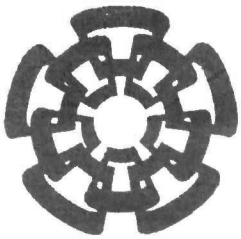
xx (147071.1)



Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Síntesis de Arquitecturas de Coordinación en Sistemas Automatizados de Manufactura

Tesis que presenta:

Liz Eréndira Llamas López

para obtener el grado de:

Maestro en Ciencias

en la especialidad de:

Ingeniería Eléctrica

Directores de Tesis

Dr. Arturo del Sagrado Corazón Sánchez Carmona
Dr. Raúl Ernesto González Torres

Guadalajara, Jalisco, Marzo de 2008.

Síntesis de Arquitecturas de Coordinación en Sistemas Automatizados de Manufactura

Tesis de Maestría en Ciencias
Ingeniería Eléctrica

Por:
Liz Eréndira Llamas López
Ingeniero en Sistemas Computacionales
Instituto Tecnológico de Tepic 1999-2003

Directores de Tesis
Dr. Arturo del Sagrado Corazón Sánchez Carmona
Dr. Raúl Ernesto González Torres

CINVESTAV del IPN Unidad Guadalajara, Marzo de 2008.

Resumen

En este trabajo de investigación se realiza el diseño e implementación de diferentes estructuras de control supervisor jerárquicas y modulares para la coordinación de actividades de un prototipo de un sistema automatizado de manufactura flexible hecho en *Lego*®. El modelado de las estructuras del prototipo se basó en el modelo híbrido construido con los estándares industriales ISA 88 e ISA 95, permitiendo así separar las actividades de control de equipo de las actividades propias de producción. Por lo tanto, no es necesario modificar los supervisores que controlan los equipos para cambiar las órdenes de producción. La construcción en general de las estructuras del prototipo se realizó con la teoría de control supervisor (TCS). La construcción de la 1^{ra}, 2^{da} y 4^{ta} estructuras está respaldada por las metodologías citadas en [20] (capítulo 5), en [10], y en ([9] (los conceptos de sistema confiable, lenguaje realizable) y en la sección 4 de este trabajo (teorema 4.1 y 4.2)) respectivamente que garantizan la controlabilidad y el no bloqueo global La 3^{ra} estructura de control jerárquico fue propuesta en [17]. Sin embargo, en las 3^{ra} y 4^{ta} estructuras existe un método para construir los componentes de los módulos del nivel superior y métodos automatizados para la validez de la teoría; y requieren menos recursos en su implementación, así como son mas fáciles de construir. Por otra parte, en la 2^{da} estructura solo si cualquier interfaz de un módulo del nivel inferior es cambiada entonces también es cambiada el nivel superior. En el caso de que otra cosa sea modificada en cualquier módulo del nivel inferior, solo se modifica el nivel inferior; y esta también requiere menos recursos en su implementación. Por lo tanto, se concluye que la mejor de las estructuras por sus características es la 4^{ta} estructura, aún cuando todas cumplen con el mismo propósito de control. Las estructuras del prototipo se implementaron en Simulink de Matlab, utilizando dos bibliotecas de Simulink (la biblioteca "Control Supervisor Lego" desarrollada por [17]; y la otra biblioteca "Control Supervisor (Wonham)" desarrollada en este trabajo) que permiten la construcción y el análisis de la estructura de control jerárquico y modular de forma gráfica.

Abstract

This thesis realizes the design and implementation of different hierarchicals and modulates control structures of a prototype automated manufacturing system built in Lego. The modeling methodology of the prototype structures is based in the coordination architecture built with industrial standards ISA 88 and ISA 95, it is permitting so separate the activities of equipment control of the own activities of production. Therefore, it is not necessary to modify the supervisors that control the equipments for change the production orders. The building in general of the prototype structures are realized with supervisor control theory (SCT). The construction of the first, second and fourth structures is supported for the methodologies cited in [20](chapter 5), in [10], and in ([9] (the concepts of reliable system and language achievable) and in the section 4 of this wok (theorem 4.1 and 4.2)) respectively that guarantee the global controllability and the global nonblocking. The third hierarchical control structure was proposed in [17]. However, in the third and fourth structures existence a method to constructs the components of the modules of the high level and the automation method for the validation of the theory, and it requires less resource in its implementation, such as it is easier to built. On the other hand, in the second structure only if anyone interface of a module of low level is updated then also is updated the high level. In the case of that other thing is updated in anyone module of low level, it only is update the low level, and this also requires less resource in its implementation Therefore, it is concluded that the better of the structures for its characteristics is the fourth structure, even when all fulfill with the same control aim. The structures of prototype are implemented in Simulink at Matlab, it used tow libraries of Simulink (The "Control Supervisor Lego" library was developed for [17]; and the other "Control Supervisor (Wonham)" library was developed in this work) that permit the construction and the analysis of the way graphic of the hierarchical and modular control structure.

Agradecimientos

A Dios por acompañarme cada día de mi vida, dándome salud, fortaleza, esperanza y luz a mi vida.

A mis asesores Dr. Raúl Ernesto González Torres y Dr. Arturo Sánchez Carmona por su tiempo, supervisión y profesionalismo.

Al CONACYT por su apoyo con la beca número 199615, para la realización de esta tesis.

A mi mamá María Altagracia por ser mi inspiración, fortaleza, apoyo y lo más grande que tengo en la vida.

A mi papá Erasmo y el resto de mi familia por su apoyo incondicional.

A mi hermano Felipe Gabriel por ser mi inspiración, mostrándome que tan lejos se puede llegar.

A mi hermano Erasmo Otoniel por ser la tenacidad, mostrándome que siempre hay un mañana aún cuando la vida sea difícil.

A mi hermana Ana Irene por ser la fortaleza, que siempre me acompaña.

A mis sobrinos Carlos Xavier, Sofía Carolina, Carlos Gabriel y Felipe Alejandro por ser la esperanza de un futuro.

A mi novio Moisés por su amor y apoyo incondicional.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Los sistemas automatizados de manufactura (SAM) actuales en la industria tienen que satisfacer las demandas del mercado como una mayor variedad de producto, mayor calidad, a un mejor precio, en un menor tiempo y costo. Para que los SAM logren esto, ellos realizan un conjunto de procesos (cambios químicos y físicos que alteran la geometría, las propiedades o el aspecto de un determinado material o ensambles de partes, etc.), utilizando un conjunto de recursos (estaciones de trabajo automatizadas, robots industriales, etc.) para la fabricación de un producto. Por lo tanto, la dificultad radica en que actualmente la cantidad de procesos y recursos es muy grande haciendo así que los SAM y su control sean muy complejos.

En el contexto de la investigación los SAM son tomados como sistemas de eventos discretos (SEDs). Para el control de los SEDs se han publicado nuevas técnicas y herramientas que abordan este tema, donde la teoría de control supervisor (TCS) citada en [20] es una de las técnicas que más se ha desarrollado, debido a que los conceptos de síntesis y modelado de SEDs están basados en nociones matemáticas sencillas; también se ha demostrado su gran potencial de aplicación en los SEDs [4, 6, 7, 10, 14, 16, 20, 22, 21]. Sin embargo, como las aplicaciones en el mundo real de los SAM son más complejas, los investigadores se han enfocado en atacar uno de los problemas (explosión de estados) presentado en el proceso de síntesis de supervisores, originado por el gran tamaño existente actualmente en los SAM.

Algunos autores han abordado este problema descentralizando el control de los SAM, modularizando el SAM y combinando la modularización con la descentralización.

Otros autores han creando técnicas para jerarquizar el control de un SAM. En [20, 7, 10, 16] se permite crear una estructura de control jerárquico de dos niveles basadas en estados vocalizados, proyecciones naturales e interfaces.

La mayoría de las técnicas propuestas para el control de los SAM consideran el producto

que se desea fabricar como una especificación más del sistema. De esta forma el supervisor sintetizado posee una estructura muy rígida de acuerdo al producto deseado. Una importante restricción de esta estrategia de modelado es que si se desea modificar las órdenes de producción, práctica muy común en los sistemas de producción actuales, es necesario sintetizar nuevamente el supervisor. Sin embargo, es posible separar las actividades de control de equipo de las actividades propias de producción, mediante la síntesis de supervisores que establecen rutas de manufactura realizables para productos específicos, llamados supervisores de producto. Esta estrategia es propuesta en [9] e implementada en [8].

También algunos autores como [9, 13, 15] se han basado en los estándares industriales ISA para establecer una metodología de modelado de los SAM y algunos trabajos [10, 17] se han demostrado en un prototipo de un SAM.

En este trabajo de investigación se realiza el diseño e implementación de diferentes estructuras de control jerárquicas y modulares de un prototipo de un SAM hecho en *Lego*® [17] con una interfaz electrónica destinada al acondicionamiento de señales y la comunicación entre el prototipo y una PC [5].

Este prototipo es un tipo de SAM llamado sistema flexible, el cual tiene un mediano volumen de producción continua de mezclas variables de producto. La estructura y su funcionamiento del SAM exhibe un comportamiento con cierto grado de complejidad, representativo del comportamiento en sistemas mas grandes donde si se presenta, durante la síntesis, el problema de explosión de estados.

En este caso la metodología de modelado del prototipo del SAM se basó en el modelo híbrido construido con los estándares industriales ISA 88 y 95 [2, 1] en [17], permitiendo así separar las actividades de control de equipo de las actividades propias de producción. Por lo tanto, no es necesario modificar los supervisores que controlan los equipos para cambiar las órdenes de producción.

La construcción en general de las estructuras del SAM se realizó con la teoría de control supervisor (TCS). La 1^{ra} estructura se construyó aplicando la metodología (supervisión jerárquica de SED citada en [20] en el capítulo 5) que garantiza la controlabilidad global y el no bloqueo. Sin embargo, no existen métodos para construir las abstracciones del nivel inferior (es a prueba, error y experiencia adquirida) y métodos automatizados para la validez de la teoría, si un módulo del nivel inferior es modificado, el nivel superior también es modificado; esta estructura es difícil de construir y requieren más recursos computacionales en su implementación. La 2^{da} estructura se construyó aplicando la metodología (control jerárquico basada en interfaz citado en [10]) que garantiza la controlabilidad global y el no bloqueo. Sin

embargo, no existen métodos para construir las interfaces de cada modulo del nivel inferior y métodos automatizados para la validez de la teoría, solo si cualquier interfaz de un módulo del nivel inferior es modificada entonces también es modificada el nivel superior. En el caso de que otra cosa sea modificada en cualquier módulo del nivel inferior, solo se modifica ese módulo del nivel inferior; esta estructura es difícil de construir, pero requieren menos recursos computacionales en su implementación. La 3^{ra} estructura de control jerárquico fue propuesta en [17], donde la modularidad de la estructura con alfabetos disjuntos del nivel fase del caso de estudio fue tomado de [20, 21] y la jerarquía de la estructura se construyó aplicando la proyección natural de los eventos con estatus operacional de los supervisores del nivel Fase. Sin embargo, existen métodos para construir los componentes de los módulos del nivel superior y métodos automatizados para la validez de la teoría, si un módulo del nivel inferior es modificado, el nivel superior también es modificado; esta estructura es fácil de construir y requieren menos recursos computacionales en su implementación. La 4^{ta} estructura se construyó aplicando los conceptos de sistema confiable, lenguaje realizable presentados en el capítulo 2, sección 2.2; proyección natural confiable presentada en el teorema 4.1 y lenguaje realizable por contención presentado en el teorema 4.2 establecido en este trabajo de investigación. Todo esto permite garantizar la controlabilidad global y el no bloqueo. Sin embargo, existen métodos para construir los componentes de los módulos del nivel superior y métodos automatizados para la validez de la teoría, si un módulo del nivel inferior es modificado, el nivel superior también es modificado; esta estructura es fácil de construir y requieren menos recursos computacionales en su implementación. Por lo tanto, se concluye que la mejor de las estructuras por sus características es la 4^{ta} estructura, aún cuando todas cumplen con el mismo propósito de control.

La implementación del sistema de control del SAM se realizó en Simulink de Matlab en una PC, utilizando dos bibliotecas de Simulink (la biblioteca "Control Supervisor Lego" desarrollada por [17]; y la otra biblioteca "Control Supervisor (Wonham)" desarrollada en este trabajo de investigación) que permiten la construcción y el análisis de la estructura de control jerárquico y modular de forma gráfica.

El presente documento está organizado como sigue. En el capítulo 2 se proporciona una pequeña introducción a los conceptos básicos de sistemas de eventos discretos (SED), sistema confiable y lenguaje realizable. En el capítulo 3 se describe el caso de estudio y el modelo híbrido. El caso de estudio es el prototipo de un SAM hecho en *Lego*® En el capítulo 4 se obtienen 4 principales estructuras de control jerárquico del caso de estudio. En el capítulo 5 se realiza la implementación de las estructuras de control jerárquico del capítulo 4. Finalmente, en el capítulo 6 se presentan algunas conclusiones del trabajo de investigación y recomendaciones para trabajo futuro.

Capítulo 2

Fundamento Teórico

En este capítulo se proporciona una breve introducción a los conceptos básicos de Sistemas de Eventos Discretos (SED), sistema confiable y lenguaje realizable.

2.1. Conceptos Básicos de SED

En esta sección se proporciona una breve introducción de los conceptos básicos de Sistemas de Eventos Discretos (SED) enfocados a la TCS. Para más detalles consultar [20].

2.1.1. Autómata

En la Teoría de Control Supervisor [20] se da un marco de trabajo para modelar un comportamiento de SED usando autómatas finitos llamados generadores, así como ciertas operaciones comunes, tales como: la proyección natural, proyección inversa, el producto síncrono y asíncrono.

