Modular Requirements for Hierarchical Interface-Based Supervisory Control with Multiple Levels

R. C. Hill, J.E.R. Cury, M.H. de Queiroz and D. M. Tilbury

***Abstract*— Hierarchical Interface-Based Supervisory Control employs interfaces that allow properties of a monolithic system to be verified through local analysis. By avoiding the need to verify properties globally, significant computational savings can be achieved. In this paper we provide local requirements for a multiple-level architecture. This multiple-level architecture allows for a greater reduction in complexity and improved reconfigurability over the two-level case that has been previously studied since it allows the global system to be partitioned into smaller modules. This paper also provides a relaxation of existing interface requirements resulting in less restrictive control and easier achievement of the multiple-level architecture.**

I. Introduction

In recent years, quite a well-formed body of theory has been developed with regard to the control of discrete-event systems (DES). Application of this theory has been hindered by the well-known *state-space explosion* problem. One approach to address this problem is to introduce interfaces between various components of a larger system [1] [2] [3]. The purpose of these interfaces is to limit the interaction of various components in such a way that global properties can be verified locally. By not requiring analysis of the global system, state-space explosion can often be avoided. This architecture also provides for improved reconfigurability since a system component can be modified without having to re-analyze the entire global system.

This type of approach does however have its drawbacks, namely that the increased restrictiveness of the interfaces can result in suboptimal control. In many cases, this exchange of optimality for a reduction in computational complexity and improved reconfigurability may be acceptable.

Considering existing research in this area, the results provided by [1] do not address blocking. The work of [2] [3] addresses the properties of controllability and nonblocking, but is somewhat restrictive in that its proposed architecture only allows for two levels of modules.

Other works that exist for reducing the complexity associated with synthesizing global nonblocking control rely on incremental construction and abstraction [4] [5] [6]. Still other research employs similar approaches, but for verification [7] [8]. These works are quite useful, but they do not strictly rely on local analysis and design; rather, their techniques are in essence applied to incrementally constructed abstractions of the global system. As such, these techniques are not very reconfigurable and can still suffer from state-space explosion.

This paper gives local conditions that guarantee global controllability and nonblocking of a multiple-level system with interfaces like the one pictured in Fig. 1. This generalized architecture allows the system to be partitioned into smaller modules than the two-level architecture of [2] [3], thereby further limiting the complexity of analysis and design. We also demonstrate that the interface consistency requirements of [2] [3] can be relaxed. This change makes the necessary requirements easier to satisfy, especially in the multiple-level case. Methodologies for designing the modules and interfaces themselves are not directly provided by this work.

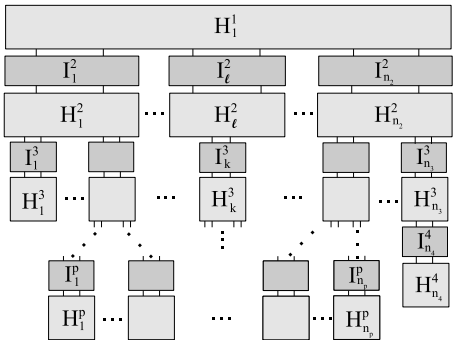

Fig. 1. Illustration of the multiple-level architecture

The organization of the remainder of this paper is as follows: Section II will introduce notation, Section III will demonstrate results for a multiple-level interface-based architecture, Section IV will demonstrate the application of this architecture to a manufacturing example, and Section V will conclude the paper with a summary of its contributions.

II. Notation and Preliminaries

We will consider DES modeled by automata, where each automaton is represented by the five-tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the set of states, Σ is the set of events, $\delta : Q \times \Sigma \to Q$ is the partial state transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked states representing successful termination of a process. Let Σ^* be the set of all finite strings of elements of Σ, including the empty string ε. The partial function δ can be extended to $\delta : Q \times \Sigma^* \to Q$ in the natural way. The notation $\delta(q, s)!$ for any $q \in Q$ and any $s \in \Sigma^*$ denotes that $\delta(q, s)$ is defined.

This work was supported in part by NSF grant CMS-05-28287.

R. Hill and D. Tilbury are with the University of Michigan, Ann Arbor, MI 48109-2125, USA (email:rchill,tilbury@umich.edu).

J. Cury and M. Queiroz are with the Federal University of Santa Catarina, Florianópolis, SC 88040-900, Brazil (email: cury,max@das.ufsc.br).

The notation $\Sigma(G)$ will be employed to denote the *relevant* event set of the automaton G. By relevant, it is meant all events over which G is defined that are not self-looped at every state.

The *generated* and *marked languages* of G, denoted by $\mathcal{L}(G)$ and $\mathcal{L}_m(G)$ respectively, are defined by $\mathcal{L}(G) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}$ and $\mathcal{L}_m(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$. The notation $\overline{L}$ represents the set of all prefixes of strings in the language L, and is referred to as the *prefix-closure* of L. The following eligibility operator will be employed to denote which events in the set Σ are enabled in the language L following the occurrence of a string $s \in \Sigma^*$, $\text{Elig}_L(s) := \{\sigma \in \Sigma \mid s\sigma \in L\}$. A precise definition for relevant events in terms of languages can be found in [9].

An automaton is said to be *nonblocking* when all of its reachable states can reach a marked state. From a language point of view, this is defined as $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$. If an automaton enters a state from which it cannot reach a marked state, the automaton is said to have *blocked*.

The operation of two automata G_1 and G_2 together is captured via the *synchronous composition* (parallel composition) operator, $\|$. When composed, events not shared by both automata are allowed to occur without participation of the other automaton, while those events that are shared must occur with the two automata synchronized.

We will use the following *projection* operator $P_i : \Sigma^* \to \Sigma_i^*$.

$$P_i(\varepsilon) := \varepsilon \qquad P_i(e) := \begin{cases} e, & e \in \Sigma_i \subseteq \Sigma \\ \varepsilon, & e \notin \Sigma_i \subseteq \Sigma \end{cases}$$
$$P_i(se) := P_i(s)P_i(e), s \in \Sigma^*, e \in \Sigma$$

Given a string $s \in \Sigma^*$, the projection P_i erases those events in the string that are in the global alphabet Σ, but not in the local alphabet Σ_i. We can also define the inverse projection for a string $t \in \Sigma_i^*$ as follows, $P_i^{-1}(t) := \{s \in \Sigma^* : P_i(s) = t\}$. These definitions can naturally be extended to languages.

A. Traditional Supervisory Control

In supervisory control [10], the event set of an automaton is partitioned into *controllable* and *uncontrollable* events, $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$, where controllable events can be disabled and uncontrollable events cannot. A supervisor, denoted $\mathcal{S}$, is a mapping that outputs a list of events to be disabled based on the observation of strings generated by a plant G. Keeping in mind that uncontrollable events are not allowed to be disabled, a supervisor $\mathcal{S} : \mathcal{L}(G) \to 2^{\Sigma_c}$ can be represented by an automaton S such that the closed loop system behavior $\mathcal{S}/G = S\|G$.

