Reconfiguration Mechanism for Holonic Manufacturing Systems**

Fu-Shiung Hsieh*

* Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taiwan
(e-mail: fshsieh@cyut.edu.tw)

** This paper is supported in part by National Science Council of Taiwan under grant NSC95-2416-H-324-009-MY2

Abstract: Although holonic manufacturing systems (HMS) have been recognized as a paradigm to cope with changes in manufacturing environment based on a flexible architecture, several characteristics of HMS such as re-configurability have not been characterized quantitatively. To realize these advantages, more concrete mechanisms or methodologies need to be developed. The objectives of this paper are to probe into the characteristics of reconfigurability for HMS and propose design methodologies to realize these characteristics. We formulate and study a holarchy reconfiguration problem to deal with dynamic removal/addition of holons in HMS due to failure/recovery of resources. Finding a solution from the scratch to deal with resource failures is not an appropriate approach as it may lead to chaos at the shop floor. To achieve effective reconfiguration in HMS, a viable solution must be based on the nominal configuration and cooperation of holons. We define an impact function to characterize the impact of changes due to different holons in a holarchy. A reconfiguration mechanism based on the impact function is proposed to effectively reconfigure the systems to achieve minimal cost solutions.

1. INTRODUCTION

Since its invention, holonic manufacturing systems (HMS) (Balasubramanian 2001; Wyss, 1999; Van Brussel et al. 1998; Christensen 1994) have been recognized as a paradigm to accommodate changes and meet customers’ requirements dynamically. HMS is based on the notion of holon (Koestler, 1967) to provide a reconfigurable, flexible and decentralized manufacturing architecture. Koestler proposed the word “holon” to describe a basic unit of organization in biological and social systems. A holon is an autonomous, co-operative and intelligent entity able to collaborate with other holon to process the tasks. In HMS, a holon can be part of another holon and a set of holons that autonomously cooperate to achieve a goal forms a holarchy. The loosely coupled structure of holons makes it possible to attain for manufacturing systems the benefits of stability in the face of disturbances, adaptability and flexibility in the face of change, and efficient use of available resources.

In spite of the promising perspective and the research developed by the holonic community (Brennan, Fletcher and Norrie, 2002; Doen et al., 2003), HMS leave some important questions open (Marik, 2004), namely how to achieve global optimisation in decentralised systems? how should the production control structure evolve to adapt to change? how to specify formally the dynamic behaviour of holonic systems? how to introduce self-organisation capabilities? and how to develop holonic-based control applications? In existing literature, works on how to specify formally the dynamic behaviour of holonic systems appear in (Leitão, Colombo and Restivo, 2003; Hsieh, 2006). Existing results on how to achieve global optimisation in decentralised systems can be found in (Hsieh 2008). Studies on how should the production control structure evolve to adapt to changes have been made by Leitão et al. (Leitão & Restivo, 2006). Leitão et al. present an agile and adaptive manufacturing control architecture that addresses the need for the fast reaction to disturbances at the shop floor level, increasing the agility and flexibility of the enterprise.