Definición 2.1. (Generador). Un generador es un autómata finito $G := (Q,\ \Sigma,\ \delta,\ q_0,\ Q_m)$ que consiste de :

Q, un conjunto de estados

Σ, conjunto de eventos $\Sigma = \Sigma_c \cup \Sigma_u$

Donde Σ_u es el conjunto de eventos incontrolables y Σ_c es el conjunto de eventos controlables.

$\delta : Q \times \Sigma \rightarrow Q$, función parcial $\Sigma_c \cap \Sigma_u = \varnothing$

Q_m, $Q_m \subseteq Q$ estados marcados

q_0, estado inicial

El lenguaje generado o comportamiento cerrado de G está definido por:

$$L(G) = \{s \in \Sigma^* | \delta(q_0, s)!\}$$

El lenguaje marcado es:

$$L_m(G) = \{s \in \Sigma^* | \delta(q_0, s) \in Q_m\}$$

$L(G)$ contiene todas las cadenas que pueden ocurrir en G y $L_m(G)$ contiene todas las cadenas que son de aceptación, es decir, aquellas cadenas que son originadas desde un estado inicial y finalizan en un estado marcado.

Cerradura de Prefijo $\overline{H}$ es aquella que contiene todas las cadenas de H (lenguaje de un generador) y los prefijos de dichas cadenas, es decir:

$$\overline{H} = \{s \in \Sigma^* | \exists t \in \Sigma^* \; tal \; que \; st \in H\}$$

Un estado q es alcanzable si existe una secuencia de eventos s que llevan de un estado inicial q_0 a q. El conjunto de estados alcanzables de G es:

$$Q_{cr} = \{q \in Q | (\exists s \in \Sigma^*) \, \delta(q_0, s) = q\}$$

Un estado es coalcanzable si a partir de este, mediante una secuencia de eventos se llega a un estado marcado. El conjunto de estados coalcanzables de G es:

$$Q_r = \{q \in Q | (\exists s \in \Sigma^*) \, \delta(q, s) \in Q_m\}$$

G es no-bloqueante si cada estado alcanzable es coalcanzable, es decir:

$$\overline{L_m}(G) = L(G)$$

Definición 2.2. (Proyección Natural). Dado $\Sigma_i \subseteq \Sigma$, la proyección natural $P_i : \Sigma^* \longrightarrow \Sigma_i^*$, está definida como sigue:

$P_i(\epsilon) = \epsilon$

$P_i(\sigma) = \epsilon \quad Si \quad \sigma \notin \Sigma_i$

$P_i(\sigma) = \sigma \quad Si \quad \sigma \in \Sigma_i$

$P_i(s\sigma) = P_i(s)P_i(\sigma) \quad s \in \Sigma^* \quad \sigma \in \Sigma$

Para toda cadena $s \in \Sigma^*$, $P_i(s)$ filtra todos los eventos que pertenecen a $\Sigma - \Sigma_i$.

Definición 2.3. (Proyección Inversa). Dado $\Sigma_i \subseteq \Sigma$, la proyección inversa $P_i^{-1} : Pwr(\Sigma_i^*) \rightarrow Pwr(\Sigma^*)$, es la imagen inversa de la función P_i. Para $L \subseteq \Sigma_i$ obtenemos:

$$P_i^{-1}(L) = \{s \in \Sigma^* | P_i(s) \in L\}$$

Definición 2.4. (Producto Síncrono). Para $L_1 \subseteq \Sigma_1^*$ y $L_2 \subseteq \Sigma_2^*$ se define el producto síncrono de L_1 y L_2 como sigue:

$$L_1 \| L_2 = P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$$

Por lo tanto, $s \in L_1 \| L_2$ *sii* $P_1(s) \in L_1$ y $P_2(s) \in L_2$

Definición 2.5. (Producto Síncrono Total de Autómatas Finitos). El producto síncrono total de dos generadores G_1 y G_2, donde $\Sigma_1 = \Sigma_2$, es:

$G_1 \times G_2 = (Q, \Sigma, \delta, q_0, Q_m)$

$Q = Q_1 \times Q_2$

$\delta = \delta_1 \times \delta_2 \longrightarrow (\delta_1 \times \delta_2)(q_1 q_2, \sigma) = [\delta_1(q_1, \sigma)! \wedge \delta_2(q_2, \sigma)!]$

Definición 2.6. (Producto Asíncrono Total de Autómatas Finitos). El producto asíncrono total de dos generadores G_1 y G_2 donde $\Sigma_1 \cap \Sigma_2 = \oslash$, es:

$G_1 \times G_2 = (Q, \Sigma, \delta, q_0, Q_m)$

$Q = Q_1 \times Q_2$

$\delta = \delta_1 \times \delta_2 \longrightarrow (\delta_1 \times \delta_2)(q_1 q_2, \sigma) = [\delta_1(q_1, \sigma)! \vee \delta_2(q_2, \sigma)!]$

Definición 2.7. (No Conflicto). Dos lenguajes L_1 y L_2 se dice que son libres de conflicto si:

$$\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$$

2.1.2. Control de Supervisión

Los conceptos que se presentan en esta subsección son la base de la TCS. Para más detalles consultar [20, 3].

El esquema de control se ilustra en la figura 2.1. La planta es el sistema de eventos discretos representado por un autómata de acuerdo a la Definición 2.1. La planta, en un estado x, envía la información al controlador de los eventos que pueden ser habilitados en ese estado. El control, entonces habilita un subconjunto de eventos $\gamma(x) \subseteq \Sigma$, esto es, permite bajo ciertas consideraciones la habilitación de ciertos eventos. El control solo puede deshabilitar eventos controlables y dejar que los eventos incontrolables siempre ocurran en ese estado.

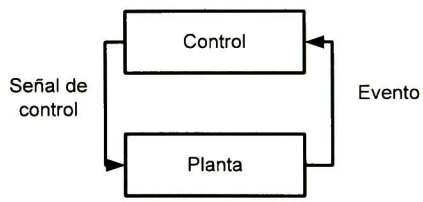


Figura 2.1: Esquema de Control.

El patrón de control $\gamma(x)$ es un subconjunto de eventos controlables que pueden ser habilitados en el estado x. El conjunto de todos los patrones de control es:

$$\Gamma = \{\gamma \in Pwr(\Sigma) | \Sigma_u \subseteq \gamma\}$$

Entonces, un control supervisor para G es un mapeo $V : L(G) \rightarrow \Gamma$ y se escribe como V/G para indicar que G está bajo la supervisión de V

El comportamiento del sistema a lazo cerrado es $L(V/G) \subseteq L(G)$ se determina como:

- $\epsilon \in L(V/G)$
- Si $s\sigma \in L(V/G)$, $\sigma \in V(s)$ y $s\sigma \in L(G)$ $entonces$ $s\sigma \in L(V/G)$
- No hay otra cadena que pertenece a $L(V/G)$

El lenguaje marcado de V/G está dado por:

$$L_m(V/G) = L(V/G) \cap L_m(G)$$

Con respecto a G, decimos que V es no bloqueante si:

$$\overline{L_m(V/G)} = L(V/G)$$

Definición 2.8. (Controlabilidad). Se dice que un lenguaje $K \subseteq \Sigma^*$ es controlable con respecto a un autómata G si:

$$\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$$

En otras palabras, K es controlable sii cualquier cadena que pertenece a la cerradura de prefijo de K concatenada con un evento incontrolable y que sea permitida por $L(G)$, debe de estar contenida también en la cerradura de prefijo de K.

Definición 2.9. (Lenguaje Cerrado). Dado un lenguaje $K \subseteq L \subseteq \Sigma^*$ se dice que K es $L - cerrado$ si:

$$K = \overline{K} \cap L$$

Teorema 2.1. *Dado $K \subseteq L_m(G)$, $K \neq \emptyset$. Existe un supervisor no bloqueante V para G tal que $L_m(V/G) = K$ sii:*

- *K es controlable con respecto a G y*
- *K es $L_m(G) - cerrado$*

Teorema 2.2. *Sea $K \subseteq L_m(G)$, $K \neq \emptyset$. Existe un supervisor marcado y no bloqueante V para (K, G) tal que:*

$$L_m(V/G) = K$$

Si y solo si K es controlable con respecto a G.

Síntesis de un Supervisor

En un sentido práctico, un supervisor limita el comportamiento de una planta al deshabilitar determinadas transiciones controlables basado en las especificaciones del sistema.

Las especificaciones del sistema son secuencias deseadas para el comportamiento de la planta. Estas especificaciones pueden ser de seguridad, procedimiento o recursos compartidos. El generador de la especificación se obtiene al sincronizar todas las especificaciones de funcionamiento individuales.

El supervisor es el supremo sublenguaje controlable [20] que cumple con todas las especificaciones planteadas para el sistema y que, como su nombre lo indica, produce un lenguaje controlable según [20] para restringir el comportamiento de la planta.

La síntesis de los supervisores presentada en este trabajo fue obtenida utilizando TCT[1] SSPC[2] y Supremica[3]

2.2. Sistema Confiable y Lenguaje Realizable

La construcción de un control de supervisión que garantice la fabricación de un producto dado está basada en la noción de un sistema confiable y lenguaje realizable, cuyas definiciones son presentadas en esta sección. Para más detalles consultar [8].

A causa de los conceptos de sistema confiable y lenguaje realizable, $\Sigma = \Sigma_c \cup \Sigma_u$ se clasifica en: eventos imperativos Σ_{imp} los cuales son eventos controlables que describen la inicialización de una tarea ó proceso, es decir, $\Sigma_{imp} \subseteq \Sigma_c$. El resto de los eventos son llamados informativos, es decir, $\Sigma_{inf} \subseteq \Sigma$; donde $\Sigma_{imp} \cap \Sigma_{inf} = \varnothing$.

Definición 2.10. (Estado de Decisión). Para un autómata G dado, un estado $q \in Q$ se dice que es un estado de decisión si, para cualquiera $\sigma \in \Sigma$ tal que $\delta(q, \sigma)$ está definida, se cumple que $\sigma \in \Sigma_c$ y existe al menos una $\rho \in \Sigma_{imp}$ tal que $\delta(q, \rho)$ también está definida.

Es decir, un estado $q \in Q$ se dice que es un estado de decisión si todos los eventos habilitados en él son controlables y al menos uno de ellos es imperativo.

[1] Para mayor información www.control.utoronto.ca/people/profs/wonham/wonham.html
[2] Para mayor información www.gdl.cinvestav.mx/arturo/sspc/html/intro_2.htm
[3] Para mayor información www.supremica.org/

Definición 2.11. (Estado Lograble). Para un autómata G dado, un estado $\widetilde{q} \in Q$ se dice que es un estado lograble de $q \in Q$ y es denotado como $q \dashrightarrow \widetilde{q}$ si existe $s \in \Sigma_{inf}^{*}$, tal que $\delta(q, s) = \widetilde{q}$.

Un estado es lograble desde otro estado si existe un camino del estado de origen al estado lograble formado sólo con eventos informativos.

Definición 2.12. (Autómata Confiable). Un autómata G se dice que es un autómata confiable si todos los eventos imperativos σ cumplen:

(1) Para toda $q \in Q$ con $\delta(q, \sigma)$ definida, existe un estado de decisión $\widetilde{q}$ tal que q es lograble, es decir, $q \dashrightarrow \widetilde{q}$.

(2) $\delta(\widetilde{q}, \sigma)$ también está definida.

En un autómata confiable todos los eventos imperativos pueden eventualmente ser ejecutados. El siguiente algoritmo verifica si un autómata G es confiable.

Sea $f : \Sigma_{imp} \longrightarrow 2^{Q}$ donde $f(\sigma)$ es un conjunto de estados de decisión, donde existe una transición $\sigma \in \Sigma_{imp}$.

Sea $g : \Sigma_{imp} \longrightarrow 2^{Q}$ donde $g(\sigma)$ es un conjunto de estados, donde existen dos transiciones $\sigma \in \Sigma_{imp}$ y $\sigma'' \in \Sigma_{u}$, tal que $\delta(x, \sigma)$ y $\delta(x, \sigma'')$ están definidas.

For each $\sigma \in \Sigma_{imp}$ **do**

If $f(\sigma) = \emptyset$ **then**

G no es confiable, finalizar

Else

If $f(\sigma) \neq \emptyset \;\; y \;\; g(\sigma) \neq \emptyset$ **then**

Eliminar todos los eventos imperativos en G

If $g(\sigma) \neq \emptyset \not\subseteq Co(f(\sigma))$ **then**

G no es confiable, finalizar

Next

Return G es confiable

$Co(f)$ es el conjunto coalcanzable de f iterativamente computado como la pre-imagen de f. Es fácil ver que el producto asíncrono de dos autómatas confiables es también un autómata confiable.

Sea $M := (Q,\ \Sigma,\ \delta,\ q_0,\ Q_m)$ un autómata cuyo lenguaje describe el conjunto de secuencias de comandos de fabricación para un producto particular en términos de eventos imperativos.

Definición 2.13. (Lenguaje realizable). Para los autómatas G y M, el lenguaje $L(M)$ es realizable en G si:

(1) $L(M) \subseteq Proj(L(G))$, donde $Proj : \Sigma^* \longrightarrow \Sigma^*_{imp}$

(2) Si $w_i \in L(G)$ y $w_i \in Proj^{-1}(s_i)$ con $s_i \notin L_m(M)$, entonces existe un estado de decisión $q_i = \delta_G(q_o, z_i)$ tal que $\delta_G(q_o, z_i) \dashrightarrow q_i$ y $Proj(z_i)\xi \in L(M)$, donde ξ está habilitado en q_i.