To ensure that a given automaton S with alphabet Σ represents a supervisor $\mathcal{S}$ that restricts the plant G to the behaviour of S, it is necessary that the following Σ_u-*controllability* condition be satisfied, where $\Sigma_u \subseteq \Sigma$. The following expression can be interpreted as providing that the language $\mathcal{L}(S)$ is Σ_u-controllable with respect to $\mathcal{L}(G)$, where the two languages have the same alphabet.

$$(\forall s \in \mathcal{L}(S) \cap \mathcal{L}(G))\ \text{Elig}_{\mathcal{L}(G)}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathcal{L}(S)}(s)$$

B. Hierarchical Interface-Based Supervisory Control

We will now define the notation and definitions necessary for proving results with regard to a multiple-level application of hierarchical interface-based supervisory control. We will specifically assume a connected tree architecture with a single root node. Figure 1 illustrates this situation. Our component-wise specified system is split up into modules, each consisting of a plant G_k^i and a supervisor S_k^i constructed with respect to a local specification resulting in the closed-loop subsystem $H_k^i = G_k^i \| S_k^i$. The superscript i reflects the level of the hierarchy and takes values $\{1, \ldots, p\}$. The subscript k indicates the index within a given level and takes the values $\{1, \ldots, n_i\}$, where this set represents all modules and interfaces on a given level i, including modules and interfaces that have different corresponding high-level neighbors.

All interaction between modules takes place through corresponding interfaces, I_k^i. These interfaces restrict the behaviour of the overall system in such a way that global properties can be guaranteed by local analysis. In a sense, these interfaces may apply additional control. Figure 2 shows a detail of the multiple-level architecture. We do not provide a general methodology for the construction of supervisors or interfaces, though we do provide conditions for guaranteeing global properties and provide an example multiple-level system that satisfies these requirements in Section IV.

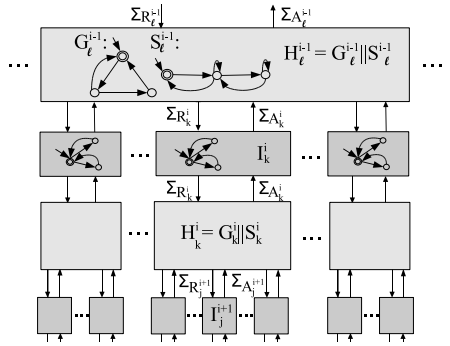

Fig. 2. Detail of the multiple-level architecture

In this architecture, all events shared between a given module H_k^i and its high-level neighbor are classified as either request events $\rho \in \Sigma_{R_k^i}$ or answer events $\alpha \in \Sigma_{A_k^i}$. The occurrence of each of these events must then be accepted by the corresponding interface I_k^i. Conceptually, request events are thought of as being under the control of the higher-level module and answer events as being under the control of the lower-level module. For the purposes of this paper we will assume the interfaces take the form of a *command-pair interface* defined below in the manner of [2].

Definition 1: [2] A DES I_k^i is a *command-pair interface* if the following are true:

A) $\mathcal{L}(I_k^i) \subseteq \overline{(\Sigma_{R_k^i} . \Sigma_{A_k^i})^*}$

B) $\mathcal{L}_m(I_k^i) = (\Sigma_{R_k^i} . \Sigma_{A_k^i})^* \cap \mathcal{L}(I_k^i)$

We will assume that the global alphabet is partitioned as shown in (1), where the set $\Sigma_{I_k^i}$ represents those events

relevant to H_k^i but no other modules. The following also assumes there is only a single module on level 1, the top level.

$$\Sigma := \Sigma_1^1 \dot\cup \dot\bigcup_{i=2,\ldots,p} \left(\dot\bigcup_{k=1,\ldots,n_i} \left(\Sigma_k^i \dot\cup \Sigma_{A_k^i} \dot\cup \Sigma_{R_k^i} \right) \right) \quad (1)$$

A consequence of (1) is that each interface is completely disjoint from all other interfaces, that is, $\Sigma(I_k^i) \cap \Sigma(I_{k'}^{i'}) = \emptyset, \forall((i \neq i') \vee (k \neq k'))$. We will further assume that the event set of each module H_k^i is constrained to have the partitioning given in (2). The following is consistent with the connected tree architecture of this approach. Since there are no interfaces with superscript 1 or $p+1$, we will consider $\Sigma(I_k^1) = \Sigma(I_j^{p+1}) = \emptyset$.

$$\begin{aligned} \Sigma(H_k^i) &= \Sigma_k^i \dot\cup \Sigma(I_k^i) \dot\cup \dot\bigcup_{j \in J_k^i} \Sigma(I_j^{i+1}) \\ \text{where} \quad & J_k^i := \{ j \mid \Sigma(H_k^i) \cap \Sigma(I_j^{i+1}) \neq \emptyset \} \end{aligned} \quad (2)$$

In the above, the index sets J_k^i for modules on level i partition the set $\{1, \ldots, n_{i+1}\}$ into disjoint subsets. An implication of (2) is that each module H_k^i may share relevant events only with modules from the $i+1$ level and a single module from the $i-1$ level. We will employ script letters to represent the languages generated by the corresponding automata lifted to the global alphabet. This convention is employed in the following definitions.

$$\begin{aligned} P_{H_k^i} &: \Sigma^* \to \Sigma(H_k^i)^* & P_{I_k^i} &: \Sigma^* \to \Sigma(I_k^i)^* \\ \mathcal{H}_k^i &:= P_{H_k^i}^{-1}(\mathcal{L}(H_k^i)) & \mathcal{H}_{m_k}^i &:= P_{H_k^i}^{-1}(\mathcal{L}_m(H_k^i)) \\ \mathcal{G}_k^i &:= P_{H_k^i}^{-1}(\mathcal{L}(G_k^i)) & \mathcal{S}_k^i &:= P_{H_k^i}^{-1}(\mathcal{L}(S_k^i)) \\ \mathcal{I}_k^i &:= P_{I_k^i}^{-1}(\mathcal{L}(I_k^i)) & \mathcal{I}_{m_k}^i &:= P_{I_k^i}^{-1}(\mathcal{L}_m(I_k^i)) \end{aligned}$$

The following requirements modified from [11] will be employed to guarantee global properties through local analysis for a given set of DES. Each definition is given with respect to a two-level portion of the larger multiple-level system. In order to make these definitions easier to follow, the following substitutions will be made: each two-level subsystem will have a single high-level module $H = H_\ell^{i-1}$, a set of low-level modules $L_k = H_k^i$ and corresponding interfaces $I_k = I_k^i$, where k represents those indices in the set $K = \{k \mid \Sigma(H_\ell^{i-1}) \cap \Sigma(I_k^i) \neq \emptyset\}$. Also let $\mathcal{I} = \bigcap_{k \in K} \mathcal{I}_k$ and $\mathcal{I}_m = \bigcap_{k \in K} \mathcal{I}_{m_k}$. We will employ the notation $\{L_k\}$ and $\{I_k\}$ to represent respectively the set of low-level modules and interfaces with $k \in K$.