Despite the existing research results, there is still a lack of study on how to design an effective reconfiguration mechanism taking advantage of the holonic system architecture. The goal of this research aims at proposing a systematic design methodology to develop reconfiguration mechanism for HMS to deal with changes in holons, including dynamic removal/addition of holons in HMS. While the process to handle removal of holons involves identifying of the parts of holarchy influenced and replacing them by alternative holons, the process to handle addition of holons only involves searching for alternative holons. Therefore, we will focus on the mechanism to handle removal of holons in this paper. Failure and recovery of resources are two common events leading to removal/addition of holons in HMS. Resources may fail at any point in time, reduce the set of available resources, take place in any holarchy and change its characteristics. For some types of tolerable resource failures, the operation of a holarchy may still be maintained. In this case, reconfiguration may not be required. An interesting question is how to quantitatively characterize the condition under which reconfiguration is not required. The effects of resource failures may propagate from one holon to another. Development of an effective method to analyze the impact of resource failures on the operation of an HMS is a significant issue. Another question is to establish the condition under which an HMS can be dynamically reconfigured to accommodate changes due to resource failures. A challenge in this research direction is due to the distributed architecture of HMS, which limits the availability of information that can be used to reconfigure the system. How to design a reconfiguration mechanism based on the architecture of HMS is a significant issue. The key question of interest is: how to reconfigure HMS to cope with resources failures cost-effectively without causing chaos at the shop floor. Answers to the above questions are crucial to the development of an effective reconfiguration mechanism. Finding a solution from the scratch from the point resource failures occur is not an appropriate approach as it may lead to chaos at the shop floor. An effective approach to reconfiguration must be based on
the nominal solution so that the impact on the operation of the other existing active holarchies can be significantly reduced. To study the reconfiguration mechanism, we define a holarchy reconfiguration problem (HRP), which is based on the holarchy optimization problem (HOP) in (Hsieh, 2008). We study the condition for the existence of a solution to HRP. To propose a solution methodology, a hybrid model that combines contract net protocol (CNP) (Smith, 1980) with Petri nets (Murata, 1989) for HOP has been proposed (Hsieh, 2008). Based on a similar modeling technique, we formulate and study the HRP problem. We propose Petri net models for the holarchies formed in HMS and study the impact of resource failures on the system. We define an impact function to characterize the impact of resource failures on a holarchy. We study the tolerable failure conditions for HMS to survive without reconfiguration. In case the resource failures do not satisfy the condition, the reconfiguration algorithm based on the impact function is initiated to reconfigure the systems to achieve minimal cost solutions while minimizing the impact on the existing tasks at the shop floor.

To achieve effective reconfiguration in HMS, a viable solution must be based on cooperation of holons. The reconfiguration algorithm relies on the interactions and cooperation between four types of holons: detector, initiator, standby and optimizer. A detector is a resource holon that detects resource failures. There are two types of initiators: Type I initiator and Type II initiator. A Type I initiator is a holon to be removed from the system and it usually contains a detector and the operation will be blocked eventually due to resource failures. A Type 2 initiator is a holon to be dynamically added to the system. While the reconfiguration process triggered by Type I initiators to remove holons from a system and is broken down into two stages: (1) initiator identification & standby notification and (2) process optimization, the reconfiguration process triggered by Type II initiators is simpler and is broken down into two stages: (1) initiator identification and (2) process optimization. For this reason, we will focus on the development of reconfiguration mechanism for Type I initiator in this paper to save space.

A Type I initiator will forward standby request to the upstream composite holons which are called standby holons. A standby holon will stay in pending state and may attempt to submit a proposal and establish a contract with another downstream composite holon within a time period again. If no new contract is established by the time period, the standby holon will restore their resources back to the idle state. An optimizer is the downstream holon of the manager of the initiator responsible for finding the optimal configuration based on the remaining resources. We establish a condition for the existence of an optimal solution to HRP.

This paper is differentiated from the works of Leitão et al. (Leitão, Colombino and Restivo, 2003; Leitão and Restivo, 2006) in several aspects. Although all these works intend to contribute to the development of a dynamic and adaptive control approach that improves the agility and reaction to unexpected disturbances by taking advantage of the flexibility offered by HMS without compromising the global optimisation, the approaches are different. The self-organization adaptation mechanism proposed by Leitão et al. introduces the concept of autonomy factor and a pheromone-like spreading mechanism to propagate emergence and reorganize the system. Our approach first checks whether resource failures are tolerable. If the resource failures are tolerable, no reconfiguration is required. Otherwise, only the composite holons influenced by the resource failures will be notified by a standby request. In addition, as our strategy achieves reconfiguration cost-effectively based on the nominal solution, it minimizes the impact on existing tasks and avoids chaos in the shop floor.

The remainder of this paper is organized as follows. In Section 2, we introduce the reconfiguration problem in HMS. In Section 3, we review Petri net models for holarchies. In Section 4, we study the impact of resource failures. In Section 5, we propose our reconfiguration algorithm. In Section 6, we conclude this paper.