Si el lenguaje $L(M)$ es realizable sobre el autómata a lazo cerrado V/G entonces todas las secuencias de fabricación en $L(M)$ pueden ser seguidas evento por evento en el autómata a lazo cerrado V/G y así todos los eventos imperativos pueden ser ejecutados de una manera confiable.

Proposición 2.1. *Dado un autómata confiable G, el lenguaje $L(M)$ es realizable en G si $Proj(L(M)\|L(G))$ es isomorfo a $L(M)$.*

Capítulo 3

Caso de Estudio: Un Sistema Automatizado de Manufactura

En este capítulo se describe el caso de estudio (SAM) y el modelo híbrido del SAM. En la sección 3.1 se da una descripción detallada de la estructura y el funcionamiento del sistema automatizado de manufactura (SAM), el cual es un prototipo construido con *Lego*® y cuenta con una interfaz electrónica que nos permite interactuar con él. Finalmente, en la sección 3.2 se obtiene el modelo híbrido del SAM en base a los estándares ISA95[2] e ISA88[1].

3.1. Descripción del Prototipo

El sistema automatizado de manufactura representa un esquema típico de configuración de una celda de manufactura, el cual consiste de dos despachadores de materia prima A (piezas rojas) y B (piezas negras) y dos estaciones de trabajo unidas por una banda transportadora cuya función es transportar las piezas liberadas por los despachadores hacia las estaciones de trabajo. En la figura 3.1 se muestra la estructura del SAM.

El sistema automatizado de manufactura fue construido en *Lego*® ya que es una herramienta práctica y económica para construir el prototipo.

En las secciones 3.1.1, 3.1.2 y 3.1.3 se da una descripción de los componentes que conforman al SAM . En la sección 3.1.4 se da una descripción de la interfaz electrónica. Finalmente, en la sección 3.1.5 se presenta el ensamble final del SAM.

3.1.1. Banda Transportadora

El sistema de la banda transportadora consiste de cuatro bandas (dos largas y dos cortas) que forman un rectángulo que une el área de despachado con las estaciones de trabajo. Un

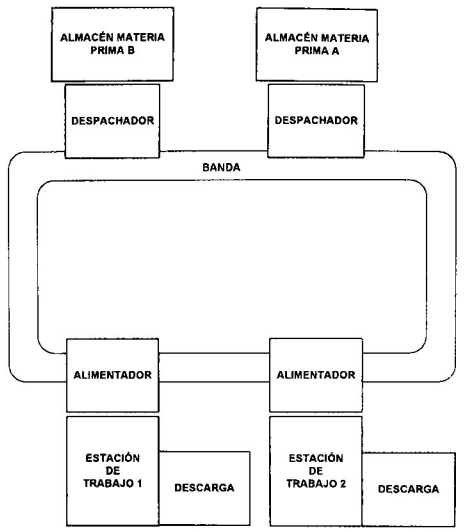


Figura 3.1: Estructura del SAM.

motor impulsa el movimiento de las bandas mediante un sistema de engranes. Para efectos de este trabajo no se diseñó un control para el movimiento de la banda, sino que se considera siempre en movimiento.

3.1.2. Despachador de Materia Prima

Un despachador se muestra en la figura 3.2. Una cremallera es desplazada hacia adelante y hacia atrás mediante el giro de un engrane que es movido por un motor mediante un sistema de banda y polea. Este movimiento permite que una pieza (materia prima) se deslice de la rampa al canal de la cremallera y sea depositada sobre la banda trasportadora.

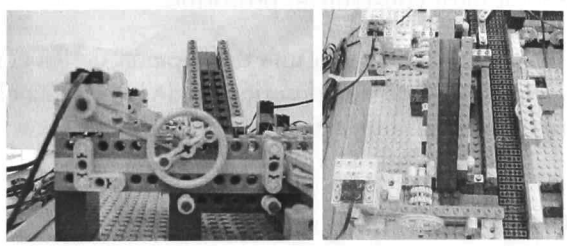


Figura 3.2: Despachador de Materia Prima.

3.1.3. Estación de Trabajo

La estación de trabajo está constituida por un alimentador de materia prima (que alimenta las piezas de la banda hacia el área de trabajo de la estación), maquinado y entrega (que procesa las piezas y entrega el producto terminado), los cuales son descritos acontinuación.

Alimentador de Materia Prima

La figura 3.3 muestra el alimentador de materia prima de la estación de trabajo. Este alimentador está compuesto de un sensor de luz un sensor de tacto y un motor. Este alimentador está colocado sobre el sistema de bandas. El sensor de luz detecta el color de la pieza que pasa sobre la banda y entonces la pieza es introducida al proceso por medio de un eslabón (pateador) que está unido al eje del motor. Al girar el brazo de este eslabón presiona el sensor de tacto que indica que el pateador ha dado una vuelta.

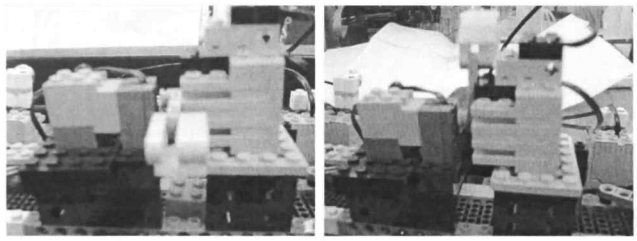


Figura 3.3: Alimentador de Materia Prima.

Maquinado y Entrega

El diseño de la máquina de procesos y el mecanismo de salida de piezas es mostrado en la figura 3.4. El maquinado de la pieza solo es simulado. La máquina cuenta con espacio para 2 piezas. La orden de iniciar el maquinado inicia con un temporizador que simula el inicio y el fin del maquinado. Cuando el tiempo es cumplido se envía una señal de "fin de maquinado" El sistema de salida de las piezas (producto final) consiste de un engrane (el cual es movido por un motor mediante un sistema de banda y polea) que mueve una cremallera que empuja las piezas hacia afuera de la máquina.

3.1.4. Interfaz Electrónica

La figura 3.5 muestra la interfaz electrónica que es destinada al acondicionamiento de señales y la comunicación de las mismas entre un sistema automatizado de manufactura

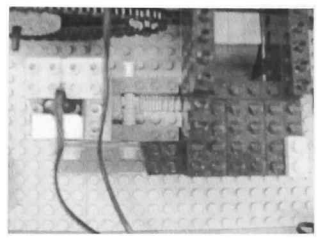


Figura 3.4: Maquinado y Entrega.

(SAM) implementado en *Lego*® y una PC que cuenta con un sistema desarrollado en Matlab y una tarjeta de adquisición de datos Sensoray© modelo 626.

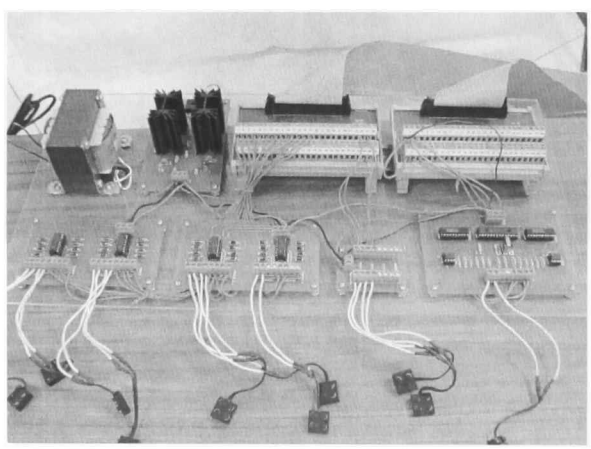


Figura 3.5: Interfaz Electrónica.

La interfaz electrónica está formada por:

- 2 tarjetas electrónicas para los motores, donde cada una permite controlar la dirección del giro de los 4 motores y la cantidad de corriente necesaria.

  1 tarjeta electrónica para 2 sensores de luz.

  1 tarjeta electrónica para 4 sensores de tacto.

  1 fuente de alimentación doble de 9 y 5 V, con un amperaje máximo de 5A.

- 2 racks de conexiones, uno para entradas y salidas digitales y el otro para entradas y salidas analógicas.

EL SAM cuenta con 7 motores (1 en cada despachador, 2 en cada estación de trabajo y 1 en la banda transportadora), 1 sensor de luz y 1 sensor de tacto en cada estación de trabajo. Para más detalle del diseño de la interfaz consultar [5].

3.1.5. Ensamble Final

La figura 3.6 muestra el ensamble final de los componentes del sistema automatizado de manufactura y su interfaz electrónica, obteniendo así un sistema con complejidad suficiente para aplicar estrategias de control de interés. Mas aún, las ventajas de tener el prototipo en *Lego*® es que se pueden construir configuraciones más complejas simplemente reproduciendo más elementos como los descritos anteriormente o introduciendo nuevos equipos al SAM.

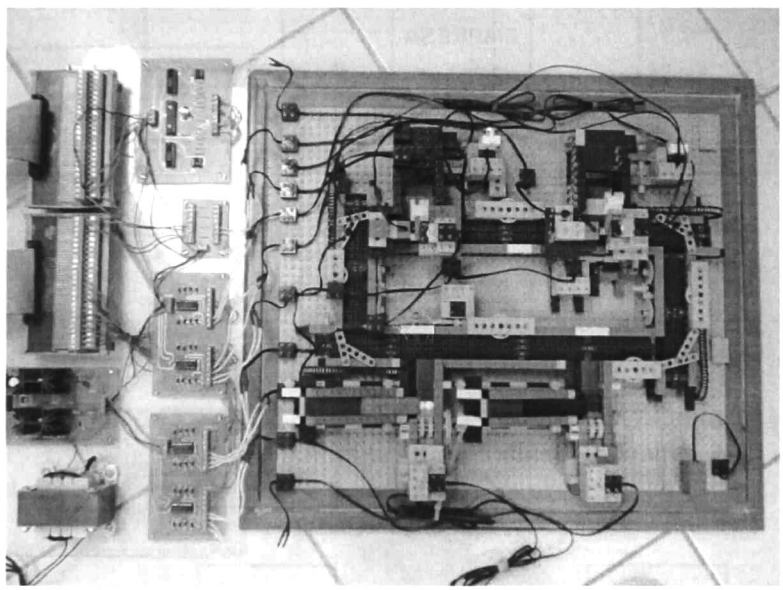


Figura 3.6: Ensamble Final.

3.2. Modelado de Equipo y Procedimiento

En esta sección se obtiene el modelo híbrido del SAM en base a los estándares ISA95[2] e ISA88 [1]. Este modelo se toma como base para construir 4 estructuras de control jerárquico aplicando diferentes enfoques al caso de estudio en el capítulo 4.

El estándar ISA95 [2] proporciona una arquitectura para definir la interfaz entre los sistemas de negocio y los sistemas de control de manufactura dentro de una empresa. El modelo de jerarquía de control y planeación presentado por el estándar está basado en la Vista Jerárquica de Planeación y Control presentada en el Modelo Jerárquico de Purdue [18],[19]. Además de este modelo jerárquico, el estándar presenta una jerarquía de equipos basada en el estándar ISA88[1] que permite organizar los activos presentes en una empresa.

El modelo jerárquico basado en el estándar ISA88 se presenta en la figura 3.7. La agrupación de elementos de un mismo nivel forma elementos de un nivel superior, aunque está permitido que la combinación de elementos resulte en otro elemento del mismo nivel.

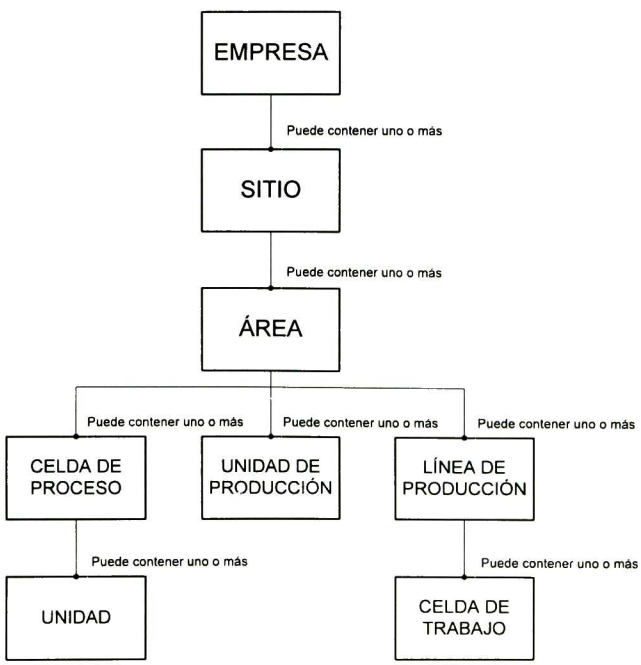


Figura 3.7: Modelo Jerárquico del Estándar ISA88.

En la figura 3.8 se presenta el modelo de referencia de la arquitectura para las actividades realizadas en el nivel de coordinación de recursos de sistemas por lotes, continuos y discretos basado en el estándar ISA88.

El modelo de activos es un agrupamiento jerárquico de los activos físicos. El modelo de control de procedimientos establece las actividades que ejecutan los activos físicos para realizar tareas con un sentido de proceso. El resultado de combinar los equipos con los procedimientos es llamado modelo de proceso.