The event set of each interface I_k is still partitioned into request and answer events where $\Sigma_{I_k} = \Sigma(I_k^i)$, $\Sigma_{A_k} = \Sigma_{A_k^i}$ and $\Sigma_{R_k} = \Sigma_{R_k^i}$. The event set of each module H and L_k will be denoted $\Sigma_{IH} = \Sigma(H_\ell^{i-1})$ and $\Sigma_{IL_k} = \Sigma(H_k^i)$, respectively, where each set includes those events shared with its corresponding interfaces. The event set Σ_{L_k} is defined as those events relevant to L_k that are not shared with its high-level interface, that is, $\Sigma_{L_k} = \Sigma(H_k^i) - \Sigma(I_k^i)$. Furthermore, the high-level supervisor and plant will be denoted S_H and G_H respectively. Likewise for each low-level module, S_{L_k} is the supervisor and G_{L_k} is the plant. Script letters will again represent the languages generated by the corresponding automata lifted to the global alphabet. The alphabet partitions of (1) and (2) will still hold.

Definition 2: [11] A two-level interface system composed of H, $\{L_k\}$, and $\{I_k\}$ is said to be *level-wise nonblocking* if the following conditions are satisfied.

i) $\overline{\mathcal{H}_m \cap \mathcal{I}_m} = \mathcal{H} \cap \mathcal{I}$

ii) $\overline{\mathcal{L}_{m_k} \cap \mathcal{I}_{m_k}} = \mathcal{L}_k \cap \mathcal{I}_k, \ \forall k$

Definition 3: [11] A two-level interface system composed of plant components G_H, $\{G_{L_k}\}$, supervisors S_H, $\{S_{L_k}\}$, and interfaces $\{I_k\}$ is said to be *level-wise controllable* if for all k, the following conditions are satisfied.

i) The alphabet of G_H and S_H is Σ_{IH}, of G_{L_k} and S_{L_k} is Σ_{IL_k}, and of I_k is Σ_{I_k}.

ii) $(\forall s \in \mathcal{G}_{L_k} \cap \mathcal{S}_{L_k} \cap \mathcal{I}_k)$
$\text{Elig}_{\mathcal{G}_{L_k}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathcal{S}_{L_k} \cap \mathcal{I}_k}(s)$

iii) $(\forall s \in \mathcal{G}_H \cap \mathcal{S}_H \cap \mathcal{I})$
$\text{Elig}_{\mathcal{G}_H \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathcal{S}_H}(s)$

Definition 4: A two-level interface system composed of H, $\{L_k\}$, and $\{I_k\}$ is said to be *interface consistent* if for all k, the following properties are satisfied.

1) The event set of H is Σ_{IH} and the event set of each L_k is Σ_{IL_k}.
2) Each I_k is a command-pair interface.
3) $(\forall s \in \mathcal{H} \cap \mathcal{I})$ $\text{Elig}_{\mathcal{I}_k}(s) \cap \Sigma_{A_k} \subseteq \text{Elig}_{\mathcal{H}}(s)$
4) $(\forall s \in (\Sigma^*.\Sigma_{A_k})^*.\Sigma_{L_k}^* \cap \mathcal{L}_k \cap \mathcal{I}_k)$,
$\text{Elig}_{\mathcal{L}_k \cap \mathcal{I}_k}(s\Sigma_{L_k}^*) \cap \Sigma_{R_k} = \text{Elig}_{\mathcal{I}_k}(s) \cap \Sigma_{R_k}$
5) $(\forall s \in \Sigma^*.\Sigma_{R_k} \cap \mathcal{L}_k \cap \mathcal{I}_k)$,
$\text{Elig}_{\mathcal{L}_k \cap \mathcal{I}_k}(s\Sigma_{L_k}^*) \cap \Sigma_{A_k} = \text{Elig}_{\mathcal{I}_k}(s) \cap \Sigma_{A_k}$
6) $(\forall s \in \mathcal{L}_k \cap \mathcal{I}_k)$
$s \in \mathcal{I}_{m_k} \Rightarrow (\exists l \in \Sigma_{L_k}^*)\ sl \in \mathcal{L}_{m_k} \cap \mathcal{I}_{m_k}$

In Definition 4, $\text{Elig}_{\mathcal{L}_k \cap \mathcal{I}_k}(s\Sigma_{L_k}^*)$ is defined to be equal to $\cup_{l \in \Sigma_{L_k}^*} \text{Elig}_{\mathcal{L}_k \cap \mathcal{I}_k}(sl)$. Furthermore, in words, Point 3 requires that the high-level module be Σ_{A_k}-controllable with respect to each of its low-level interfaces I_k. Point 4 and 5 require that request and answer events, respectively, be reachable in the low-level by events not shared with the corresponding high-level module. Point 6 requires that if a string is marked and accepted by an interface, then it can be extended to a marked string in the corresponding low-level language by events that again are not shared with the high-level module. Some of these requirements are discussed further in Section III-A.

We will now define properties analogous to the above level-wise conditions for multiple-level interface systems.

Definition 5: Consider a multiple-level interface system composed of modules H_k^i and interfaces I_k^i, where each module H_k^i consists of a plant G_k^i and supervisor S_k^i. Furthermore, consider a series of two-level portions of the multiple-level system each composed of a high-level H_ℓ^{i-1}, interfaces $\{I_k^i \mid k \in J_\ell^{i-1}\}$, and a low-level of modules corresponding to these interfaces $\{H_k^i || (||_{j \in J_k^i} I_j^{i+1})\}$ where $i = \{2, \ldots, p\}$ and $\ell = \{1, \ldots, n_{i-1}\}$ for each i. Note, the low-level "modules" include the interfaces associated with those modules directly below them in the hierarchy, except for those modules that are not preceded by any interfaces. For those low-level modules that are preceded by interfaces, their

associated plants are then $\{G_k^i || (||_{j \in J_k^i} I_j^{i+1})\}$. This multiple-level interface system is said to be:
A) *multi-level nonblocking* if each two-level interface system defined above is level-wise nonblocking by Definition 2.
B) *multi-level controllable* if each two-level interface system defined above is level-wise controllable by Definition 3.
C) *multi-level consistent* if each two-level interface system defined above is interface consistent by Definition 4.