2. HOLARCHY RECONFIGURATION PROBLEM

The dynamics and behaviours of HMS are determined by the interactions of three types of holons: resource holons, product holons, and order holons. A resource holon consists of a production resource with relevant components to control the resource. A product holon contains the production process information to manufacture products. An order holon represents an order. Individual product holons or resource holons cannot process a complex task alone. To process a task, a set of resource holons and product holons form a holarchy. In HMS, a holon can be part of another holon and a set of holons that can autonomously cooperate to achieve a goal forms a holarchy. Holarchy formation problem has been studied in (Hsieh; 2008) to propose models and develop collaborative algorithms to guide the holons to form a holarchy that coherently moves toward the desired goal state ultimately. A holarchy consists of a set of composite holons, where a composite holon consists of a product holon and a set of resource holons. Fig. 1(a) is a diagram that shows the dependency between holons, where each node denotes a composite holon. For a node, an outgoing arc labelled with a number i means the corresponding holon may provide type-i tasks to other holons. An incoming arc labelled with a number i means the holon may accept type-i tasks from other holons.

A composite holon is denoted as $c_n(U_n, R_n)$ and may depend on a set of upstream composite holons $U'_n \subseteq U_n$ that provide parts to it and a set of resource holons $R'_n \subseteq R_n$, where we use $U_n$ to denote the set of potential upstream composite holons of $c_n$ and $R_n$ to denote the set of potential resources holons of $c_n$, respectively.

In Fig. 1(a), $U_7 = \{1,2\}$, $U_9 = \{3,4,5,6\}$. We use $U_n(i)$ to denote the set of potential upstream composite holons that provide type-i tasks to $c_n$, $c_n(U_n, R_n)$. A potential composite holon $c_{n,r}$, where $n' \in U_{n'}$, plays the role of a bidder and submits a proposal $\pi_{n,r}$ to product holon $h_{n,r}$ to indicate the part types provided and the related timing information. A resource holon $h_{n,r}$, $r \in R_n$, may submit a proposal $\pi_{n,r}$ to product holon $h_{n,r}$ to indicate its ability, the number of resources to be allocated, the timing and costs to execute the required operations by $h_{n,r}$. Let $\Pi_n(S)$ denote the set of all proposals submitted to the manager $h_{n,r}$ under state $S$. Let $w_n$ denote the cost function that specifies the cost of $c_n(U_n, R_n)$. A composite holon is feasible if there exists an execution sequence to accomplish the desired task. The holarchy optimization problem can be stated informally as follows: Holarchy optimization Problem (HOP):

$$\min w_n = \min \left( \sum_{R_n \subseteq R_n} \sum_{R_n \subseteq R_n} w_{n,r} \right)$$

s.t. $c_n(U_n, R_n)$ is feasible, $R_n \subseteq R_n$ and $U_n \subseteq U_n$. 

15793
3.2: Petri net

\[ H_n = (P_n, T_n, F_n, m_n, SI_n) \]

where \( P_n \) is a finite set of places, \( T_n \) is a finite set of transitions, \( F_n \subseteq (P_n \times T_n) \) is the flow relation, \( m_n : P_n \rightarrow Z_{\geq 0} \) is the initial marking of the \( P_n \) with \( Z \) as the set of non-negative integers and \( SI_n : T_n \rightarrow R^+ \times R^+ \) is a mapping called the static interval that specifies the earliest firing time and the latest firing time for each transition. A marking of \( G \) is a vector \( m \in Z_{\geq 0}^U \) that indicates the number of tokens in each place under a state.

**Definition 3.1:** The workflow of product holon \( h_n \) is an acyclic marked graph \( H_n = (P_n, T_n, F_n, m_n, SI_n) \). As each transition represents a distinct operation in a task, \( T_j \cap T_k = \emptyset \) for \( j \neq k \).

An activity is a sequence of operations to be performed by a certain type of resources. \( \Gamma_n \) denotes the set of resource types required to perform the operations of \( h_n \). Let \( \gamma \) denote the idle state place of type- \( \gamma \) resources. The \( k-th \) activity is described by either a circuit place that starts and ends with an idle state place \( \gamma \) or a directed path that starts with an idle state and ends with a non-idle state place in Petri net. A circuit indicates that the resource activity includes resource allocation and de-allocation. A directed path indicates that the resource activity includes either resource allocation or de-allocation but not both. The Petri net model for the \( k-th \) activity for a type- \( \gamma \) resources, where \( \gamma \in \Gamma_n \), is described by a Petri net \( H_n^\gamma \) as defined in [3].