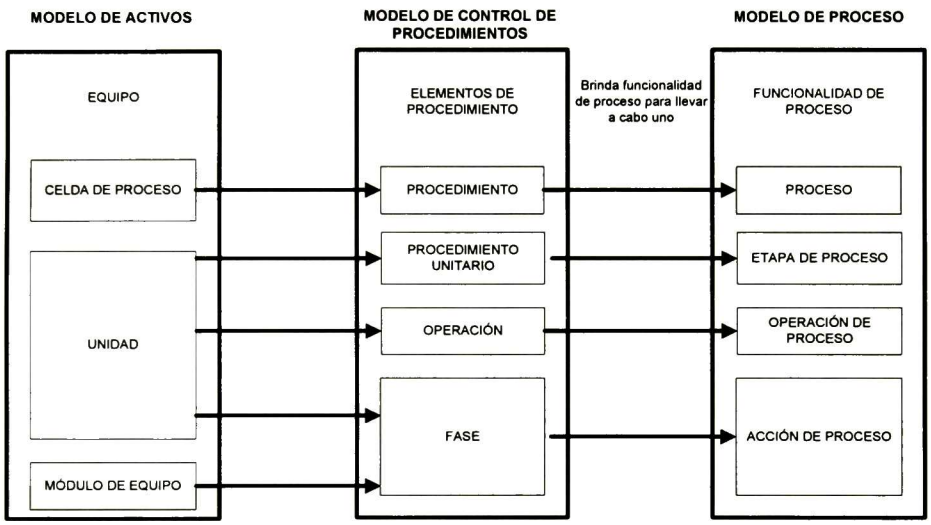


Figura 3.8: Modelo de Referencia de la Arquitectura.

Aplicando lo mencionado anteriormente de los estándares ISA95[2] e ISA88[1] al SAM, se obtienen los modelos de activos, de control de procedimientos y híbrido del SAM.

El modelo híbrido del SAM es de gran importancia porque fue el punto de partida para construir 4 estructuras de control jerárquico aplicando diferentes enfoques al caso de estudio en el capítulo 4.

3.2.1. Modelo de Activos

En la tabla 3.1 tomada de [17] se muestra el ordenamiento jerárquico-modular de los activos del SAM, es decir, se en listan aquellos componentes de cada módulo de equipo relacionados con el control del sistema.

3.2.2. Modelo de Control de Procedimientos

La tabla 3.2 muestra el modelo de control de procedimientos del sistema empleado en [17]. Las acciones descritas para una fase son acciones que deben realizar los equipos que participan en esa fase. En la figura 3.9 se muestra el modelo híbrido del SAM, el cual muestra el modelo de referencia de la arquitectura del SAM de una perspectiva enfocada más a la arquitectura del SAM.

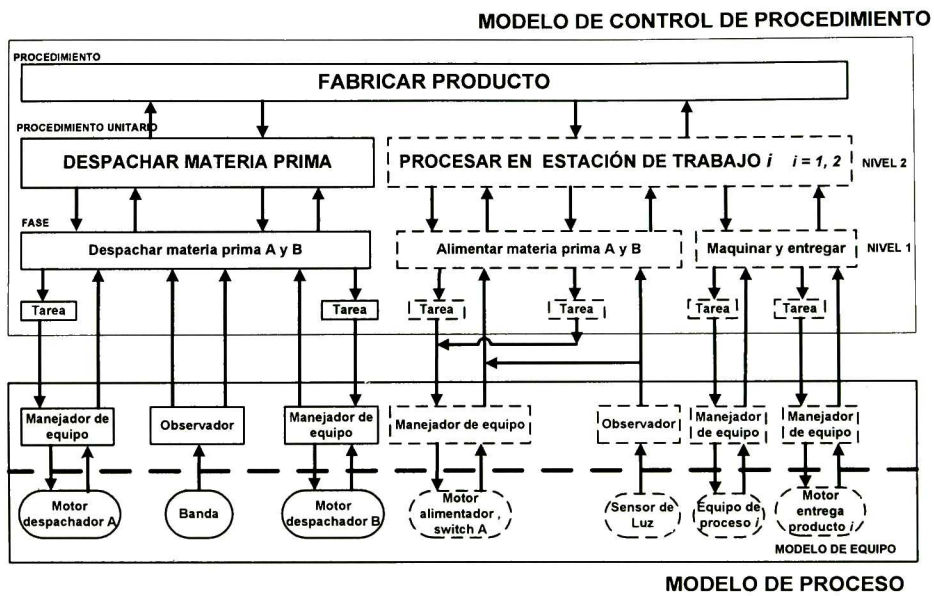


Figura 3.9: Modelo Híbrido del SAM.

Tabla 3.1: Modelo de Activos

Celda de Proceso	Unidad	Módulo de Equipo	Componentes
Celda de Manufactura	Despachadores	Despachadores de materia prima A o B	Motores de despachadores
		Banda	Contador de piezas en banda
	Estación de Trabajo 1	Alimentador de materia prima	Motor, sensor de luz y tacto
		Maquinado	Motor de entrega
	Estación de Trabajo 2	Alimentador de materia prima	Motor, sensor de luz y tacto
		Maquinado	Motor de entrega

Tabla 3.2: Modelo de Control de Procedimientos

Procedimiento	Unidad Procedimiento	Fase De Equipo	Acción
PRODUCIR PRODUCTO	DESPACHAR MATERIA PRIMA	Despachadores de Materia Prima A o B	1. Verificar si no hay más de 4 piezas en banda 2. Si hay espacio en banda, entonces: a)Accionar motor en una dirección por 2 seg. b)Accionar motor en otra dirección por 2 seg.
	PROCESAR MATERIA PRIMA EN ESTACIÓN DE TRABAJO 1	Alimentar Materia Prima A o B	1. Detectar una pieza (tipo A o B) 2. Accionar motor en una dirección hasta que el sensor de tacto se presione 3 veces y entonces detenerlo.
		Maquinar y sacar las piezas	1. Iniciar máquina 2. Detener máquina 3. Accionar el motor de salida de piezas en una dirección por 1 seg. 4. Accionar el motor de salida de piezas en otra dirección por 1 seg.
	PROCESAR MATERIA PRIMA EN ESTACIÓN DE TRABAJO 2	Alimentar Materia Prima A o B	1. Detectar una pieza (tipo A o B) 2. Accionar motor en una dirección hasta que el sensor de tacto se presione 3 veces y entonces detenerlo.
		Maquinar y sacar las piezas	1. Iniciar máquina 2. Detener máquina 3. Accionar el motor de salida de piezas en una dirección por 1 seg. 4. Accionar el motor de salida de piezas en otra dirección por 1 seg.

Capítulo 4

Aplicación del Control Jerárquico al Caso de Estudio

La descentralización del control en forma jerárquica y modular permite combatir el problema de explosión de estados que suele presentarse durante la síntesis. Por este motivo, en este capítulo se obtienen 4 estructuras de control jerárquico del caso de estudio: las dos primeras estructuras se basan respectivamente en las metodologías de control jerárquico presentadas en [20](capítulo 5) y [10]; la tercer estructura de control jerárquico fue propuesta en [17] y la última está basada en los conceptos de sistema confiable y lenguaje realizable dados en [9]; permitiéndonos así conocer sus ventajas y desventajas respectivamente de cada estructura.

En éstas 4 estructuras de control jerárquico, solo se presenta la síntesis de los supervisores que pertenecen a los módulos de Despachar Materia Prima y Procesar en Estación de Trabajo 1 del modelo híbrido del SAM obtenido en el capítulo 3, sección 3.9. La síntesis de los supervisores que pertenecen al módulo Procesar en Estación de Trabajo 2 se omite, ya que es semejante a la del módulo Procesar en Estación de Trabajo 1.

En las tablas 4.1, 4.2 y 4.3 se muestran las listas de los eventos utilizados en la síntesis de los supervisores; estos eventos tienen la notación presentada en TCT, es decir, los eventos controlables están representados con numeros impares y los eventos incontrolables con numeros pares.

En la sección 4.1 se aplica la metodología de supervisión jerárquica de SED citada en [20](capítulo 5) y en la sección 4.2 se aplica la metodología de control de supervisión jerárquico basada en interfaz citada en [10]. En la sección 4.3 se presenta el control de supervisión jerárquico y modular basado en la proyección natural del caso de estudio, propuesto en [17]. En la sección 4.4 se aplican los conceptos de sistema confiable, lenguaje realizable dados en

[9], y proyección natural para la construcción del control jerárquico y modular del caso de estudio. Por último, en la sección 4.5 se muestra un análisis de las diferentes estructuras de control jerárquico.

Tabla 4.1: Despachar Materia Prima A o B

Dispositivo	Evento	Descripción	Tipo
Señales de	101	Inicio Fase "Despachar Materia Prima"	Imperativo
software	199	Fin Fase "Despachar Materia Prima"	Informativo
	103	Iniciar proceso de "Despachar Materia Prima A"	Imperativo
	104	Finalizar proceso de "Despachar Materia Prima A"	Informativo
	203	Iniciar proceso de "Despachar Materia Prima B"	Imperativo
	204	Finalizar proceso de "Despachar Materia Prima B"	Informativo
Contador	173	Consultar registro de piezas en banda	Informativo
de	170	Banda no llena (hay menos de 4 piezas)	Informativo
banda	172	Banda llena (hay 4 piezas)	Informativo
Motor del	111	Encender el motor hacia la derecha	Imperativo
despachador	113	Encender el motor hacia la izquierda	Imperativo
A	115	Detener el motor	Imperativo
Temporizador del motor	117	Iniciar Temporizador	Informativo
del despachador A	116	Terminar Temporizador	Informativo
Motor del	211	Encender el motor hacia la derecha	Imperativo
despachador	213	Encender el motor hacia la izquierda	Imperativo
B	215	Detener el motor	Imperativo
Temporizador del motor	217	Iniciar Temporizador	Informativo
del despachador B	216	Terminar Temporizador	Informativo

4.1. Estructura de Control Jerárquico I

En esta sección se aplica la metodología de supervisión jerárquica de SED citada en [20](capítulo 5). Esta estructura de control jerárquico es mostrada en la figura 4.1, la cual consta de dos niveles.

La estructura de control jerárquico está formada por los siguientes componentes: G_{lo} es la PLANTA que representa el comportamiento del sistema en el nivel inferior, C_{lo} es el controlador (representado por una función de mapeo) en el nivel inferior, G_{hi} es la PLANTA (es una abstracción de G_{lo}) del nivel superior, C_{hi} es el supervisor del nivel superior, Inf_{lohi} es el canal de información que permite actualizar a G_{hi} con respecto de G_{lo}, Inf_{lo} e Inf_{hi} son canales de información cuya función es retroalimentar a C_{lo} y C_{hi} con respecto de G_{lo} y G_{hi}; Con_{lo} y Con_{hi} son canales de control que permiten aplicar el control a G_{lo} y G_{hi}; Com_{hilo} es

Tabla 4.2: Estación de Trabajo 1

Dispositivo	Evento	Descripción	Tipo
Señales de software	301	Inicio Fase "Alimentar Materia Prima"	Imperativo
	399	Fin Fase "Alimentar Materia Prima"	Informativo
	303	Iniciar proceso de "Alimentación de Materia Prima A"	Imperativo
	304	Finalizar proceso de "Alimentación de Materia Prima A"	Informativo
	305	Iniciar proceso de "Alimentación de Materia Prima B"	Imperativo
	306	Finalizar proceso de "Alimentación de Materia Prima B"	Informativo
Sensor de luz	307	Se habilita el funcionamiento del sensor de luz	Informativo
	308	El sensor detecta una pieza roja (tipo A)	Informativo
	310	El sensor detecta una pieza negra (tipo B)	Informativo
Motor del alimentador de piezas	311	Encender motor hacia la derecha	Imperativo
	313	Encender motor hacia la izquierda	Imperativo
	312	Sensor de tacto presionado	Informativo
	315	Detener motor	Informativo
Señal de software	501	Inicio Fase "Maquinado y Entrega de Producto"	Imperativo
	599	Fin Fase "Maquinado y Entrega de Producto"	Informativo
	503	Iniciar proceso de "Maquinado y Entrega de Producto"	Imperativo
	504	Finalizar proceso de "Maquinado y Entrega de Producto"	Informativo
Proceso de maquinado	509	Inicia Maquinado	Imperativo
	508	Termina Maquinado	Informativo
Motor de sacar piezas	511	Encender motor hacia la derecha	Imperativo
	513	Encender motor hacia la izquierda	Imperativo
	515	Detener motor	Imperativo
Temporizador del motor de sacar piezas	507	Iniciar Temporizador	Informativo
	506	Terminar Temporizador	Informativo

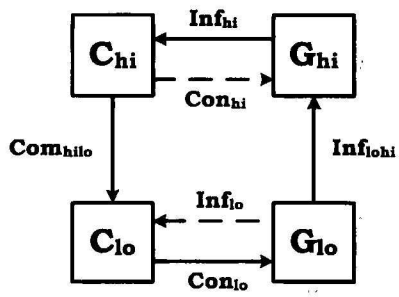


Figura 4.1: Control Jerárquico de Dos-Niveles.

Tabla 4.3: Estación de Trabajo 2

Dispositivo	Evento	Descripción	Tipo
Señal de software	401	Inicio Fase "Alimentar Materia Prima"	Imperativo
	499	Fin Fase "Alimentar Materia Prima"	Informativo
	403	Iniciar proceso de "Alimentación de Materia Prima A"	Imperativo
	404	Finalizar proceso de "Alimentación de Materia Prima A"	Informativo
	405	Iniciar proceso de "Alimentación de Materia Prima B"	Imperativo
	406	Finalizar proceso de "Alimentación de Materia Prima B"	Informativo
Sensor de luz	407	Se habilita el funcionamiento del sensor de luz	Informativo
	408	El sensor detecta una pieza roja (tipo A)	Informativo
	410	El sensor detecta una pieza negra (tipo B)	Informativo
Motor del alimentador de piezas	411	Encender motor hacia la derecha	Imperativo
	413	Encender motor hacia la izquierda	Imperativo
	412	Sensor de tacto presionado	Informativo
	415	Detener motor	Informativo
Señal de software	601	Inicio Fase "Maquinado y Entrega de Producto"	Imperativo
	699	Fin Fase "Maquinado y Entrega de Producto"	Informativo
	603	Iniciar proceso de "Maquinado y Entrega de Producto"	Imperativo
	604	Finalizar proceso de "Maquinado y Entrega de Producto"	Informativo
Proceso de maquinado	609	Inicia Maquinado	Imperativo
	608	Termina Maquinado	Informativo
Motor de sacar piezas	611	Encender motor hacia la derecha	Imperativo
	613	Encender motor hacia la izquierda	Imperativo
	615	Detener motor	Imperativo
Temporizador del motor de sacar piezas	507	Iniciar Temporizador	Informativo
	506	Terminar Temporizador	Informativo

el canal de comando que permite enviar la señal de control como un comando a C_{lo}, éste a su vez tiene que enviar ese comando por el canal de control Con_{lo} para aplicarlo en G_{lo}. Para más detalles, consulte [20](capítulo 5).