III. Global Nonblocking and Controllability

In this section we will present the main results of this paper. Specifically, we will show that if a set of local conditions based on the definitions of Section II-B are satisfied, then the global multiple-level system with its interfaces is nonblocking and controllable. These results are presented in Theorem 1 and Theorem 2 given below. Their proofs are presented after some important special cases are discussed.

Theorem 1: Let there be a multiple-level interface system composed of modules H_1^1 and $\{H_k^i\}$, and interfaces $\{I_k^i\}$, where $i = \{2, \ldots, p\}$ and $k = \{1, \ldots, n_i\}$ for each i. If this system is *multi-level nonblocking* and *multi-level consistent* with respect to the alphabet partition of (1) and (2), then the complete system is nonblocking:

$$\overline{\mathcal{H}_m^1 \cap \mathcal{H}_m^2 \cap \mathcal{I}_m^2 \cap \ldots \cap \mathcal{H}_m^p \cap \mathcal{I}_m^p} = \mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^2 \cap \ldots \cap \mathcal{H}^p \cap \mathcal{I}^p$$

where

$$\mathcal{H}_m^i = \mathcal{H}_{m_1}^i \cap \ldots \cap \mathcal{H}_{m_{n_i}}^i, \quad \mathcal{I}_m^i = \mathcal{I}_{m_1}^i \cap \ldots \cap \mathcal{I}_{m_{n_i}}^i$$
$$\mathcal{H}^i = \mathcal{H}_1^i \cap \ldots \cap \mathcal{H}_{n_i}^i, \qquad \mathcal{I}^i = \mathcal{I}_1^i \cap \ldots \cap \mathcal{I}_{n_i}^i$$

Theorem 2: Let there be a multiple-level interface system composed of component plants G_1^1 and $\{G_k^i\}$, component supervisors S_1^1 and $\{S_k^i\}$, and interfaces $\{I_k^i\}$, where $i = \{2, \ldots, p\}$ and $k = \{1, \ldots, n_i\}$ for each i. If this system is *multi-level controllable* with respect to the alphabet partition of (1) and (2), then the supervisor language $\mathcal{S} = \mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{I}^2 \cap \ldots \cap \mathcal{S}^p \cap \mathcal{I}^p$ is Σ_u-controllable with respect to the plant language $\mathcal{G} = \mathcal{G}^1 \cap \ldots \cap \mathcal{G}^p$.

Where: $\mathcal{S}^i = \mathcal{S}_1^i \cap \ldots \cap \mathcal{S}_{n_i}^i$, $\mathcal{I}^i = \mathcal{I}_1^i \cap \ldots \cap \mathcal{I}_{n_i}^i$ and $\mathcal{G}^i = \mathcal{G}_1^i \cap \ldots \cap \mathcal{G}_{n_i}^i$

A. Two-Level Case

In this subsection we present the following results which are special cases of Theorem 1 and Theorem 2 for a two-level system. Specifically, Theorem 3 is a result modified from [11] for the new interface consistency definition of Section II-B, while Theorem 4 is taken directly from [11].

Theorem 3: Let there be a two-level interface system composed of modules H and $\{L_k\}$, and interfaces $\{I_k\}$. If this system is *level-wise nonblocking* and *interface consistent*, then

$$\overline{\mathcal{H}_m \cap \bigcap_{k \in K} (\mathcal{L}_{m_k} \cap \mathcal{I}_{m_k})} = \mathcal{H} \cap \bigcap_{k \in K} (\mathcal{L}_k \cap \mathcal{I}_k)$$

Theorem 4: [11] Let there be a two-level interface system composed of plant components G_H and $\{G_{L_k}\}$, supervisors S_H and $\{S_{L_k}\}$, and interfaces $\{I_k\}$. If this system is *level-wise controllable*, then the supervisor language $\mathcal{S} = \mathcal{S}_H \cap \bigcap_{k \in K} (\mathcal{S}_{L_k} \cap \mathcal{I}_k)$ is Σ_u-controllable with respect to the plant language $\mathcal{G} = \mathcal{G}_H \cap \bigcap_{k \in K} (\mathcal{G}_{L_k})$.

Proof of Theorem 3 can be found in [12] and follows closely the logic presented in [13], where the only difference is that the interface consistency requirement has been relaxed. Specifically, Point 4 of Definition 4 has been modified from what was originally a controllability requirement to the reachability requirement prescribed in this paper. The original Point 4 required that each low-level module $\mathcal{L}_k$ be Σ_{R_k}-controllable with respect to its interface $\mathcal{I}_k$:

$$(\forall s \in \mathcal{L}_k \cap \mathcal{I}_k),\ \text{Elig}_{\mathcal{I}_k}(s) \cap \Sigma_{R_k} \subseteq \text{Elig}_{\mathcal{L}_k}(s) \tag{3}$$

The spirit of this requirement is that the low-level has control only over those events shared with the high-level that are answer events. Therefore, the high-level knows that if it issues a request that is accepted by the interface, the low-level will not disable it. This requirement is mirrored by Point 3 that specifies that the high-level module $\mathcal{H}$ be Σ_{A_k}-controllable with respect to each of its interfaces $\mathcal{I}_k$. These requirements are at the core of what enables us to draw conclusions about the global system with only local analysis.

The new Point 4 still captures the intent of the original requirement by requiring instead that the low-level be able to reach, via a string of low-level events, each request event allowed by the interface. Therefore, even though the low-level may not allow a request event immediately (as dictated by the original controllability requirement), it will eventually be able to execute the required request event following the occurrence of a string of low-level events. Since the request event is reached by local low-level events, we know that the low-level cannot be prevented from reaching the request event by interaction with the interface or high-level module. The following example helps to illustrate the difference between the original and modified requirements.

Example 1: Consider the interface I and low-level module L displayed in Fig. 3. Let $\mathcal{I}$ and $\mathcal{L}$ be the respective generated languages lifted to the global alphabet. For the set of request events $\Sigma_R = \{r_1, r_2\}$ and answer events $\Sigma_A = \{a_1, a_2\}$, I is a command-pair interface. It can be seen by inspection that $\mathcal{L}$ is not Σ_R-controllable with respect to $\mathcal{I}$. Specifically, $r_1 \notin \text{Elig}_{\mathcal{L}}(\varepsilon)$, but $r_1 \in \text{Elig}_{\mathcal{I}}(\varepsilon)$, implying a violation of Σ_R-controllability since r_1 is not enabled at state 0 of L. Also, $r_2 \notin \text{Elig}_{\mathcal{L}}(l_1 r_1 a_1 l_2)$, but $r_2 \in \text{Elig}_{\mathcal{I}}(l_1 r_1 a_1 l_2)$, therefore, implying a violation since r_2 is not enabled at state 4 of L. L can be modified to remove state 4 (and subsequently state 5) and will generate a non-trivial language such that $l_1 r_1 a_1 l_2 \notin \mathcal{L}$, but the only way to remove the string ε from $\mathcal{L}$ is to make L the empty automaton. Another possible remedy is to replace the event r_1 in the set Σ_R by the event l_1. The problem that we run into here is that it could be the case in a multiple-level architecture that l_1 was employed in an interface from a lower level of the architecture.

