_1, r_2, i_2, t_2, t_3, t, t_1 \le t_2 \le t_3 \le t,\\ &r_1 \ne r_2 \textit{ if } i_1 = i_2 \cap t_1 = t_2\end{aligned} \tag{15}$$

State 11: An error has been identified at time t as the cause of a conflict.

$$\begin{aligned}&\exists E(u(r_1,i,t_1)),\ C(n(r_2,t_2)),\ \textit{such that}\\ &E(u(r_1,i,t_1)) \xrightarrow{Cause} C(n(r_2,t_2)) \cap\\ &r_1, i, t_1, r_2, \textit{and } t_2 \textit{ are known at time } t_3\\ &\cap E(u(r_1,i,t_1)) \xrightarrow{Cause} C(n(r_2,t_2)) \textit{ is known at time } t_3,\\ &\forall r_1, i, t_1, r_2, t_2, t_3, t, t_1 \le t_2 \le t_3 \le t\end{aligned} \tag{16}$$

State 12: An error has not been identified at time t as the cause of a conflict.

$$\begin{aligned}&\exists E(u(r_1,i,t_1)),\ C(n(r_2,t_2)),\ \textit{such that}\\ &E(u(r_1,i,t_1)) \xrightarrow{Cause} C(n(r_2,t_2)) \cap\\ &r_2 \textit{ and } t_2 \textit{ are known at time } t_3 \cap\\ &\begin{pmatrix}\textit{at least one of } r_1 \cup i \cup t_1 \textit{ is unknown at time } t_3 \cup\\ E(u(r_1,i,t_1)) \xrightarrow{Cause} C(n(r_2,t_2)) \textit{ is unknown at time } t_3\end{pmatrix},\\ &\forall r_1, i, t_1, r_2, t_2, t_3, t, t_1 \le t_2 \le t_3 \le t\end{aligned} \tag{17}$$

State 13: A conflict has been identified at time t as the cause of an error.

$$\begin{aligned}&\exists C(n(r_1,t_1)),\ E(u(r_2,i,t_2)),\ \textit{such that}\\ &C(n(r_1,t_1)) \xrightarrow{Cause} E(u(r_2,i,t_2)) \cap\\ &r_1, t_1, r_2, i, \textit{and } t_2 \textit{ are known at time } t_3\\ &\cap C(n(r_1,t_1)) \xrightarrow{Cause} E(u(r_2,i,t_2)) \textit{ is known at time } t_3,\\ &\forall r_1, t_1, r_2, i, t_2, t_3, t, t_1 \le t_2 \le t_3 \le t\end{aligned} \tag{18}$$

State 14: A conflict has not been identified at time t as the cause of an error.

$$\begin{aligned}&\exists C(n(r_1,t_1)),\ E(u(r_2,i,t_2)),\ \textit{such that}\\ &C(n(r_1,t_1)) \xrightarrow{Cause} E(u(r_2,i,t_2)) \cap\\ &r_2, i, \textit{and } t_2 \textit{ are known at time } t_3 \cap\\ &\begin{pmatrix}\textit{at least one of } r_1 \cup t_1 \textit{ is unknown at time } t_3 \cup\\ C(n(r_1,t_1)) \xrightarrow{Cause} E(u(r_2,i,t_2)) \textit{ is unknown at time } t_3\end{pmatrix},\\ &\forall r_1, t_1, r_2, i, t_2, t_3, t, t_1 \le t_2 \le t_3 \le t\end{aligned} \tag{19}$$

State 15: A conflict has been identified at time t as the cause of another conflict.

$$\begin{aligned}&\exists C(n(r_1,t_1)),\ C(n(r_2,t_2)),\ \textit{such that}\\&C(n(r_1,t_1))\xrightarrow{Cause}C(n(r_2,t_2))\cap\\&r_1,\ t_1,\ r_2,\ \textit{and}\ t_2\ \textit{are known at time}\ t_3\\&\cap C(n(r_1,t_1))\xrightarrow{Cause}C(n(r_2,t_2))\ \textit{is known at time}\ t_3,\\&\forall r_1,\ t_1,\ r_2,\ t_2,\ t_3,\ t,\ t_1\le t_2\le t_3\le t,\ r_1\ne r_2\ \textit{if}\ t_1=t_2\end{aligned}\tag{20}$$

State 16: A conflict has not been identified at time t as the cause of another conflict.