La estructura de control jerárquico del caso de estudio es como se muestra en la figura 4.2, la cual está apegada al modelo híbrido del caso de estudio obtenida en el capítulo 3, sección 3.9. La estructura está formada por los siguientes componentes: GD_{lo} es la PLANTA que representa el comportamiento del sistema y CD_{lo} es el controlador (representado por una función de mapeo) del módulo Despachar Materia Prima A y B en el nivel Fase. GA_{lo} es la PLANTA que representa el comportamiento del sistema y CA_{lo} es el controlador (representado por una función de mapeo) del módulo Alimentar Materia Prima A y B en el nivel Fase. GM_{lo} es la PLANTA que representa el comportamiento del sistema y CM_{lo} es el controlador (representado por una función de mapeo) del módulo Maquinar y Entregar en el nivel Fase. GD_{hi} es la PLANTA (es una abstracción de GD_{lo}) y CD_{hi} es el supervisor del módulo Despachar Materia Prima del nivel Procedimiento Unitario. $GPET1_{lo}$ es la PLANTA (es el producto asíncrono de las abstracciones de GA_{lo} y GM_{lo}) y $CPET1_{lo}$ es el controlador (representado por una función de mapeo) del módulo Procesar en la Estación de Trabajo 1 en el nivel Procedimiento Unitario. $GPET1_{hi}$ es la PLANTA (es una abstracción de $GPET1_{lo}$) y $CPET1_{hi}$ es el supervisor del módulo Procesar en la Estación de Trabajo 1 en el nivel Procedimiento Unitario.

En la sección 4.1.1 se obtienen GD_{lo} y GA_{lo}, GM_{lo} para el nivel Fase de la estructura del caso de estudio. En la sección 4.1.2 se obtienen GD_{hi}, $GPET1_{lo}$, $GPET1_{hi}$ y los supervisores CD_{hi} y $CPET1_{hi}$ del nivel Procedimiento Unitario de la estructura del caso de estudio. Finalmente, en la sección (4.1.3) se da un ejemplo de las recetas de producción de los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

4.1.1. Síntesis de Supervisores de Fase

En esta sección se obtienen GD_{lo}, GA_{lo}, GM_{lo} del nivel fase de la estructura del caso de estudio.

En las figuras 4.3, 4.8 y 4.13 se muestran los componentes elementales de cada módulo del nivel Fase, que son los modelos de los equipos físicos que intervienen y cierto comportamiento causal requerido.

En las figuras 4.4, 4.9 y 4.14 se muestran las especificaciones requeridas en cada módulo del nivel Fase, y en las tablas 4.4, 4.5 y 4.6 se describen las especificaciones de cada módulo

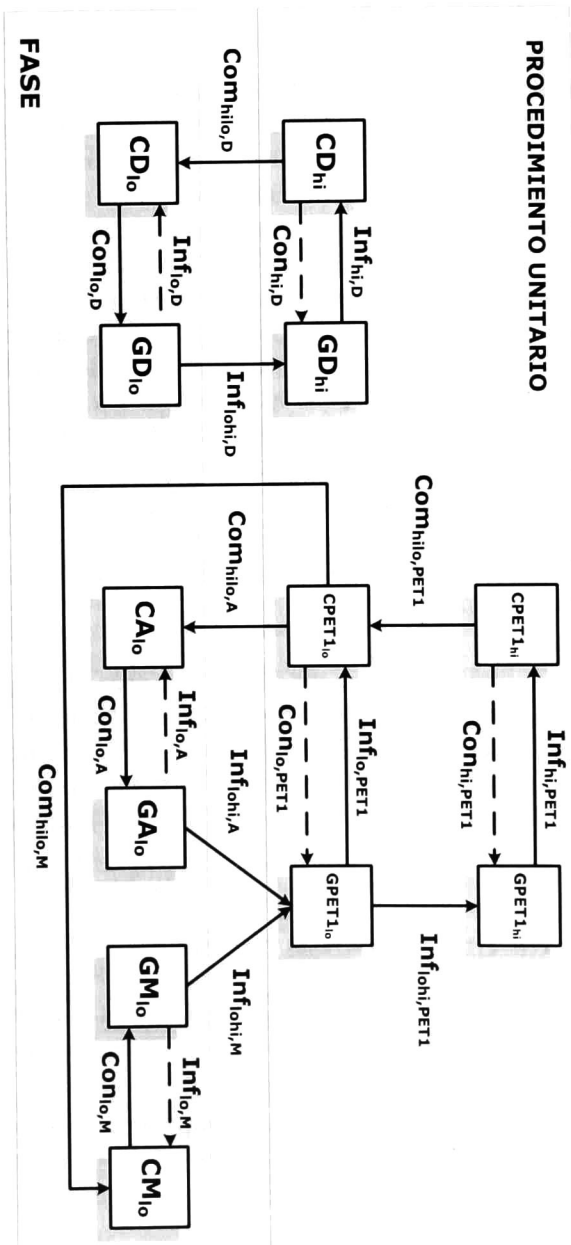


Figura 4.2: Control Jerárquico del Caso de Estudio.

del nivel Fase. De la misma forma las figuras 4.5, 4.10 y 4.15 muestran los supervisores de cada módulo del nivel Fase que son el punto inicial en el desarrollo de la estructura jerárquica.

Estos supervisores son vocalizados como se muestran en las figuras 4.6, 4.11 y 4.16, los cuales de acuerdo a la metodología [20](capítulo 5) son GD_{lo}, GA_{lo} y GM_{lo} del nivel Fase. Los supervisores Despachar Materia Prima A o B y Alimentar Materia Prima A o B, mostrados en las figuras 4.5 y 4.10, fueron modificados en su representación gráfica para poder obtener la abstracción requerida en el siguiente nivel.

La forma de como se vocalizan los estados es de acuerdo a los requerimientos del diseñador, siempre y cuando se respete la metodología dada. La vocalización se realiza usando el procedimiento *Vocalice* de TCT.

Se verifica que GD_{lo}, GA_{lo} y GM_{lo} sean estrictamente consistentes en el control de salida; esto se realiza usando el procedimiento *Hiconsis* de TCT. Para más detalles consultar [20](capítulo 5). Como resultado se obtienen GD_{lo}, GA_{lo} y GM_{lo} tal como muestran las figuras 4.7, 4.12 y 4.17.

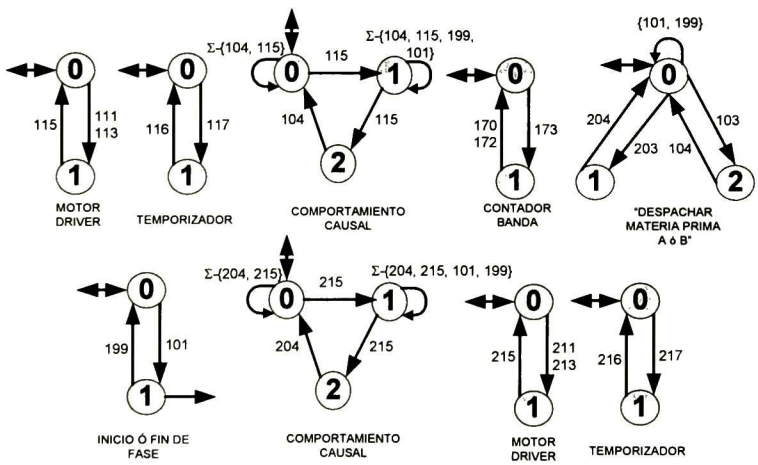


Figura 4.3: Componentes Elementales de Fase: Despachar Materia Prima A o B.

4.1.2. Síntesis de Supervisores de Procedimiento Unitario

En esta sección se obtienen GD_{hi}, $GPET1_{lo}$, $GPET1_{hi}$ y los supervisores CD_{hi} y $CPET1_{hi}$ del nivel Procedimiento Unitario de la estructura del caso de estudio.

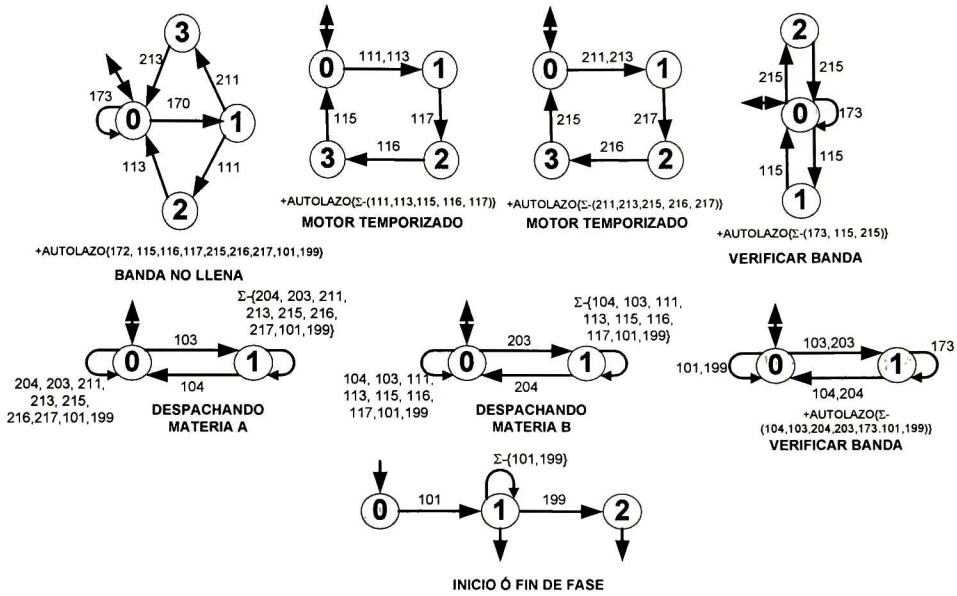


Figura 4.4: Especificaciones de Fase: Despachar Materia Prima A o B.

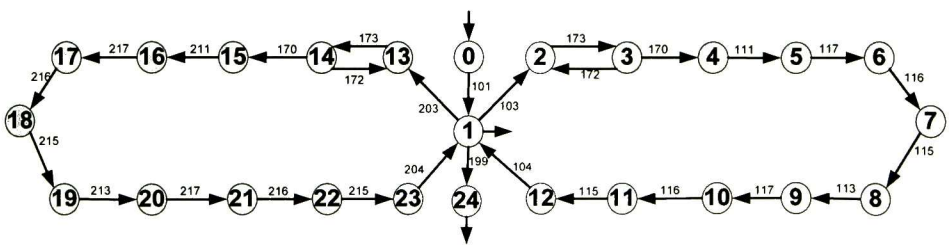


Figura 4.5: Supervisor de Fase: Despachar Materia Prima A o B.

Tabla 4.4: Especificaciones de Fase: Despachar Materia Prima A y B

Especificación	Descripción
Banda no llena	Verifica que la banda no se este llena (Evento 170) para realizar el proceso de despachar materia prima A (Evento 103 y 104) o el proceso de despachar materia prima B (Evento 203 y 204).
Motor Temporizado	Enciende el motor del despachador A hacia la derecha o hacia la izquierda (Eventos 111 o 113), posteriormente espera cierto tiempo dado por el temporizador (Eventos 117 y 116) para detener el motor (Evento 115).
Motor Temporizado	Enciende el motor del despachador B hacia la derecha o hacia la izquierda (Eventos 211 o 213), posteriormente espera cierto tiempo dado por el temporizador (Eventos 217 y 216) para detener el motor (Evento 215).
Verificar Banda	Consulta registros de piezas en la banda (Evento 173), después de detener el motor con el que se despacha la materia prima A o B (Eventos 115 o 215).
Despachando Materia A	Define el proceso de despachar materia prima A (Eventos 103 y 104).
Despachando Materia B	Define el proceso de despachar materia prima A (Eventos 203 y 204).
Verificar Banda	Después de iniciar el proceso de despachar materia prima A o B (Eventos 103 o 203), se consulta registros de piezas en la banda (Evento 173), para así realizar el procedo de de despachar materia prima A o B (Eventos 104 o 204).
Inicio o Fin de Fase	Define la fase de despachar materia prima (Eventos 101 y 199).

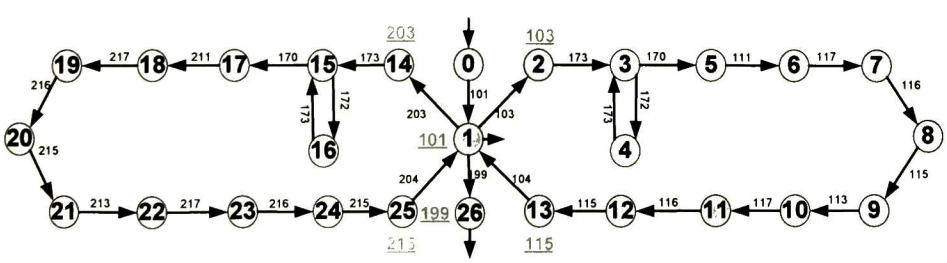


Figura 4.6: GD_{lo}.