**Definition 3.2:** Petri net \( H_n^\gamma = (P_n^\gamma, T_n^\gamma, F_n^\gamma, m_n^\gamma, SI_n^\gamma) \) denotes
the \( k \)-th activity for a type-\( \gamma \) resource, where \( \gamma \in \Gamma_n \). Remark that \( T^+_k \cap T'^+_k = \emptyset \) for \( k \neq k' \).

Let \( K_\gamma \) be the number of activities of a type-\( \gamma \) resource.

Let \( \Omega^\gamma \subseteq \{1,2,3,...,K_\gamma\} \) denote the set of type-\( \gamma \) activity IDs in \( H_n \).

The initial marking \( m_0^\gamma \) is determined based on the set \( \Pi_n \) of proposals submitted to product holon \( h_n \). More specifically, \( m_0^\gamma = \sum_{\pi \in \Pi_n, \pi \in \Omega^\gamma} \pi^\gamma \), where \( \gamma \) is the type-\( \gamma \) resource idle state place.

To capture the interactions between a product holon \( h_n \) and resource holons, we define operator \( \| \) to combine two Petri nets into a new one by merging common places, transitions, or arcs.

**Definition 3.3**: Composite holon \( c_n(U_n, R_n) \) that depends on the set \( U_n \) of composite holons is modeled by \( C_n = \|_{\gamma \in \Gamma_n} H_\gamma \| H_n \), where \( H_\gamma = \sum_{\pi \in \Omega^\gamma} \pi^\gamma \). We use \( C_n(m_0^\gamma, u_n) \) to denote the Petri net model of \( c_n(U_n, R_n) \), where \( m_0^\gamma = \sum_{\pi \in \Omega^\gamma} \pi^\gamma \) and \( u_n \) is a control policy of \( C_n \).

Fig. 2 shows examples of \( C_n \).

Let \( N_A \) denote the set of composite holons in a holarchy \( A \).

**Definition 3.4**: The overall Petri net model for \( A \) is constructed based on composite holon \( C_n \) \( \forall n \in N_A \) by \( C_A = \|_{n \in N_A} C_n = (P_A, T_A, F_A, m_{A0}, SL_A, u_A) \), where \( u_A \) is a control policy of \( C_A \).

A holarchy \( A \) is feasible if there exists a control policy \( u_A \) under which \( C_A(m_{A0}, u_A) \) is live. The following property provides a condition to check whether holarchy \( A \) is feasible.

**Property 3.1**: There exists a control policy \( u_A \) under which \( C_A(m_{A0}, u_A) \) is live if and only if there exists a marking \( m'_{A} \) that can be reached from \( m_{A0} \) and \( m'_{A} \geq m_{A0}^* \).

Fig. 3 shows examples of \( \hat{C}_n \). Liveness condition of \( \hat{C}_n \) can be characterized as follows.

**Definition 3.5**: Let \( \hat{C}_n(m_{n0}, u_n) = (\hat{P}_n, \hat{T}_n, \hat{F}_n, \hat{m}_{n0}, \hat{SL}_n, u_n) \) be the Petri net that augments \( C_n \) by adding an output transition \( t'_n \) to the terminal output place in \( h_n^* \) and adding an arc from \( t'_n \) to each resource place without input transition, where \( u_n \) is a control policy of \( C_n \).

Fig. 3 shows examples of \( \hat{C}_n \). Liveness condition of \( \hat{C}_n \) can be characterized as follows.

**Definition 3.6**: \( \hat{M}_{n0} \) denotes the set of minimal markings under which there exists a control policy \( u_n \) under which \( \hat{C}_n(m_{n0}, u_n) \) is live.

**Property 3.2**: There exists a control policy \( u_n \) under which \( \hat{C}_n(m_{n0}, u_n) \) is live if and only if the initial marking \( \hat{m}_{n0} \geq \hat{m}_{n0}^* \), where \( \hat{m}_{n0}^* \in \hat{M}_{n0} \).

In addition to the above properties, the following property establishes a relation between \( m_{A}^* \) and \( m_{A0}^* \).

**Property 3.3**: An upper bound of is \( m_{A}^*(\gamma) \leq \sum_{n \in N_A} \hat{m}_{n}^*(\gamma) \forall \gamma \in \Gamma_A \).