$$\begin{aligned}&\exists C(n(r_1,t_1)),\ C(n(r_2,t_2)),\ \textit{such that}\\&C(n(r_1,t_1))\xrightarrow{Cause}C(n(r_2,t_2))\cap\\&r_2\ \textit{and}\ t_2\ \textit{are known at time}\ t_3\cap\\&\begin{pmatrix}\textit{at least one of}\ r_1\cup t_1\ \textit{is unknown at time}\ t_3\cup\\C(n(r_1,t_1))\xrightarrow{Cause}C(n(r_2,t_2))\ \textit{is unknown at time}\ t_3\end{pmatrix},\\&\forall r_1,\ t_1,\ r_2,\ t_2,\ t_3,\ t,\ t_1\le t_2\le t_3\le t,\ r_1\ne r_2\ \textit{if}\ t_1=t_2\end{aligned}\tag{21}$$

These 16 states clarify the difference between prediction and detection, and between prognostics and diagnostics. It is then necessary for CEPD to identify different types of collaboration in an e-Work network.

3.2 Types of e-Work collaboration: Illustrations

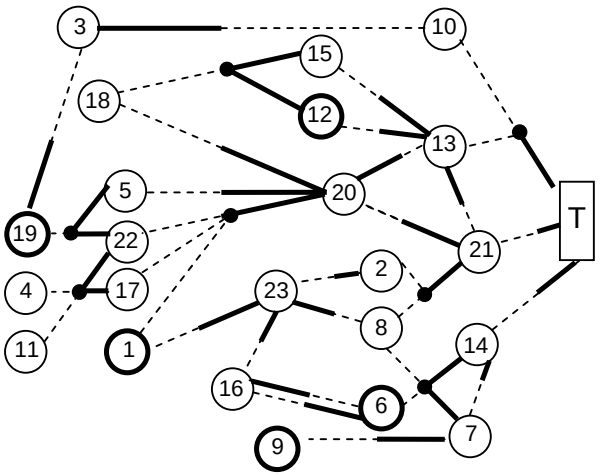
To detect CEs in industrial operations, it is important to understand network topologies and task dependencies. *Tasks* are products and services requested by customers. A network is task-driven and must complete tasks through Co-Us. A task may be divided into subtasks. Each subtask may be further divided into several other subtasks. A Co-U that is needed to complete a task is assigned one or more subtasks. Task dependencies are ways of collaboration among Co-Us. For instance, Figure 3 shows 23 Co-Us that are networked together according to task dependencies to complete a task requested by customer(s). Six types of collaboration are defined:

1. Cooperate and/or coordinate to Provide (CP). Co-Us collaborate (cooperate and/or coordinate) to provide products/services to other Co-Us. For instance, $u(4,t)$ and $u(11,t)$ collaborate to provide products/services to $u(17,t)$ and $u(22,t)$; $u(1,t)$, $u(17,t)$, and $u(22,t)$ collaborate to provide products/services to $u(17,t)$ and $u(20,t)$. A pair of parenthesis with a mark 'P' are used to denote this type of collaboration among Co-Us. The two examples given can be expressed as $(u(4,t),u(11,t))^P$ and $(u(1,t),u(17,t),u(22,t))^P$.
2. Cooperate and/or coordinate to Receive (CR). Co-Us collaborate to receive products/services from other Co-Us. For instance, $u(17,t)$ and $u(22,t)$ collaborate to receive products/services from $u(4,t)$ and $u(11,t)$; $u(5,t)$ and $u(22,t)$ collaborate to receive products/services from $u(19,t)$. Similiarly, using a pair of parenthesis with a mark 'R', the two examples given can be expressed as $(u(17,t),u(22,t))^R$ and $(u(5,t),u(22,t))^R$.
3. One-to-One (OO) dependency. If one Co-U provides products/services to the other Co-U, the type of collaboration between these two Co-Us is OO. For instance, the task dependency between $u(13,t)$ and $u(20,t)$ in Figure 3 is OO. $u(13,t)$ provides products/services to $u(20,t)$ and $u(20,t)$ is dependent on $u(13,t)$ to complete its subtasks. Two Co-Us may have two-way OO dependencies, e.g., task dependencies between $u(6,t)$ and $u(16,t)$. $u(6,t)$ provides products/services to $u(16,t)$ and $u(16,t)$ provides products/services to $u(6,t)$.
4. Many-to-One (MO) dependency. If Co-Us collaborate to provide products/services to the other Co-U, the collaboration between Co-Us that provide products/services and the Co-U that receives products/services is MO. Some examples are: (1) two Co-Us together provide 100 products every hour to the third Co-U; (2) two Co-Us provide same amount of product As and Bs, respectively, each day to the third Co-U; (3) three Co-Us provide products A, B, and C, respectively, to the fourth Co-U at the same speed; (4) two Co-Us provide continuous services, e.g., electricity and water, respectively, to the third Co-U to complete a task. In Figure 3, $u(1,t)$, $u(17,t)$, and $u(22,t)$ collaboratively provide products/services to $u(20,t)$.
5. One-to-Many (OM) dependency. If Co-Us collaborate to receive products/services from the other Co-U, the collaboration between Co-Us that receive products/services and the Co-U that provides products/services is OM. For instance, a Co-U provides products/services to two other Co-Us at the same speed, the collaboration between the first Co-U and two other Co-Us is OM because the two Co-Us that receive products/services must coordinate to maintain the same supply speed. There are two OM collaboration examples in Figure 3, e.g., the collaboration between $u(18,t)$ and $(u(12,t),u(15,t))^R$. The collaboration between a group of collaborative Co-Us and a single Co-U is either MO or OM.
6. Many-to-Many (MM) dependency. If a group of Co-Us collaborate to provide products/services to the other group of Co-Us which also collaborate to receive the products/services, the collaboration between Co-Us from different groups is MM. For instance, three Co-Us collaborate to provide the same amount of products to two other Co-Us. Figure 3 shows a example between $(u(4,t),u(11,t))^P$ and $(u(17,t),u(22,t))^R$.