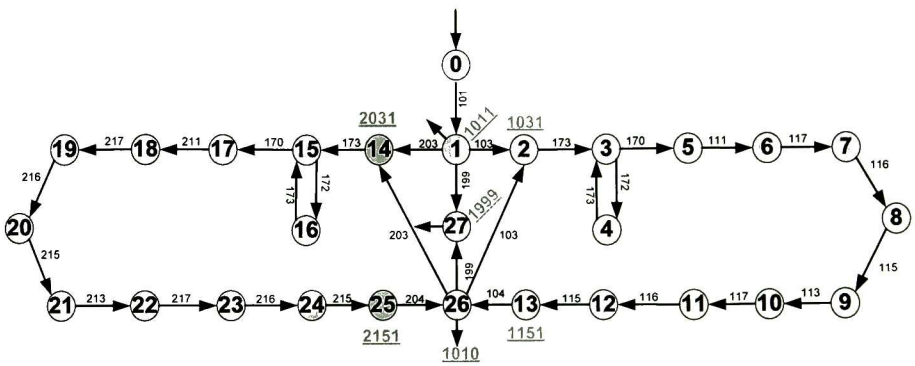


Figura 4.7: GD_{lo} Estrictamente Consistente en el Control de Salida.

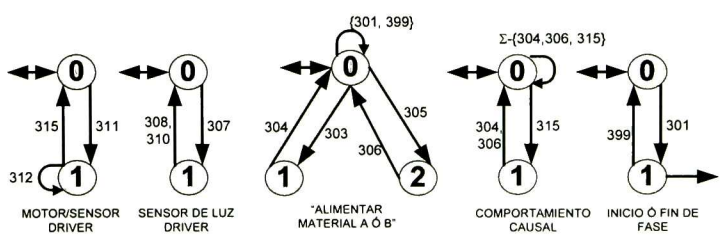


Figura 4.8: Componentes Elementales de Fase: Alimentar Materia Prima A o B.

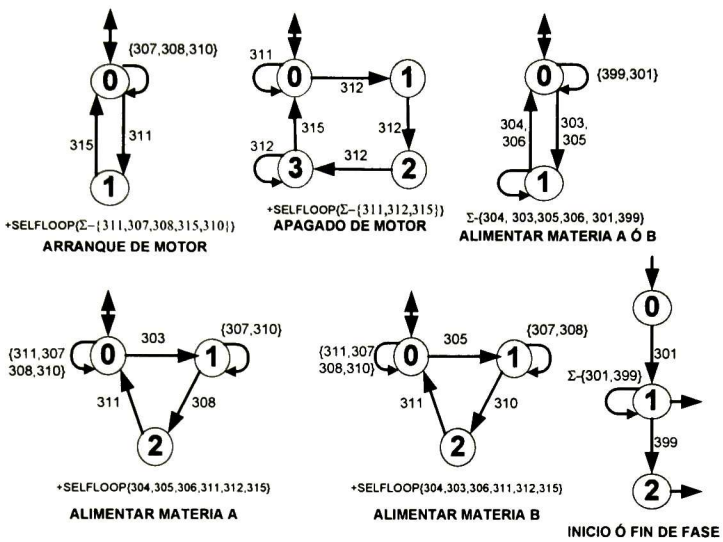


Figura 4.9: Especificaciones de Fase: Alimentar Materia Prima A o B.

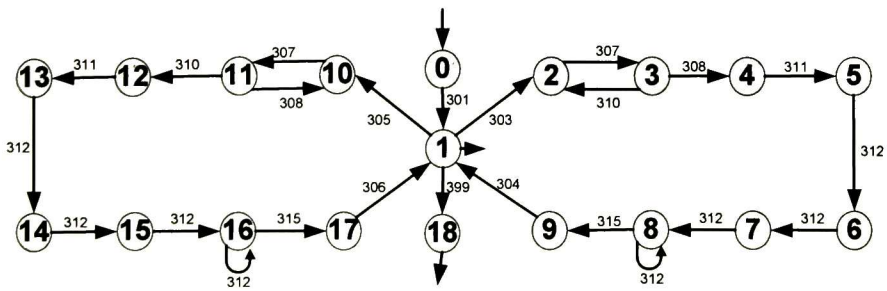


Figura 4.10: Supervisor de Fase: Alimentar Materia Prima A o B.

Tabla 4.5: Especificaciones de Fase: Alimentar Materia Prima A o B

Especificación	Descripción
Arranque de Motor	Permite encender el motor del alimentador de piezas hacia la derecha (Evento 311), una vez que se habilita el sensor de luz (Evento 307) y se ha detectado una pieza A o B (Eventos 308 o 310).
Apagado de Motor	Permite que después de haberse presionado el sensor de tacto (Evento 312) 3 veces o más, el motor del alimentador de piezas se detiene (Evento 315).
Alimentar Materia Prima A o B	Define el proceso de alimentar materia prima A o B (Eventos 303 y 304 o 305 y 306).
Alimentar Materia A	Define el proceso de alimentar materia prima A más detallado, es decir, una vez iniciado el procedo de alimentar materia prima A (Evento 303) y habilita el sensor de luz (Evento 307) y detectado una pieza A (Eventos 308), se enciende el motor del alimentador de piezas (Evento 311).
Alimentar Materia B	Define el proceso de alimentar materia prima B más detallado, es decir, una vez iniciado el procedo de alimentar materia prima B (Evento 305) y habilita el sensor de luz (Evento 307) y detectado una pieza B (Eventos 310), se enciende el motor del alimentador de piezas (Evento 311).
Inicio o Fin de Fase	Define la fase de alimentar materia prima (Eventos 301 y 399).

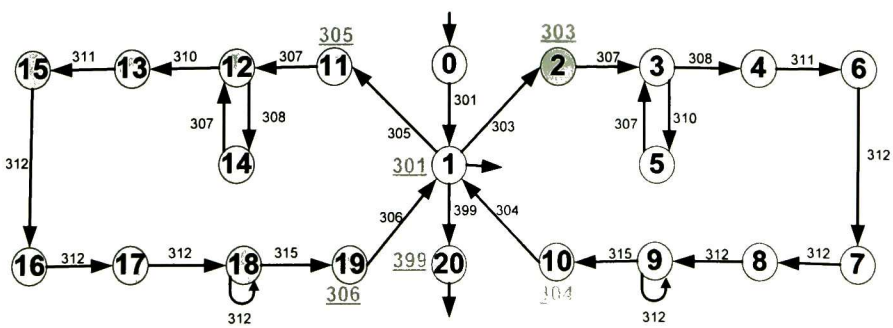


Figura 4.11: GA_{lo}.

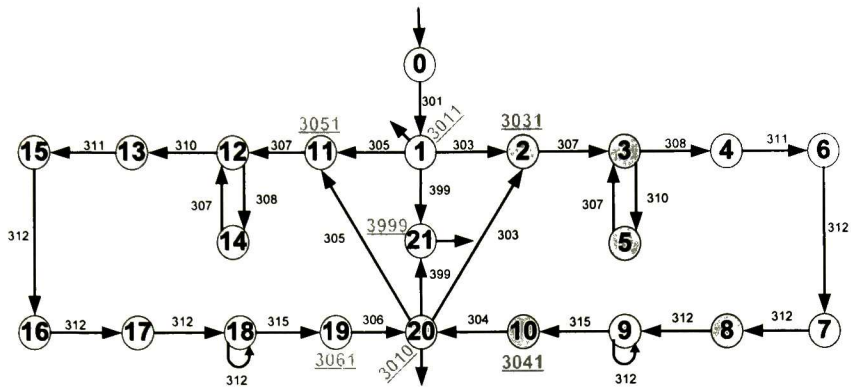


Figura 4.12: GA_{lo} Estrictamente Consistente en el Control de Salida.

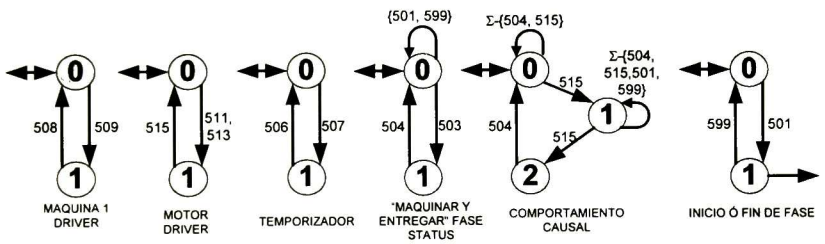


Figura 4.13: Componentes Elementales de Fase: Maquinado y Entrega.

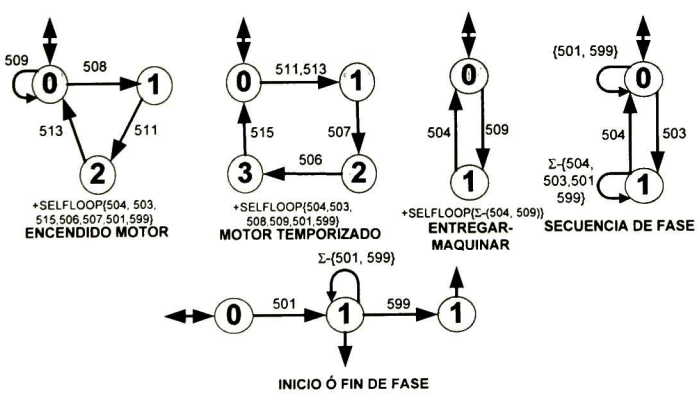


Figura 4.14: Especificaciones de Fase: Maquinado y Entrega.

Tabla 4.6: Especificaciones de Fase: Maquinado y Entrega

Especificación	Descripción
Encendido de Motor	Permite que después de haberse terminado el proceso de simulación de maquinado (Evento 508), se enciende el motor hacia la derecha e izquierda para sacar las piezas (Evento 511, 513).
Motor Temporizado	Enciende el motor hacia la derecha o izquierda para sacar las piezas (Eventos 511 o 513), posteriormente espera cierto tiempo dado por el temporizador (Eventos 507 y 506) para detener el motor (Evento 515).
Entrega-Maquinado	Define que una vez iniciado el proceso de simulación de maquinado (Evento 509) y realizado todo lo necesario, se termina el proceso de maquinado y entrega de producto (Evento 504).
Secuencia de Fase	Define el proceso de maquinado y entrega de producto (Eventos 503 y 504).
Inicio o Fin de Fase	Define la fase de maquinado y entrega de producto (Eventos 501 y 599).

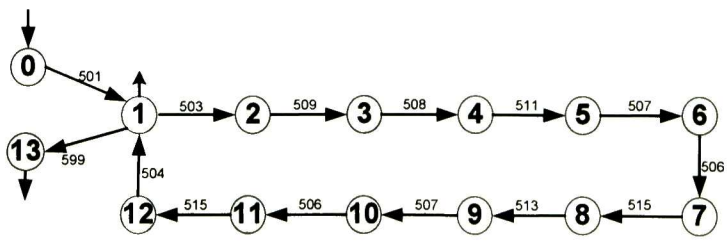


Figura 4.15: Supervisor de Fase: Maquinado y Entrega.

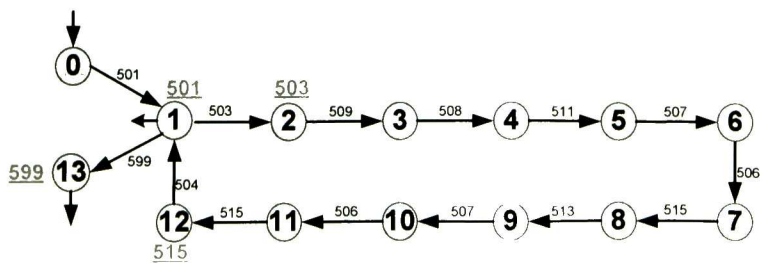


Figura 4.16: GM_{lo}.

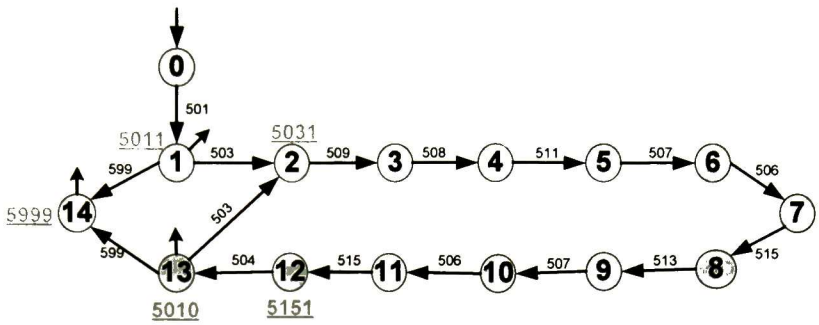


Figura 4.17: GM_{lo} Estrictamente Consistente en el Control de Salida.

Procedimiento Unitario : Despachar Materia Prima

En esta sección se obtienen GD_{hi} y CD_{hi} del módulo Despachar Materia Prima del nivel Procedimiento Unitario.

GD_{hi} se muestra en la figura 4.18, la cual es una abstracción (representación del comportamiento de los estados vocalizados) de GD_{lo} del nivel Fase, mostrada en la figura 4.7. GD_{hi} se obtiene mediante el procedimiento *Higen* de TCT.

En este nivel no se tiene ninguna especificación, por lo tanto GD_{hi} se toma como el supervisor (CD_{hi}) en este nivel.