4. IMPACT OF RESOURCE FAILURES

To realize our proposed scheme for reconfiguration, we develop a method to identify whether a holarchy is a reconfiguration initiators. By definition, a reconfiguration initiators is a holarchy in which resource failures occur and influence the operation. Resource failures in holarchy \( c_n \) can be represented by a perturbation \( \delta_n \) in marking \( m_n \) of \( C_n \). To characterize the impact of resource failures, we need the following definition.

**Definition 4.1**: Let \( N_A \) denote the set of composite holons in the system. Suppose there exist control
If there does not exist control policies $u'_n \forall n \in N_A$ such that $C_n$ can be kept live under $u'_n$, $\forall n \in N_A$. $C_n$ is influenced by $\delta_n$ if there does not exist control policies $u'_n \forall n' \in N_A$ such that $C_n$ can be kept live under $u'_n$ from marking $m'_n$ after removing $\delta_n$ from $m'_n$. We use the impact function $I(\delta_n, C_n)$ to denote whether $C_n$ is influenced by $\delta_n$.

**Definition 4.2:** The impact function is defined by

$$ I(\delta_n, C_n) = \begin{cases} 1 & \text{if } C_n \text{ is influenced by } \delta_n \\ 0 & \text{otherwise} \end{cases} $$

**Property 4.1:** $C_n$ is an initiator for a given failure $\delta_n$ if $I(\delta_n, C_n) = 1$.

Occurrence of $\delta_n$ may block the operation of $C_n$ and the effects may ripple backward to the upstream of $C_n$. The following property states the ripple effects of $\delta_n$.

**Property 4.2:** If $I(\delta_n, C_n) = 1$, $I(\delta_n, C_n') = 1 \forall n' \in U_n$.

Example: For Fig. 7, suppose $I(\delta_8, C_4) = 1$ and $I(\delta_8, C_1) = 1$.

The set $[n]' \in U_n$, $I(\delta_n, C_n) = 1, n' \in U_n$, $I(\delta_n, C_n') = 1$ denotes the standby holons with respect to $\delta_n$.

**Property 4.3:** If $I(\delta_n, C_n) = 2$, $I(\delta_n, C_n') = 0 \forall n \in N_A$.

Although $I(\delta_n, C_n)$ provides a way to determine whether a holarchy $C_n$ is an initiator with respect to $\delta_n$, due to computational complexity, an approximate impact function $\tilde{I}(\delta_n, C_n)$ which satisfies $\tilde{I}(\delta_n, C_n) \geq I(\delta_n, C_n)$ $\forall n, n' \in N_A$ is defined based on the potential resource release function $PR_n(\bullet)$ below. The function $PR_n(\bullet)$ is defined based on a mathematical model $AC_n$ that combines the model of product holons with the resource holons involved to compute the number of resources that will return to idle state.

**Definition 4.3:** Let $\tilde{H}^k_n = (\tilde{T}^k_n, \tilde{F}^k_n, \tilde{m}^k_n, \tilde{S}^k_n)$ denotes the acyclic Petri net model for the $k$–th activity of a type-$\gamma$ resources, $\gamma \in \Gamma_n$.

Remark that $\tilde{T}^k_n \cap \tilde{T}^k_n = \Phi$ for $k \neq k'$. $\tilde{H}^k_n$ is constructed by splitting each resource idle place in $H^k_n$ into two places: one resource allocation place $\gamma_i^k$ and the other resource de-allocation place $\gamma_o^k$. $AC_n = \prod \gamma \in \Gamma_n \tilde{H}^k_n | H_n$, where $\tilde{H}^k_n = \prod k \in N_A \tilde{H}^k_n$.

Fig 4 shows examples of $AC_n$ corresponding to $C_n$ in Fig. 2.

To compute the number of tokens that will return to idle state place in $AC_n$, we define a token flow path as follows.

**Definition 4.4:** A token flow path $\lambda$ consists of the sequence of places in a directed path of $AC_n$. $L_n(\lambda(p)$ denotes the set of token flow paths in $AC_n$ ending with place $p$. The total number of tokens in $\lambda$ of $AC_n$ under submarking $\tilde{m}_n$ is denoted as $\lambda(\tilde{m}_n) = \sum_{p \in \lambda} \tilde{m}_n(p)$, where $p \in \lambda$ means that $p$ is a place in $\lambda$.