The original language $\mathcal{L}$ however does satisfy the modified Point 4 with respect to the interface language $\mathcal{I}$. For example,

even though the request r_1 is not enabled at state 0 of L, r_1 can be reached by low-level events. Likewise, the request event r_2 can be reached from state 4 via low-level events. $\diamond$

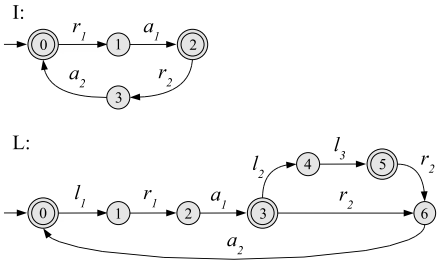

Fig. 3. Example illustrating the relaxation of Point 4

Point 4 of the interface consistency definition is specifically employed in Proposition 13 of [13]. A revised version of this proposition using the modified Point 4 can be found in [12]. This revised proposition demonstrates that the relaxed interface consistency definition holds in the two-level case. Later it will be seen that our interface consistency definition is sufficient for the multiple-level case also.

It should be noted that if this interface-based approach to control is implemented in a distributed fashion, then the new Point 4 will make it difficult for the modules to truly synchronize on a request event, since an event requested by a module does not have to occur in the lower level immediately. If the modular control is implemented on a centralized computer, then the modules can synchronize with one another since there are no problems with communication. If the modular supervisors are distributed across several computers, then when a request is made by a module, the associated lower-level module will have to queue this request until it is able to address it. Even though the modules will not actually be synchronized in time, all the necessary actions will still be performed in the correct order.

B. Multiple-Level Serial Case

We will now demonstrate results analogous to Theorem 1 and Theorem 2 for the case where each level of the hierarchy consists of only a single module. This case is referred to as a multiple-level serial-interface architecture. Examination of the proofs for this case will make the logic of the main results of this paper easier to follow.

Controllability and nonblocking of the multiple-level serial-interface architecture will follow from the results presented for the two-level case. Specifically, the requirements of Theorem 3 and Theorem 4 must be met for a series of two-level systems consisting of a high level of $H^{i-1} = S^{i-1} \| G^{i-1}$, an interface of I^i, and a low level of $H^i \| I^{i+1} = S^i \| G^i \| I^{i+1}$, where $i = \{2, \ldots, p\}$. The proofs to follow rely on this modified formulation where the low-level plant includes the interface from the level below, that is, the low-level plant is considered $G^i \| I^{i+1}$. Furthermore, the disjointness of the alphabet partition of (1) and (2) will also be needed. For the interface I^p, H^{p-1} is considered the high-level and H^p is considered the low-level since there is no interface preceding the first level of the hierarchy.

Figure 4 illustrates the approach taken in the following proofs. The proofs begin with the two-level system at the top of the hierarchy, which is immediately nonblocking and controllable by Theorems 3 and 4. We then consider this serial system to be the "high-level" and add another module that is considered the "low-level." This process continues where the high-level gets larger and larger and the low-level is just the next module considered. With this in mind, all low-level requirements are immediately met. The high-level properties are shown by induction.

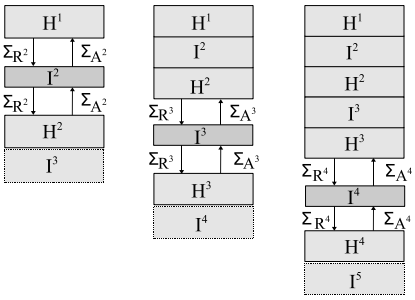

Fig. 4. Illustration of approach of proofs

The two propositions given below will be needed in the proofs to follow.

Proposition 1: Let $\mathcal{K},\ \mathcal{L} \subseteq \Sigma^*$ be prefix-closed languages. If $\mathcal{K}$ does not have any relevant events in the set $\Sigma_u \subseteq \Sigma$, then $\mathcal{K}$ is Σ_u-controllable with respect to $\mathcal{L}$.

Proof: See the proof in [12]. ■

Proposition 2: [10] Let $\mathcal{K}_1,\ \mathcal{K}_2,\ \mathcal{L} \subseteq \Sigma^*$ be prefix-closed languages. If $\mathcal{K}_1$ and $\mathcal{K}_2$ are each Σ_u-controllable with respect to $\mathcal{L}$, then the intersection $\mathcal{K}_1 \cap \mathcal{K}_2$ is also Σ_u-controllable with respect to $\mathcal{L}$.

The following two important theorems demonstrate local conditions under which the global multiple-level serial interface system is nonblocking and controllable.

Theorem 5: Let there be a multiple-level interface system composed of modules $\{H^1, \ldots, H^p\}$ and interfaces $\{I^2, \ldots, I^p\}$. If this system is *multi-level nonblocking* and *multi-level consistent* with respect to the alphabet partition of (1) and (2), then the global system is nonblocking:

$$\overline{\mathcal{H}^1_m \cap \mathcal{H}^2_m \cap \mathcal{I}^2_m \cap \ldots \cap \mathcal{H}^p_m \cap \mathcal{I}^p_m} = \mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^2 \cap \ldots \cap \mathcal{H}^p \cap \mathcal{I}^p$$

Proof:

• Beginning at the top of the hierarchy, consider a two-level system consisting of a high-level H^1, a low-level of $H^2 \| I^3$, and an interface I^2. Because the overall system is multi-level nonblocking and multi-level consistent, this two-level component is level-wise nonblocking and interface consistent. Therefore, Theorem 3 can be applied to show that:

$$\overline{\mathcal{H}^1_m \cap \mathcal{H}^2_m \cap \mathcal{I}^2_m \cap \mathcal{I}^3_m} = \mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^2 \cap \mathcal{I}^3 \quad (4)$$

• Now consider a two-level system where the high-level is $H^1 \| H^2 \| I^2$, the low-level is $H^3 \| I^4$, and the interface is I^3. Based on the given assumptions, all low-level and multi-level requirements are known to be met. The level-wise nonblocking of the high-level has been shown to be met by (4). The only necessary requirement left to be shown is that Point 3 of the interface consistency definition is satisfied, that is, $\mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^2$ is Σ_{A^3}-controllable with respect to the interface $\mathcal{I}^3$.

By the given interface consistency requirements, $\mathcal{H}^2$ is Σ_{A^3}-controllable with respect to the interface $\mathcal{I}^3$. By (1) and (2), the languages $\mathcal{H}^1$ and $\mathcal{I}^2$ do not have any relevant events in the set Σ_{A^3}, therefore, they are both Σ_{A^3}-controllable with respect to $\mathcal{I}^3$ by Proposition 1. Hence, the intersection $\mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^2$ is also Σ_{A^3}-controllable with respect to $\mathcal{I}^3$ by Proposition 2 since all languages are prefix-closed.