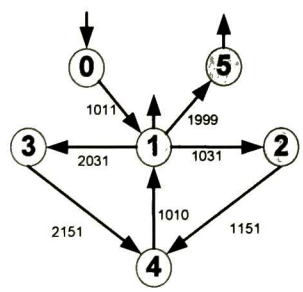


Figura 4.18: GD_{hi}.

Procedimiento Unitario : Procesar en Estación de Trabajo 1

En esta sección se obtienen $GPET1_{lo}, GPET1_{hi}$ y $CPET1_{hi}$ del módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

Los componentes elementales para sintetizar el supervisor de Procesar en Estación de Trabajo 1 del nivel de Procedimiento Unitario se muestran en la figura 4.19, los cuales son abstracciones (representaciones del comportamiento de los estados vocalizados) de GA_{lo} y GM_{lo} del nivel Fase, mostrados en las figuras 4.12 y 4.17. Los componentes elementales son obtenidos mediante el procedimiento *Higen* de TCT.

En la figura 4.20 se muestran las especificaciones requeridas en Procesar en Estación de Trabajo 1 del nivel de Procedimiento Unitario, y en la tabla 4.7 se describen estas especificaciones. De la misma manera en la figura 4.21 se muestra el supervisor de Procesar en Estación de Trabajo 1 del nivel de Procedimiento Unitario. Este supervisor es vocalizado. Como se muestra en la figura 4.22 y, de acuerdo a la metodología [20](capítulo 5), es $GPET1_{lo}$.

La forma de como se vocalizan los estados es de acuerdo a los requerimientos del diseñador, siempre y cuando se respete la metodología dada. La vocalización se realiza usando el procedimiento *Vocalice* de TCT.

Se verifica que $GPET1_{lo}$ sea estrictamente consistente en el control de salida, usando el procedimiento *Hiconsis* de TCT. Para más detalles consultar [20](Capítulo 5). Obteniendo así, a $GPET1_{lo}$ como se muestra en la figura 4.23.

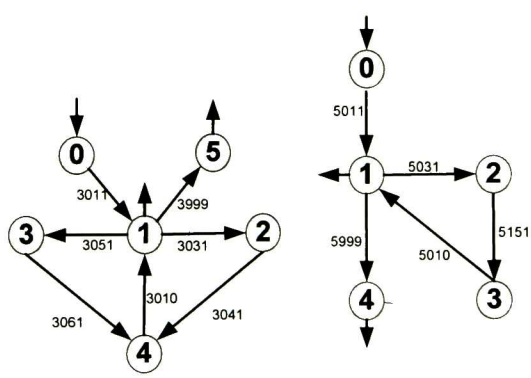


Figura 4.19: Componentes Elementales de Procedimiento Unitario : Procesar en Estación de Trabajo 1.

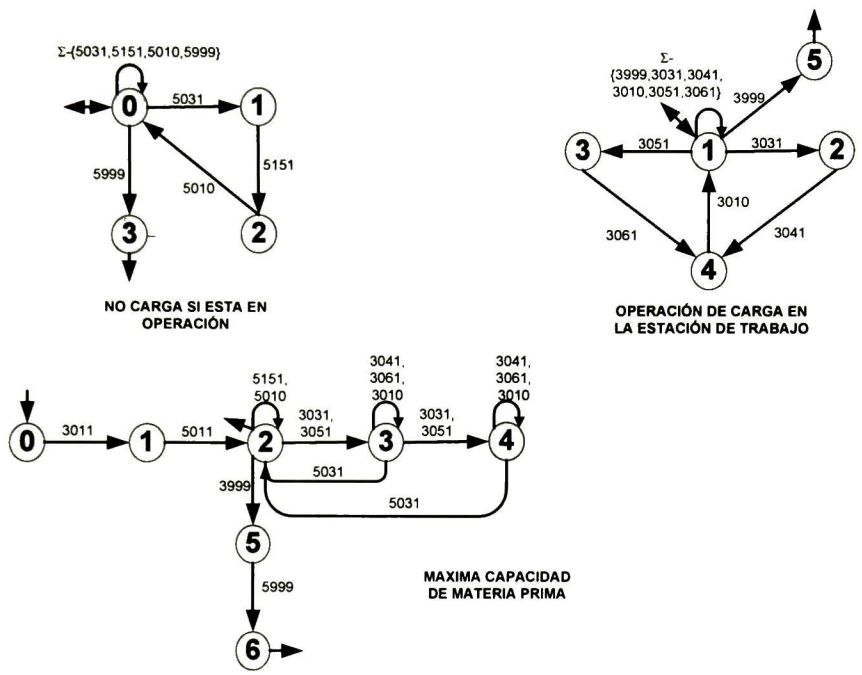


Figura 4.20: Especificaciones de Procedimiento Unitario: Procesar en Estación de Trabajo 1

Tabla 4.7: Especificaciones de Procedimiento Unitario: Procesar en Estación de Trabajo 1

Especificación	Descripción
No cargar si esta en operación	Define el proceso del módulo maquinado y entrega de producto del nivel fase (Eventos 5031, 5151, 5010 y 5999).
Operación de carga en la estación de trabajo	Define el proceso del módulo alimentar materia prima del nivel fase (Eventos (3051, 3061, 3010 o 3031, 3041, 3010) y 3999).
Maxima capacidad de materia prima	Describe la secuencia máxima de carga de materia prima para la estación de trabajo.

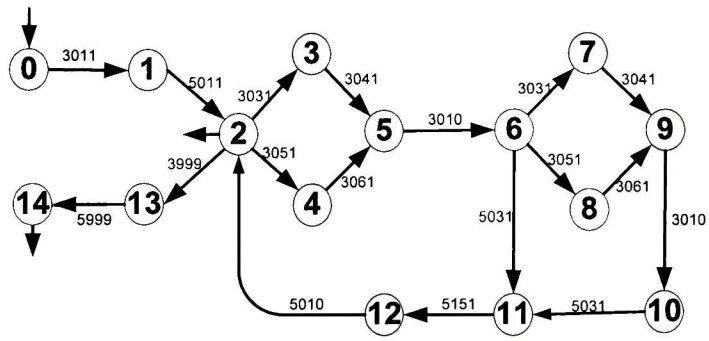


Figura 4.21: Supervisor de Procedimiento Unitario: Procesar en Estación de Trabajo 1.

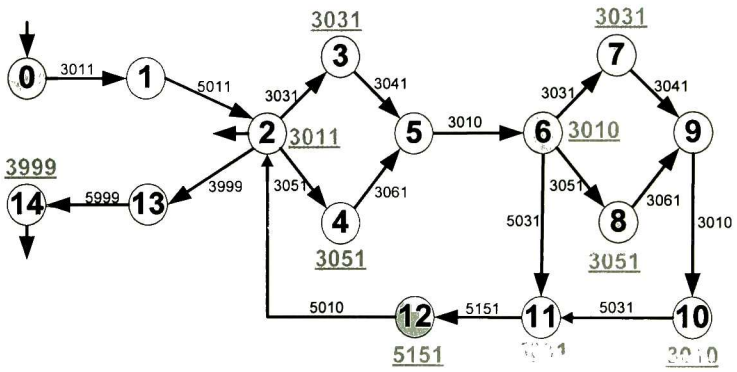


Figura 4.22: $GPET1_{lo}$.

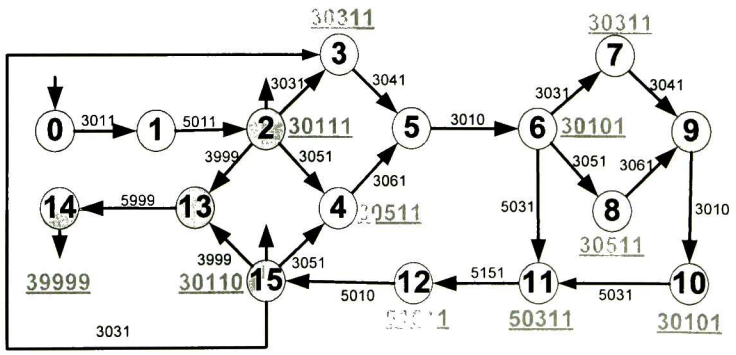


Figura 4.23: $GPET1_{lo}$ Estrictamente Consistente en el Control de Salida.

$GPET1_{hi}$ se muestra en la figura 4.24, la cual es una abstracción (representación del comportamiento de los estados vocalizados) de $GPET1_{lo}$, mostrada en la figura 4.23. $GPET1_{hi}$ se obtiene mediante el procedimiento *Higen* de TCT. En este caso se tiene que $CPET1_{hi}$ es igual a $GPET1_{hi}$, debido a que no se cuenta con ninguna especificación.

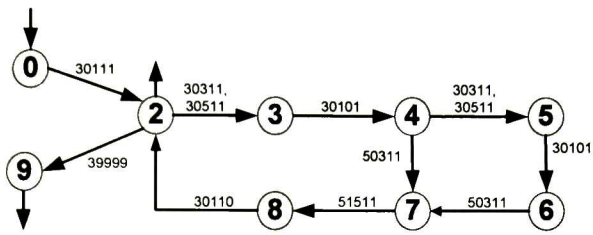


Figura 4.24: $GPET1_{hi}$.

4.1.3. Nivel Procedimiento

En este nivel de la estructura del caso de estudio se encuentran las recetas de producción de los módulo Despachar Materia Prima, Procesar en Estación de Trabajo 1 y Procesar en Estación de Trabajo 2 del nivel Procedimiento Unitario de acuerdo al estándar ISA88. En la figura 4.25 se presenta un ejemplo del procedimiento de la receta de control en los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

4.2. Estructura de Control Jerárquico II

En esta sección se aplica la metodología de control de supervisión jerárquico basada en interfaz citado en [10]. Esta estructura de control jerárquico es mostrada en la figura 4.26.

La estructura básica de control jerárquico está formada por dos niveles (superior e inferior) y una interfaz que realiza la comunicación entre ellos. En esta estructura existe una relación de maestro-esclavo, ya que el sistema del nivel superior envía un comando para un sistema en particular del nivel inferior, el cual entonces realiza la tarea indicada y retorna un respuesta. La comunicación entre los sistemas de los niveles se da de manera serial, es decir, una petición del nivel superior es seguida de una respuesta del nivel inferior antes de la siguiente petición. Este tipo de iteración es realizada por una interfaz para mediar la comunicación entre los dos subsistemas. Todo los componentes del sistema incluyendo la interfaz son modelados como

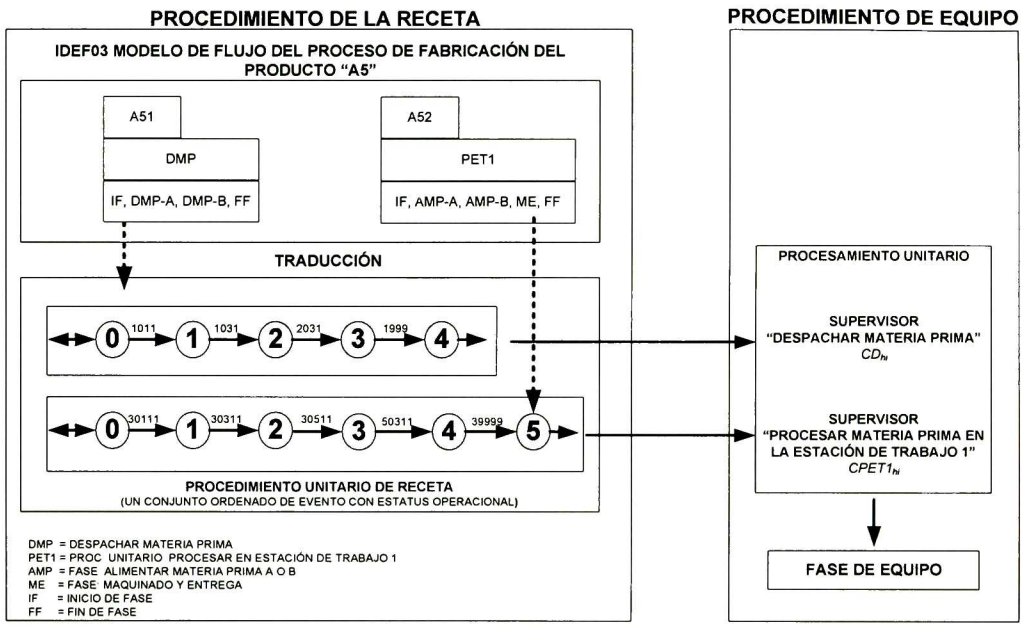


Figura 4.25: Procedimiento de la Receta de Control Jerárquico de Dos-Niveles.

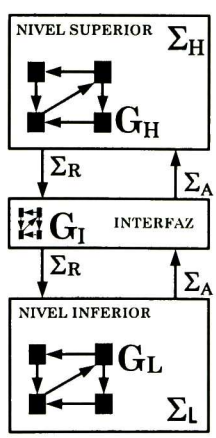


Figura 4.26: Control Jerárquico Basado en Interfaz.

autómatas. Para la estructura básica, nuestro plano del sistema es $G = G_H \| G_L \| G_I$, donde G_H está definida sobre $\Sigma_H \cup \Sigma_R \cup \Sigma_A$, G_L está definida sobre $\Sigma_L \cup \Sigma_R \cup \Sigma_A$ y G_I está definida sobre $\Sigma_R \cup \Sigma_A$. Σ_H es el conjunto de eventos del nivel superior, Σ_L es el conjunto de eventos del nivel inferior, Σ_R es el conjunto de eventos de petición y Σ_A es el conjunto de eventos de respuesta.

Para restringir la serialización de petición y respuesta, se restringe la interfaz a la subclase de interfaces de pareja de comandos (Definición 2 de [10]), esta definición tiene dos condiciones: $L(G_I) \subseteq (\overline{\Sigma_R.\Sigma_A})^*$ y $Lm(G_I) = (\Sigma_R.\Sigma_A)^* \cap L(G_I)$.