The six types of collaboration are defined in terms of the need of collaboration and work flow. They can also be categorized according to the order of collaboration, i.e., first-order collaboration and high-order collaboration.

First-order collaboration exists between two Co-Us if one of two conditions are met: (1) the two Co-Us collaborate (cooperate and/or coordinate) to provide or receive products/services; (2) one of the two Co-Us directly provides products/services to the other Co-U. The six types of collaboration defined in Figure 3 are first-order collaboration. Let '$\rightarrow$' denote the first-order collaboration between two Co-Us, $U(i_1,t)$ and $U(i_2,t)$. $U(i_1,t)\rightarrow U(i_2,t)$ indicates that $U(i_1,t)$ provides products/services to $U(i_2,t)$.

High-order collaboration exists between two Co-Us if one of them indirectly provides products/services to the other. Both first-order collaboration and high-order collaboration may exist between two Co-Us at the same time. First-order collaboration is most concerned in CEPD and high-order collaboration is always analyzed through first-order collaboration.



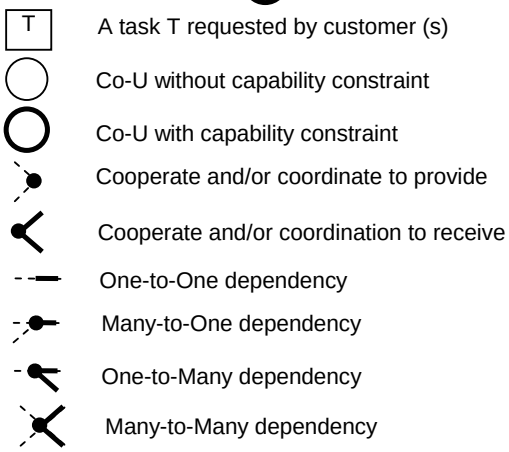


Figure 3: Collaboration among Co-Us

4 CEPD OVER E-WORK NETWORKS: CASE STUDY

To apply the CEPD logic for prognostics and diagnostics, it is necessary to describe various constraints which determine different types of collaboration among Co-Us. Constraints can be divided into two categories: capability constraints and task constraints. A *task constraint* determines what products/services a Co-U or a group of Co-Us needs to provide to the other Co-U(s). A *capability* constraint determines conditions a Co-U or a Co-net must meet. Table 1 describes examples of typical constraints.

Four e-Work networks are constructed in the case study (Fig. 4) to illustrate the proposed CEPD logic. Each network has 10 Co-Us. Co-Us in the linear network have *OO* dependency (Type 2 collaboration). Co-Us in the divergence network have *OM* dependency (Type 5 collaboration) and *CR* collaboration (Type 2). Co-Us in the convergence network have *MO* dependency (Type 4 collaboration) and *CP* collaboration (Type 1). Co-Us in the parallel network have *MM* dependency (Type 6 collaboration), and *CP* and *CR* collaboration.