Since all necessary requirements are met for this two-level system, Theorem 3 can be employed again to show that:

$$\overline{\mathcal{H}^1_m \cap \mathcal{H}^2_m \cap \mathcal{I}^2_m \cap \mathcal{H}^3_m \cap \mathcal{I}^3_m \cap \mathcal{I}^4_m} = \mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^2 \cap \mathcal{H}^3 \cap \mathcal{I}^3 \cap \mathcal{I}^4$$

• This logic is repeated until all modules have been addressed, leading to the desired result. ■

In the above proof, the "low-level" module always stands alone, thus Point 4 of Definition 4 is immediately satisfied (as well as all other low-level requirements).

Theorem 6: Let there be a multiple-level interface system composed of component plants $\{G^1, \ldots, G^p\}$, component supervisors $\{S^1, \ldots, S^p\}$, and interfaces $\{I^2, \ldots, I^p\}$. If this system is *multi-level controllable* with respect to the alphabet partition of (1) and (2), then the supervisor language $\mathcal{S} = \mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{I}^2 \cap \ldots \cap \mathcal{S}^p \cap \mathcal{I}^p$ is Σ_u-controllable with respect to the plant language $\mathcal{G} = \mathcal{G}^1 \cap \ldots \cap \mathcal{G}^p$.

Proof:

• Beginning at the top of the hierarchy, consider a two-level system consisting of a high-level plant G^1 and supervisor S^1, a low-level plant $G^2 \| I^3$ and supervisor S^2, and an interface I^2. Since the overall system is multi-level controllable, this two-level component is level-wise controllable. Therefore, Theorem 4 can be applied to show that $\mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{I}^2$ is Σ_u-controllable with respect to $\mathcal{G}^1 \cap \mathcal{G}^2 \cap \mathcal{I}^3$.

• Now consider a two-level system with the high-level plant $G^1 \| G^2$ and supervisor $S^1 \| S^2 \| I^2$, a low-level plant $G^3 \| I^4$ and supervisor S^3, and an interface I^3. Based on the given assumptions, points $i)$ and $ii)$ of the level-wise controllability requirement are satisfied. Additionally, point $iii)$ is satisfied by the previous step of this proof. Therefore, Theorem 4 can be applied again to show that $\mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{I}^2 \cap \mathcal{S}^3 \cap \mathcal{I}^3$ is Σ_u-controllable with respect to $\mathcal{G}^1 \cap \mathcal{G}^2 \cap \mathcal{G}^3 \cap \mathcal{I}^4$.

• This logic is repeated until all modules have been addressed, leading to the desired result. ■

C. General Multiple-Level Case

The proofs of the main results of this paper follow the logic of Theorems 5 and 6, but with multiple modules per level of the hierarchy. Recall Fig. 1 and Fig. 2 that illustrate the general multiple-level architecture we are considering.

Proof of Theorem 1:

• Beginning at the top of the hierarchy, consider a two-level system consisting of a high-level H^1_1, a set of interfaces $\{I^2_\ell\}$, and a corresponding set of low-level modules $\{H^2_\ell || (||_{k \in J^2_\ell} I^3_k)\}$ where $\ell = \{1, \ldots, n_2\}$. It is given that the overall system is multi-level nonblocking and multi-level consistent, therefore this two-level component is level-wise nonblocking and interface consistent and Theorem 3 can be applied to show (5). Within a given level, all modules are included since the system is connected.

$$\overline{\mathcal{H}^1_m \cap \mathcal{H}^2_m \cap \mathcal{I}^2_m \cap \mathcal{I}^3_m} = \mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^3 \cap \mathcal{I}^3 \quad (5)$$

• Now consider a system with a high-level $H^1_1 \| H^2_1 \| \ldots \| H^2_{n_2} \| I^2_1 \| \ldots \| I^2_{n_2}$, a set of interfaces $\{I^3_k\}$, and a corresponding set of low-level modules $\{H^3_k || (||_{j \in J^3_k} I^4_j)\}$ where $k = \{1, \ldots, n_3\}$. Based on the given assumptions, all low-level and multi-level requirements are known to be met. The level-wise nonblocking of the high-level has been shown to be met by (5). The only requirement left is Point 3 of the interface consistency definition, that is, it must be shown that the high-level language is $\Sigma_{A^3_k}$-controllable with respect to each $\mathcal{I}^3_k$, $\forall k = \{1, \ldots, n_3\}$.

Consider a single interface language from this two-level system, $\mathcal{I}^3_k$. On level 2, there is a single module H^2_ℓ that shares relevant events with this interface, that is, $(\Sigma(H^2_\ell) \cap \Sigma(I^3_k) \neq \emptyset)$. By construction, the language generated by this module $\mathcal{H}^2_\ell$ is $\Sigma_{A^3_k}$-controllable with respect to $\mathcal{I}^3_k$ due to the interface consistency requirements. For those modules $H^2_{\ell'}$ from level 2 that do not share relevant events with I^3_k, they do not possess any relevant events that are in the set $\Sigma_{A^3_k}$ by (2). Therefore by Proposition 1, each language $\mathcal{H}^2_{\ell'}$ for which $\ell' \neq \ell$ is also $\Sigma_{A^3_k}$-controllable with respect to $\mathcal{I}^3_k$.

Furthermore, each interface from level 2, I^2_ℓ, and the module from the level 1, H^1_1, also do not share any relevant events with the event set $\Sigma_{A^3_k}$ by (1) and (2). Applying Proposition 1 again demonstrates that each of the languages generated by these DES are $\Sigma_{A^3_k}$-controllable with respect to the interface language $\mathcal{I}^3_k$.