The number of type-$\gamma$ resources that will return to the set of places $\{\gamma_i^k\}$ by firing all the transitions in $AC_n$ as many times as possible under $\tilde{m}_n$ is $\sum_{k \in N_A, \lambda \in L_n(\gamma_i^k)} \lambda(\tilde{m}_n)$.

**Fig. 4 Petri net Models of $AC_n$**

**Definition 4.5:** The potentially released resources from $AC_n$ is represented by a vector $PR_n$, where $PR_n \in Z_{\geq 1}$ and $PR_n(\gamma, \tilde{m}_n)$ denotes the number of type-$\gamma$ resources potentially released from $\tilde{m}_n$.

A lower bound on $PR_n(\gamma, \tilde{m}_n)$ is $PR_n(\gamma, \tilde{m}_n) = \sum_{k \in N_A, \lambda \in L_n(\gamma_i^k)} \lambda(\tilde{m}_n)$.

For Fig. 4, $PR_{\gamma_1}(\gamma_1) = PR_{\gamma_1}(\gamma_2) = 1$, $PR_{\gamma_3}(\gamma_1) = 1$, $PR_{\gamma_3}(\gamma_2) = 2$, $PR_{\gamma_4}(\gamma_1) = 1$, $PR_{\gamma_4}(\gamma_2) = 2$, $PR_{\gamma_5}(\gamma_1) = 1$, $PR_{\gamma_5}(\gamma_2) = PR_{\gamma_5}(\gamma_3) = 1$, $PR_{\gamma_6}(\gamma_1) = 1$, $PR_{\gamma_6}(\gamma_2) = 1$.

Property 4.4 states a condition for a set of holarchies $N_A = \{1,2,3,\ldots,N\}$ to maintain feasibility collaboratively.

**Property 4.4:** Suppose $N_A = \{1,2,3,\ldots,N\}$.

If $PR_{\gamma}(\sum_{n \in N_A} PR_{\gamma}(\gamma, \tilde{m}_n)) \geq m_{\gamma}^*(\gamma) \forall \gamma \in \Gamma_A$, there exists a control policy under which $C_n$ is feasible for each $n \in N_A$.

For Fig. 4 and Fig. 5, $N_A = \{1,3,5,8,9,12\}$.

$PR_{\gamma_1}(\gamma_1) = 2$, $PR_{\gamma_2}(\gamma_2) = 2$, $PR_{\gamma_3}(\gamma_1) = 1$, $PR_{\gamma_3}(\gamma_2) = 2$, $PR_{\gamma_4}(\gamma_1) = 2$, $PR_{\gamma_4}(\gamma_2) = 1$, $PR_{\gamma_5}(\gamma_1) = 3$, $PR_{\gamma_5}(\gamma_2) = 1$.

Obviously, $PR_{\gamma}(\sum_{n \in N_A} PR_{\gamma}(\gamma, \tilde{m}_n)) \geq m_{\gamma}^*(\gamma) \forall \gamma \in \Gamma_A$.

There exists a control policy under which $C_n$ is feasible for each $n \in N_A$.

We first characterize the condition under which $I(\delta_n, C_n) = 0$ and then establish a condition.
for $\bar{I}(\delta_n, C_n) = 1$. $\bar{I}(\delta_n, C_n) = 0$ corresponds to the conditions under which $\delta_n$ is tolerable. Let $\Delta_n(m_n)$ denote the set of tolerable resource failures for $C_n$ under marking $m_n$.

$$\Delta_n(m_n) = \{\delta_n: \sum_{p \in N_A} PR_n(y, \bar{m}_n - \delta_n) \geq \bar{m}_n^*(y) \forall y \in \Gamma_A\}.$$

We have the following property.

Property 4.5: If $\delta_n \in \Delta_n(m_n)$, $\bar{I}(\delta_n, C_n) = 0 \forall n' \in A_n$.

Corollary 4.1: If $\delta_n \in \Delta_n(m_n)$, $\bar{I}(\delta_n, C_n) = 0 \forall n' \in A_n$. All the holarchies can maintain their operations and optimality without reconfiguration.

Theorem 4.1: If $\bar{I}(\delta_n, C_n) = 0 \forall n' \in A_n$, all the holarchies can maintain their operations and optimality without reconfiguration.