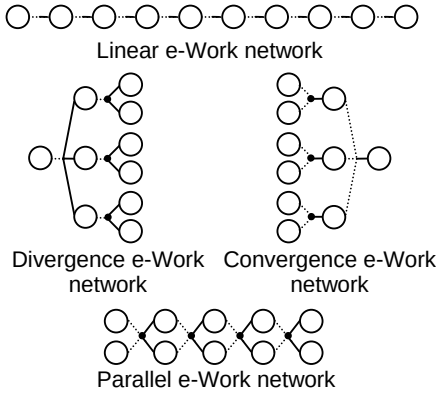


Figure 4: Four e-Work networks with ten Co-Us in each network

Constraints in each network that share the same one or more Co-Us are dependent on each other. The dependence between constraints is reflected by the dependability between Co-Us. Dependability is a real number between zero and one. For instance, if $u(2,t)$ is dependent on $u(1,t)$ with a dependability 0.75, $C(n(5,t_1))$ may occur with a probability 0.75 ($u(1,t) \in \Omega(5,t_1)$, $u(2,t) \in \Omega(5,t_1)$) if $E(u(6,1,t))$ occurs ($con(6,t) \in \pi(1,t)$).

Table 1: Constraint examples

Constraint	*Condition*	*From*	*To*	*Constraint Type*	*Co-U/Co-net Involved*
$con(3,10)$	Two units of part A in an hour	$u(6,10)$	$u(11,10)$	Task	$n(3,10)$, where $\Omega(3,10) = \{u(6,10), u(11,10)\}$ $u(6,10) \rightarrow u(11,10)$
$con(4,11)$	Air pressure between 2x10^5 and 3x10^5 Pascals	$u(6,11)$	$u(21,11)$, $u(1,11)$, $u(7,11)$	Task	$n(4,11)$, where $\Omega(4,11) = \{u(6,11), u(21,11), u(1,11), u(7,11)\}$ $u(6,11) \rightarrow (u(6,11), u(21,11), u(1,11), u(7,11))^R$
$con(2,7)$	Temperature is below 101 F°	$u(8,7)$	Not applicable	Capability	$u(8,7)$
$con(100,20)$	250 cars each day	$u(49,20)$, $u(38,20)$, $u(77,20)$	$u(33,20)$, $u(34,20)$	Task	$n(100,20)$, where $\Omega(100,20) = \begin{Bmatrix} u(49,20), u(38,20), u(77,20), \\ u(33,20), u(34,20) \end{Bmatrix}$ $(u(49,20), u(38,20), u(77,20))^P \rightarrow (u(33,20), u(34,20))^R$
$con(4,12)$	Monthly expense is less than $2 million	$u(7,12)$, $u(8,12)$	Not applicable	Capability	$n(4,12)$, where $\Omega(4,12) = \{u(7,12), u(8,12)\}$

An error is randomly assigned to one of 10 Co-Us in each network. The CEPD logic is applied to four networks to predict and detect CEs. One of the most important performance measures for CEPD logic is CEPD time, which is the sum of prediction, detection, and communication times. A simulation program is written in AutoMod 11.1 [13] to simulate the CEPD logic. There are two independent variables, network (four levels in Fig. 4) and dependability (nine levels, from 0.1 to 0.9 with an interval of 0.1), and one dependent variable, CEPD time. There are 36 combinations of two independent variables and 100 simulation runs are conducted for each combination. The CEPD time is recorded and the result is shown in Fig. 5.

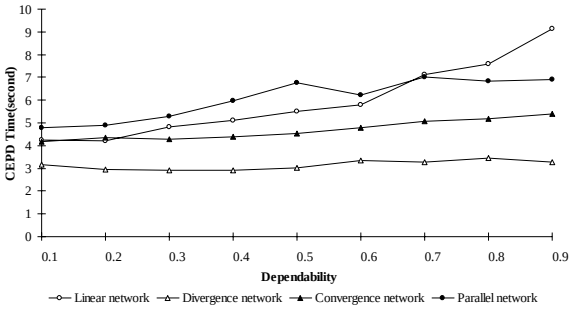


Figure 5: Mean CEPD time

The result in Fig. 5 shows that (1) the mean CEPD time for linear network and the mean CEPD time for parallel network are not significantly different at nominal level 0.05 (Table 2), and are significantly higher than the mean CEPD time for the divergence and convergence networks, and (2) the mean CEPD time for the divergence and convergence networks does not vary significantly as the dependability between Co-Us increases. These two desired properties exist because the proposed CEPD logic is an agent-based logic that can predict and detect CEs in parallel. When a network becomes more complex (more complex collaboration and increased parallel activities) and dependent (higher dependability between Co-Us), the CEPD time does not necessarily increase, and sometime even decreases.