Since the module language $\mathcal{H}^1_1$, and the interface and module languages, $\mathcal{I}^2_\ell$ and $\mathcal{H}^2_\ell$ where $\ell = \{1, \ldots, n_2\}$, are $\Sigma_{A^3_k}$-controllable with respect to $\mathcal{I}^3_k$, so is the composition of these languages by Proposition 2. Otherwise stated, $\mathcal{H} = \mathcal{H}^1_1 \cap \mathcal{H}^2_1 \cap \ldots \cap \mathcal{H}^2_{n_2} \cap \mathcal{I}^2_1 \cap \ldots \cap \mathcal{I}^2_{n_2}$ is $\Sigma_{A^3_k}$-controllable with respect to the interface language $\mathcal{I}^3_k$. Repeating this logic, this high-level language can be shown to be $\Sigma_{A^3_k}$-controllable with respect to any interface language $\mathcal{I}^3_k$, $\forall k = \{1, \ldots, n_3\}$. Therefore we have shown that Point 3 has been satisfied. Since all requirements are met for this two-level system, Theorem 3 then gives us:

$$\overline{\mathcal{H}^1_m \cap \mathcal{H}^2_m \cap \mathcal{I}^2_m \cap \mathcal{H}^3_m \cap \mathcal{I}^3_m \cap \mathcal{I}^4_m} = \mathcal{H}^1 \cap \mathcal{H}^2 \cap \mathcal{I}^2 \cap \mathcal{H}^3 \cap \mathcal{I}^3 \cap \mathcal{I}^4$$

• This logic is repeated until all modules on all p levels have been addressed. Low-level modules that do not have any interfaces below them are slightly different in that each module

just has the form H_k^i. However, they still satisfy the level-wise nonblocking and interface consistency requirements leading to the desired result. ■

Proof of Theorem 2:

• Beginning at the top of the hierarchy, consider a two-level interface system consisting of a high-level plant G_1^1 and supervisor S_1^1, a set of interfaces $\{I_\ell^2\}$, and a corresponding set of low-level plants $\{G_\ell^2||(||_{k\in J_\ell^2} I_k^3)\}$ and supervisors $\{S_\ell^2\}$ where $\ell = \{1,\ldots,n_2\}$. Because the overall system is multi-level controllable, this two-level component is level-wise controllable. Therefore, Theorem 4 can be applied to show that the language $\mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{I}^2$ is Σ_u-controllable with respect to $\mathcal{G}^1 \cap \mathcal{G}^2 \cap \mathcal{I}^3$. Within a given level, all modules are included since the system is connected.

• Now consider an interface system with a high-level plant $G_1^1 \| G_1^2 \| \ldots \| G_{n_2}^2$ and supervisor $S_1^1 \| S_1^1 \| \ldots \| S_{n_2}^2 \| I_1^2 \| \ldots \| I_{n_2}^2$, a set of interfaces $\{I_k^3\}$, and a corresponding set of low-level plants $\{G_k^3||(||_{j\in J_k^3} I_j^4)\}$ and supervisors $\{S_k^3\}$ where $k = \{1,\ldots,n_3\}$. Based on the given assumptions, points $i)$ and $ii)$ of the level-wise controllability requirement are satisfied. Additionally, point $iii)$ is known to be satisfied based on the previous step of this proof. Therefore, Theorem 4 can be applied again to show that the language $\mathcal{S}^1 \cap \mathcal{S}^2 \cap \mathcal{I}^2 \cap \mathcal{S}^3 \cap \mathcal{I}^3$ is Σ_u-controllable with respect to $\mathcal{G}^1 \cap \mathcal{G}^2 \cap \mathcal{G}^3 \cap \mathcal{I}^4$.

• This logic is repeated until all modules on all p levels have been addressed. Low-level modules that do not have any interfaces below them are slightly different in that their plant components just have the form G_k^i. They each still satisfy level-wise controllability leading to the desired result. ■

IV. Manufacturing System Example

In this section we will demonstrate an application of this new theory to the manufacturing example shown in Fig. 5 (modified from [14]). In this example the machines are considered the component plants and the buffers are considered the component specifications. The automaton models of these plants and specifications can be found in Fig. 6 and Fig. 7 respectively. In these models, states with double circles are considered marked states and the state with the short arrow is considered the initial state. Furthermore, the convention is employed where odd numbers represent controllable events and even numbers correspond to uncontrollable events.

Application of this approach depends on designer understanding. Specifically, how the system components are partitioned into modules, how request and answer events are chosen, and how interfaces and supervisors are constructed, are all areas where designer intuition can enter in. Multiple combinations may be tried to find a satisfactory solution.

Here we present a solution to the multiple-level problem that satisfies the requirements prescribed in this paper. The dashed boxes in Fig. 5 demonstrate the partition chosen for this example. Figure 8 illustrates the hierarchy imposed upon the system and the flow of information. The sets of request and answer events are chosen to be: $\Sigma_{R_1^2} = \{61\}$, $\Sigma_{A_1^2} = \{64\}$, $\Sigma_{R_2^2} = \{91\}$, $\Sigma_{A_2^2} = \{94\}$, $\Sigma_{R_1^3} = \{33\}$, $\Sigma_{A_1^3} = \{30, 38\}$, $\Sigma_{R_2^3} = \{71\}$, and $\Sigma_{A_2^3} = \{74\}$. Figure 9 shows the designed interface automata.

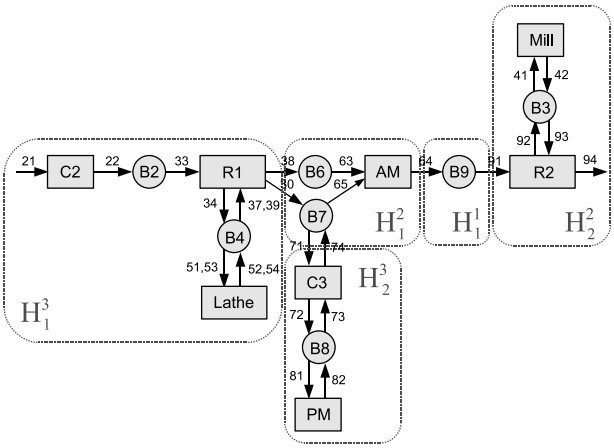

Fig. 5. Flexible manufacturing system example

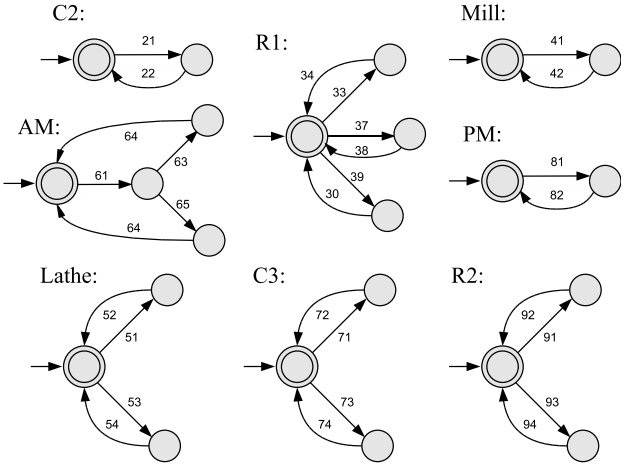

Fig. 6. Automaton models of plant components

The modular supervisors constructed according to the following equations succeed in satisfying all the necessary level-wise and interface-based requirements. Here the notation $\sup\mathcal{C}(\mathcal{K},\mathcal{L})$ represents the supremal sublanguage of $\mathcal{K}$ that is controllable with respect to $\mathcal{L}$. Notice that each specification and plant component are addressed by a modular supervisor. Also, each modular supervisor is built based on components in its given partition and on neighboring interfaces, that is, each modular supervisor is constructed employing only local information.