To characterize the condition for $\bar{I}(\delta_n, C_n) = 1$, we define the set of initiators $X$ as follows.

Definition 4.6: Let $X = \{n: \delta_n > 0, n' \in A_n\} \setminus \{n\}$

$$\sum_{n \in N_A} PR_n(y, \bar{m}_n - \delta_n) \geq \bar{m}_n^*(y) \forall y \in \Gamma_A.$$

Example: For Fig. 4, suppose $\delta_n(p_3) = 1$, $\delta_n(p) = 0 \forall p \neq p_3$, and $\delta_n(p) = 0 \forall p \in P_n, \forall n \neq 8$. In this case, $X = \{8\}$.

The approximate impact function $\bar{I}(\delta_n, C_n)$ is defined based on the set of initiators $X$ by setting $\bar{I}(\delta_n, C_n) = 1$ for each $n \in X$.

Occurrence of $\delta_n$ may block the operation of $C_n$ and the effects of blocking may ripple backward to the upstream of $C_n$. The following property states the ripple effects of $\delta_n$ is valid for $\bar{I}(\delta_n, C_n)$.

Property 4.6: If $\bar{I}(\delta_n, C_n) = 1 \forall n' \in U_n$.

Based on the set of initiators $X$, it follows from Property 4.6 that Property 4.7 holds for $\bar{I}(\delta_n, C_n)$.

Property 4.7: $\bar{I}(\delta_n, C_n) = \begin{cases} 1 & \text{if } n' \in B \cup X \\ 0 & \text{otherwise} \end{cases}$, where $B$ satisfies the following properties.

(i) For each $n \in X$, $U_n \subseteq B$.

(ii) For each $n \in B$, $U_n \subseteq B$.

Example: For Fig. 4, suppose $\delta_n(p_3) = 1$, $\delta_n(p) = 0 \forall p \neq p_3$, and $\delta_n(p) = 0 \forall p \in P_n, \forall n \neq 8$. In this case, $X = \{8\}$ and $B = \{4\}$.

Example: Suppose $\delta_n(p_3) = 1$, $\delta_n(p) = 0 \forall p \neq p_3$, and $\delta_n(p) = 0 \forall p \in P_n, \forall n \neq 8$.

$$\bar{I}(\delta_n, C_n) = 0 \forall n' \in N_A \setminus \{1, 8\}$$

$$\bar{I}(\delta_n, C_n) = 1 \text{ and } \bar{I}(\delta_n, C_1) = 1.$$

5. RECONFIGURATION ALGORITHM

The reconfiguration algorithm is based on the impact function $\bar{I}$. The impact function $\bar{I}(\delta_n, C_n)$ characterizes whether a composite holon $C_n$ is influenced by $\delta_n$.

If $\bar{I}(\delta_n, C_n) = 0$, $\delta_n$ has no influence on the operation of composite holon $C_n$. In this case, it is not required to reconfigure $C_n$. Each composite holon $C_n$ with $\bar{I}(\delta_n, C_n) = 1$ must be reconfigured properly. Otherwise the composite holon may eventually be blocked. We classify the set of holons involved in the reconfiguration process into three types: initiator, standby and optimizer. A composite holon $C_n$ is an initiator with respect to $\delta_n$ if $\bar{I}(\delta_n, C_n) = 1$.

Each composite holon in $X$ is an initiator whereas each composite holon in $B$ is a standby. An optimizer is a composite holon responsible for optimization of the reconfigured process. The reconfiguration processes depend on the collaboration of the initiator, standsys and the optimizers. The reconfiguration process is triggered by a resource failure $\delta_n$ and is broken down into two stages: (1) initiator identification & standby notification and (2) process optimization. To determine whether $n' \in X$, we define a resource monitor holon to maintain the up-to-date information of $PR_n(y, \bar{m}_n - \delta_n)$ for each $n \in N_A$. A composite holon that has detected resource failures $\delta_n$ must determine whether it is an initiator by sending a request to the resource monitor holon to obtain $PR_n(y, \bar{m}_n) \forall y, n \in N_A$.

The following algorithm is applied to identify initiator and standby holons.