Table 2: Multiple pair wise mean comparisons of CEPD time adjusted by Tukey procedure

Network	*Linear*	*Diver-gence*	*Conver-gence*	*Parallel*	*Level Mean (seconds)*
Linear		p< 0.0001	p< 0.0001	p= 0.6328	5.94
Diver-gence	p< 0.0001		p< 0.0001	p< 0.0001	3.15
Conver-gence	p< 0.0001	p< 0.0001		p< 0.0001	4.69
Parallel	p= 0.6328	p< 0.0001	p< 0.0001		6.08

5 DISCUSSION AND FUTURE RESEARCH

The 16 CE states presented in Section 3 help clarify the difference between prediction and detection, and between prognostics and diagnostics. Prognostics is the process to predict CEs with prediction logic through the analysis of CE propagation. CEs are detected by detection logic which also diagnoses CEs to identify their causes through the analysis of CE propagation. How to apply the CEPD logic over e-Work networks is illustrated in Figure 2.

The six types of collaboration describe relationships among Co-Us. A constraint defines a condition that a Co-U or a Co-net must satisfy and specifies collaboration types. The result of the case study with four e-Work networks shows that the proposed CEPD logic is robust in terms of CEPD time for complex networks with high dependability between Co-Us. The next step in this research is to improve the current CEPD logic to achieve better CEPD time, and study other performance measures, e.g.,, CE coverage ability, for different networks.

6 ACKNOWLEDGMENTS

This research has been developed by the PRISM Center with support from GMR project on "Design of Active Middleware for Error and Conflict Detection", and support from Kimberly Clark Corporation project on "Supply Networks Decision Support".

7 REFERENCES

[1] Yang C.-L., 2004, Conflict and error detection protocol with active middleware, MS Thesis, Purdue University, Aug., 2004.

[2] Klein B. D., 1997, How do actuaries use data containing errors? Models of error detection and error correction, Information Resources Management Journal, 10(4), 27-36.

[3] Barber K. S., Liu T. H., Ramaswamy S., 2001, Conflict detection during plan integration for multi-agent systems, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 31(4), 616-628.

[4] Klein M., 1992, Detecting and resolving conflicts among cooperating human and machine-based design agents, Artificial Intelligence in Engineering, 7(2), 93-104.

[5] Li X., Zhou X., Ruan X., 2002, Conflict management in closely coupled collaborative design system, International Journal of Computer Integrated Manufacturing, 15(4), 345-352.

[6] Shin M., Cha Y., Ryu K., Jung M., 2006, Conflict detection and resolution for goal formation in the fractal manufacturing system, International Journal of Production Research, 44(3), 447-465.

[7] Ceroni J. A., Velasquez A. A., 2003, Conflict detection and resolution in distributed design, Production Planning and Control, 14(8), 734-742.

[8] Jiang T., Nevill Jr G. E., 2002, Conflict cause identification in web-based concurrent engineering design system, Concurrent Engineering Research and Applications, 10(1), 15-26.

[9] Lara M. A., Nof S. Y., 2003, Computer-supported conflict resolution for collaborative facility designers, International Journal of Production Research, 41(2), 207-233.

[10] Chiang L. H., Braatz R. D., Russell E., 2001, Fault detection and diagnosis in industrial systems, London, New York, Springer.

[11] Raich A., Cinar A., 1996, Statistical process monitoring and disturbance diagnosis in multivariable continuous processes, AIChE Journal, 42(4), 995-1009.

[12] Tu F., Pattipati K. R., Deb S., Malepati, V. N., 2003, Computationally Efficient Algorithms for Multiple Fault Diagnosis in Large Graph-based Systems, IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans, 33(1), 73-85.

[13] AutoMod Version 11.1, 1998-2003, Brooks Automation, Inc.