$$\mathcal{S}_1^1 = \sup\mathcal{C}(\mathcal{L}_m(B9\|I_1^2\|I_2^2), \mathcal{L}(I_1^2\|I_2^2))$$
$$\mathcal{S}_1^2 = \sup\mathcal{C}(\mathcal{L}_m(B6\|B7\|AM\|I_1^3\|I_2^3), \mathcal{L}(AM\|I_1^3\|I_2^3))$$
$$\mathcal{S}_2^2 = \sup\mathcal{C}(\mathcal{L}_m(B3\|R2\|Mill), \mathcal{L}(R2\|Mill))$$
$$\mathcal{S}_1^3 = \sup\mathcal{C}(\mathcal{L}_m(B2\|B4\|C2\|R1\|Lathe), \mathcal{L}(C2\|R1\|Lathe))$$
$$\mathcal{S}_2^3 = \sup\mathcal{C}(\mathcal{L}_m(B8\|C3\|PM), \mathcal{L}(C3\|PM))$$

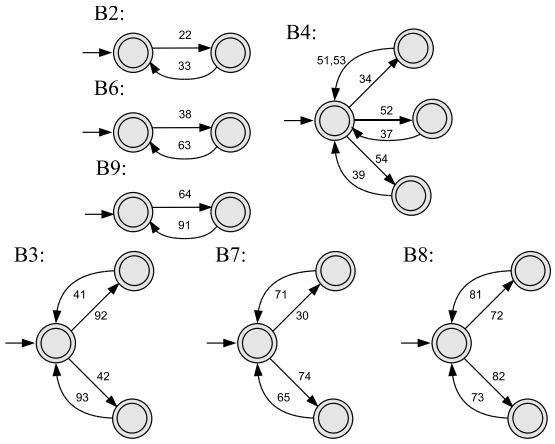

Fig. 7. Automaton models of specification components

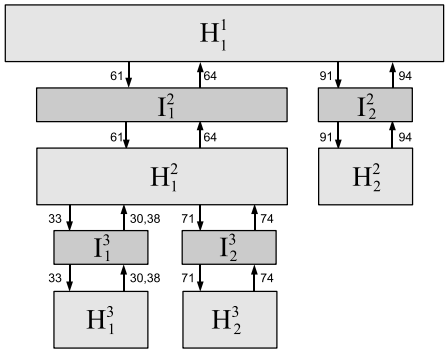

Fig. 8. Hierarchy imposed on flexible manufacturing system example

For comparison, the composition of all plant and specification components in this example results in an automaton with 291,456 states and 1,226,672 transitions. Furthermore, the resulting closed-loop behavior is generated by an automaton with 20,232 states and 80,028 transitions. The local modular solution of [15] greatly reduces the complexity of synthesizing the control for this example, but results in blocking.

In the generation of the multiple-level hierarchical interface-based control, the largest automaton that was constructed had 80 states and 182 transitions. This automaton was built in the process of constructing S_1^2. Furthermore, the resulting global closed-loop behavior is safe and nonblocking. The reduced size of the automata built in the process of constructing this solution gives some indication of its advantage. A drawback, however, is that this interface-based control only allows four pieces to be active in the factory at any given time. The monolithic solution allows a maximum of eight pieces to be active at one time.

If instead a two-level architecture is employed with low-level modules consisting of H_2^2, H_1^3, and H_2^3, and a high-level module made up of the remaining components, then the largest automaton that needs to be constructed has 320 states and 888 transitions. This solution allows a maximum of five pieces to be active at any given time.

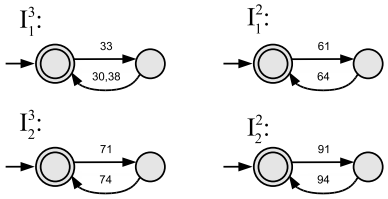

Fig. 9. Proposed interfaces for flexible manufacturing example

V. Conclusion

In this paper we have provided requirements for a multiple-level interface-based architecture by which global controllability and nonblocking can be verified locally. This general architecture is an improvement over the special two-level case of [3] in that it allows the global system to be partitioned into smaller modules, thereby leading to less complexity and improved reconfigurability, though at the possible expense of increased restrictiveness. Furthermore, the interface consistency requirements of this paper are shown to be a relaxation of the corresponding requirements of [11]. The benefits of this architecture are also demonstrated through its application to a manufacturing example.

References

[1] E. Endsley, E. Almeida, and D. Tilbury, "Modular finite state machines: Development and application to reconfigurable manufacturing cell controller generation," *Contr. Eng. Pract.*, 2006.

[2] R. J. Leduc, B. A. Brandin, M. Lawford, and W. Wonham, "Hierarchical interface-based supervisory control–part I: Serial case," *IEEE Trans. Auto. Contr.*, 2005.

[3] R. J. Leduc, M. Lawford, and W. Wonham, "Hierarchical interface-based supervisory control–part II: Parallel case," *IEEE Trans. Auto. Contr.*, 2005.

[4] K. Schmidt, T. Moor, and S. Perk, "A hierarchical architecture for nonblocking control of discrete event systems," in *Mediterranean Conf. Control and Automation*, 2005.

[5] R. Hill and D. Tilbury, "Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction," in *Proc. WODES*, 2006.

[6] L. Feng and W. Wonham, "Computationally efficient supervisor design: Abstraction and modularity," in *Proc. WODES*, 2006.

[7] P. Pena, J. Cury, and S. Lafortune, "Testing modularity of local supervisors: An approach based on abstractions," in *Proc. WODES*, 2006.

[8] H. Flordal and R. Malik, "Modular nonblocking verification using conflict equivalence," in *Proc. WODES*, 2006.

[9] B. A. Brandin, R. Malik, and P. Malik, "Incremental verification and synthesis of discrete-event systems guided by counter examples," *IEEE Trans. Contr. Sys. Tech.*, 2004.

[10] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Kluwer Academic Publishers, 1999.

[11] R. J. Leduc, M. Lawford, and P. Dai, "Hierarchical interface-based supervisory control of a flexible manufacturing system," *IEEE Trans. Contr. Sys. Tech.*, 2006.

[12] R. Hill, "Modular verification and supervisory controller design for discrete-event systems using abstraction and incremental construction," Ph.D. dissertation, University of Michigan, Ann Arbor, USA, 2008.

[13] R. Leduc, "Hierarchical interface-based supervisory control," Ph.D. dissertation, University of Toronto, Toronto, Canada, 2002.

[14] M. H. de Queiroz, J. E. R. Cury, and W. Wonham, "Multitasking supervisory control of discrete-event systems," *Discrete Event Dynamic Systems: Theory and Application*, 2005.

[15] M. H. de Queiroz and J. E. R. Cury, "Modular supervisory control of composed systems," in *Proc. ACC*, 2000.