Algorithm to Identify Initiator and Notify Standby Holons

Given $c_n(U_n, R_n), \delta_n, N_A, \bar{m}_n$

Step 1: Send $PR_n(y, \bar{m}_n - \delta_n)$ to Resource Monitor

Step 2: Request Resource Monitor to provide $PR(y) = \sum_{n \in N_A} PR_n(y, \bar{m}_n - \delta_n)$ each for $y \in \Gamma_A$.

Step 3: Identify Initiator and Notify Standby Holons

If $PR(y) \geq \bar{m}_n^*(y) \forall y \in \Gamma_A$

$$\bar{I}(\delta_n, C_n) = 0$$

Exit

Else

$$\bar{I}(\delta_n, C_n) = 1$$

For each $n' \in U_n$

Send Standby Request to $C_n'$

Wait for Response from $C_n'$

End For

For each $r \in R_n$

Restore resources to idle state

End For

Send Optimization Request to the primary optimizer

End If

An initiator sends a Standby Request to bring active relevant composite holons to standby state and to cancel the previously established contract. After finalizing the initiator identification and notification of standys, each initiator sends an optimization request to the downstream composite

15797
holon which takes the role of an optimizer and proceeds to
the process optimization stage.
Example: Suppose \( \delta_k(p) = 1 \), \( \delta_k(p) = 0 \) and \( \delta_k(p) = 0 \) \( \forall p \in P_n \). In this case, \( \delta_k(p) = 1 \).
According to Property 4.6, \( \delta_k(C_t) = 1 \). This case, \( C_t \) is an an initiator as shown in Fig. 1(c), \( C_t \) is a standby and \( C_t \) is an optimizer.
In the optimization process, there are two types of optimizers: primary optimizer and secondary optimizer. There is only one primary optimizer, which is the manager of the initiator. All the other optimizers are secondary optimizers. The optimization process is started by the primary optimizer. The optimization process propagates from the primary optimizer to its upstream. The primary optimizer first finds the set of required part types \( I_R = \{ \text{TYPE}(p) | p \in h^*_k \} \), where
\( h_k^* \subseteq h_k \) denotes the set of places with which no contract has been established and \( \text{TYPE}(\bullet) \) is a function that maps a place to the corresponding part type. The primary optimizer then issues a request for tender message which carries \( I_R \) to the set of potential bidders. The primary optimizer waits for the set of potential bidders \( U_n(i) \) to submit the set of proposals \( \Pi_n(S') \) under state \( S' \), where \( S' \) is the resulting state reached after the cancellation of the contracts between the Initiator and the primary optimizer. The optimizer needs to search for alternative composite holons to replace the set of composite holons based on \( \Pi_n(S') \). The following theorem states a condition for the existence of an optimal solution to the Holarchy Reconfiguration Problem.
Theorem 5.1: Let \( h_n \) be the primary optimizer. There exists an optimal solution to \( \Pi_n \) under state \( S' \) if and only if \( U_n(i) \neq \emptyset \) \( \forall p \in h^*_k \) and \( m_{ab} \geq m_{ab} \).
6. CONCLUSION
Effective reconfiguration mechanisms are essential to successfully taking advantage of the flexible architecture of HMS to effectively deal with dynamic removal/addition of holons due to resource failures while avoiding chaos at the shop floor. We formulate a holarchy reconfiguration problem (HRP). We define an impact function to characterize the impact of resource failures on a holarchy. We study the tolerable failure conditions for HMS to survive without reconfiguration. In case the resource failures do not satisfy the condition, the reconfiguration algorithm based on the impact function is initiated to reconfigure the systems to achieve minimal cost solutions while minimizing the impact on the existing tasks at the shop floor. We propose a reconfiguration mechanism for solving HRP based on the interactions and cooperation between four types of holons: detector, initiator, standby and optimizer. There are two types of initiators: Type I initiator and Type II initiator. A Type I initiator is a holon that initiates the actions to identify and remove holons from HMS whereas a Type 2 initiator is a holon that handles dynamic addition of holons to HMS. The reconfiguration process triggered by Type I initiator is broken down into two stages: (1) initiator identification & standby notification and (2) process optimization. As the reconfiguration process for Type I initiators is applicable to Type II initiators with the exception of standby notification step, we focus on the reconfiguration process for Type I initiators only. A condition for the existence of an optimal solution to HRP is established. Our study in this paper paves way for the development of an effective HMS reconfiguration mechanism with predictable performance.

REFERENCES