Cuando el sistema es n-ésimo grado, es decir, solo existe un subsistema en el nivel superior que interactua con $n \geq 1$ subsistemas independientes del nivel inferior, la comunicación es realizada mediante una interfaz correspondiente a cada subsistema del nivel inferior. El plano del sistema es $G = G_H \| G_{L_1} \| G_{I_1} \ldots \| G_{L_n} \| G_{I_n}$, donde G_H está definida sobre $\Sigma_H \bigcup_{j=1\ldots n} [\Sigma_{R_j} \cup \Sigma_{A_j}]$, G_{L_j} está definida sobre $[\Sigma_{L_j} \cup \Sigma_{R_j} \cup \Sigma_{A_j}]$ y G_{I_j} está definida sobre $[\Sigma_{R_j} \cup \Sigma_{A_j}]$. En la Definición 3 de [10] se presentan unas propiedades que el sistema debe de satisfacer para asegurarse que interactua correctamente con las interfaces, es decir, si la interfaz es consistente. Con lo mencionado anteriormente respecto a las interfaces en las Definiciones 2 y 3, solo se mencionan las condiciones que deben de cumplir, pero nunca menciona como realizar una interfaz; aparte para las propiedades 5 y 6 de la definición 3 no existe un programa disponible para verificar si se cumplen. Por lo tanto, tenemos que la construcción de la interfaz es a prueba y error, y la verificación de algunas condiciones es manual.

Si los subsistemas y sus interfaces satisfacen las propiedades dadas en la Definición 3 de [10] y satisfacen las condiciones presentadas en la Definición 4 de [10] se cumple el no bloqueo en los niveles, permitiendo así con el Teorema 1 de [10] se cumpla el no bloqueo global.

Los subsistemas G_H y G_{L_j} están construidos como sigue: $G_H = {G_H}^P \| S_H$ y $G_{L_j} = {G_{L_j}}^P \| S_{L_j}$. En la Definición 5 de [10] se presentan condiciones que debe cumplir el sistema compuesto de ${G_H}^P, {G_{L_1}}^P \ldots {G_{L_n}}^P, S_H, S_{L_1} \ldots S_{L_n}$ y $G_{I_1} \ldots G_{I_n}$ para que sea controlable en los niveles, permitiendo así con el Teorema 2 de [10] se cumpla la controlabilidad global. Con la Definición 5 de [10] se muestra que el supervisor del nivel superior es controlable para la planta del nivel superior combinada con toda las interfaces, es decir, ${G_H}^P$ es el producto asíncrono de los componentes del nivel superior y las interfaces; y cada supervisor del nivel inferior combinado con su subsistema de interfaz es controlable para el subsistema de la planta del nivel inferior, es decir, para sintetizar el supervisor S_{L_n} se requiere las especificaciones y su interfaz del subsistema del nivel inferior. Para más detalles consulte [10].

La estructura de control jerárquico del caso de estudio es como se muestra en la figura 4.27, la cual está apegada al modelo híbrido del caso de estudio obtenida en el capítulo 3,

sección 3.9. La estructura está formada por los siguientes componentes: GD_L es la PLANTA que representa el comportamiento del sistema, GD_I es la interfaz y SD_L es el supervisor del módulo Despachar Materia Prima A y B en el nivel Fase. GD_L y SD_L están definidas sobre $[\Sigma_{L,D} \cup \Sigma_{R,D} \cup \Sigma_{A,D}]$ y GD_I está definida sobre $[\Sigma_{R,D} \cup \Sigma_{A,D}]$, donde $\Sigma_{L,D}$ es el conjunto de eventos del módulo Despachar Materia Prima A y B en el nivel Fase, $\Sigma_{R,D} = 103, 203$ es el conjunto de eventos de petición del módulo Despachar Materia Prima A y B y $\Sigma_{A,D} = 104, 204$ es el conjunto de eventos de respuesta del módulo Despachar Materia Prima A y B. GA_L es la PLANTA que representa el comportamiento del sistema, GA_I es la interfaz y SA_L es el supervisor del módulo Alimentar Materia Prima A y B en el nivel Fase. GA_L y SA_L están definidas sobre $[\Sigma_{L,A} \cup \Sigma_{R,A} \cup \Sigma_{A,A}]$ y GA_I está definida sobre $[\Sigma_{R,A} \cup \Sigma_{A,A}]$, donde $\Sigma_{L,A}$ es el conjunto de eventos del módulo Alimentar Materia Prima A y B en el nivel Fase, $\Sigma_{R,A} = 303, 305$ es el conjunto de eventos de petición del módulo Alimentar Materia Prima A y B y $\Sigma_{A,A} = 304, 306$ es el conjunto de eventos de respuesta del módulo Alimentar Materia Prima A y B. GM_L es la PLANTA que representa el comportamiento del sistema, GM_I es la interfaz y SM_L es el supervisor del módulo Maquinar y Entregar en el nivel Fase. GM_L y SM_L están definidas sobre $[\Sigma_{L,M} \cup \Sigma_{R,M} \cup \Sigma_{A,M}]$ y GM_I está definida sobre $[\Sigma_{R,M} \cup \Sigma_{A,M}]$, donde $\Sigma_{L,M}$ es el conjunto de eventos del módulo Maquinar y Entregar en el nivel Fase, $\Sigma_{R,M} = 503$ es el conjunto de eventos de petición del módulo Maquinar y Entregar y $\Sigma_{A,M} = 504$ es el conjunto de eventos de respuesta del módulo Maquinar y Entregar. GD_H es la PLANTA que representa el comportamiento del sistema y SD_H es el supervisor del módulo Despachar Materia Prima del nivel Procedimiento Unitario. GD_H y SD_H están definidas sobre $[\Sigma_{H,D} \cup \Sigma_{R,D} \cup \Sigma_{A,D}]$, donde $\Sigma_{H,D}$ es el conjunto de eventos del módulo Despachar Materia Prima del nivel Procedimiento Unitario. $GPET1_H$ es la PLANTA que representa el comportamiento del sistema y $SPET1_{hi}$ es el supervisor del módulo Procesar en la Estación de Trabajo 1 en el nivel Procedimiento Unitario. $GPET1_H$ y $SPET1_H$ están definidas sobre $[\Sigma_{H,PET1} \cup \Sigma_{R,A} \cup \Sigma_{A,A} \cup \Sigma_{R,M} \cup \Sigma_{A,M}]$, donde $\Sigma_{H,PET1}$ es el conjunto de eventos del módulo Procesar en la Estación de Trabajo 1 del nivel Procedimiento Unitario.

En la sección 4.2.1 se obtienen GD_L, SD_L, GA_L, SA_L, GM_L, SM_L, GA_I, GM_I y GD_I del nivel Fase. Se obtienen GD_H, SD_H, $GPET1_H$ y $SPET1_H$ del nivel Procedimiento Unitario de la estructura del caso de estudio en la sección 4.2.2. Finalmente, en la sección 4.2.3 se da un ejemplo de las recetas de producción de los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

4.2.1. Síntesis de Supervisores de Fase

En esta sección se obtienen los componentes GD_L, SD_L y GD_I del módulo Despachar Materia Prima A o B; GA_L, SA_L y GA_I del módulo Alimentar Materia Prima A o B; GM_L,

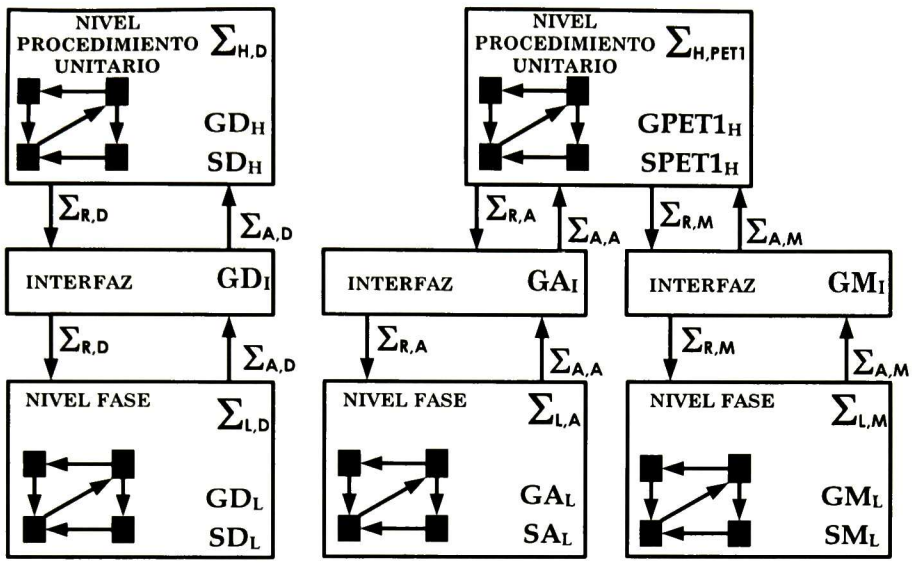


Figura 4.27: Control Jerárquico Basado en Interfaz del Caso de Estudio.

SM_L y GM_I del módulo Maquinar y Entregar del nivel Fase de la arquitectura del caso de estudio.

En las figuras 4.28, 4.31 y 4.34 se muestran los componentes elementales de cada módulo del nivel Fase que son : modelos de los equipos físicos que intervienen y cierto comportamiento causal requerido. GD_L, GA_L y GM_L son el producto asíncrono de sus componentes elementales correspondiente a cada módulo.

En las figuras 4.29, 4.32 y 4.35 se muestran las especificaciones requeridas en cada módulo del nivel Fase, donde una de las especificaciones es la interfaz correspondiente a cada módulo del nivel Fase; y en las tablas 4.4, 4.5 y 4.6 se describen algunas de las especificaciones de cada módulo del nivel Fase. De la misma manera, en las figuras 4.30, 4.33 y 4.36 se muestran respectivamente los supervisores SD_L, SA_L y SM_L del nivel Fase.

4.2.2. Síntesis de Supervisores de Procedimiento Unitario

En esta sección se obtienen los componentes GD_H y SD_H del módulo Despachar Materia Prima; también se obtienen $GPET1_H$ y $SPET1_H$ del módulo Procesar en Estación de Trabajo 1 del nivel procedimiento unitario de la estructura del caso de estudio.

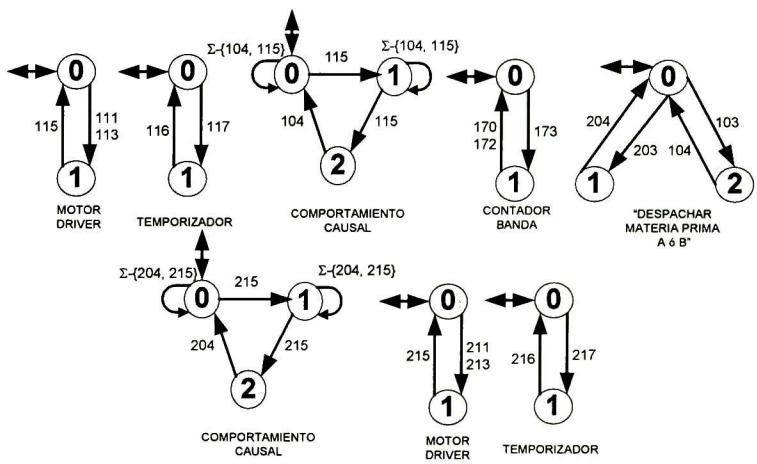


Figura 4.28: Componentes Elementales de Fase: Despachar Materia Prima A o B.

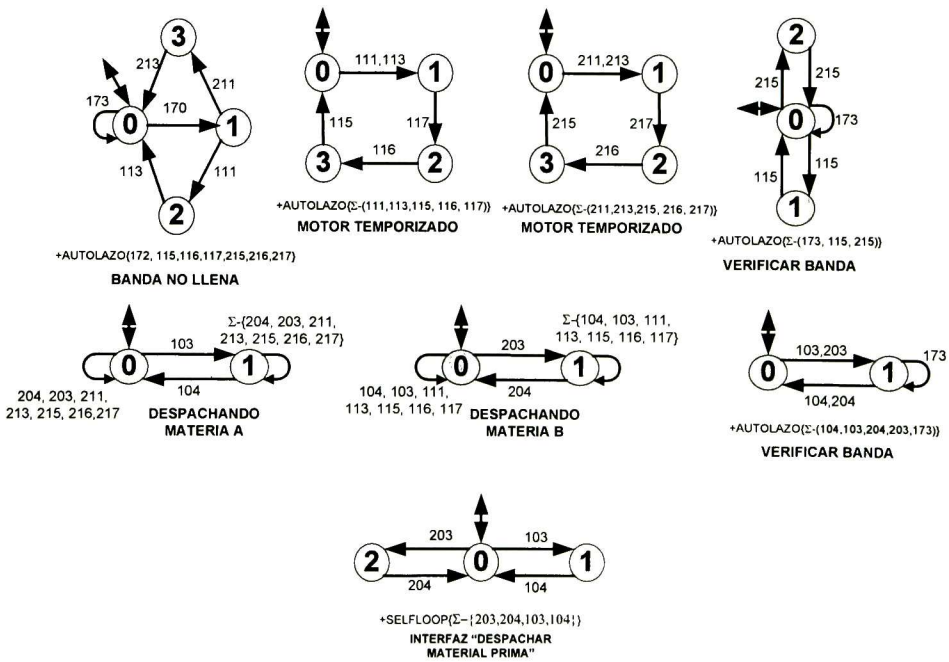


Figura 4.29: Especificaciones de Fase: Despachar Materia Prima A o B.

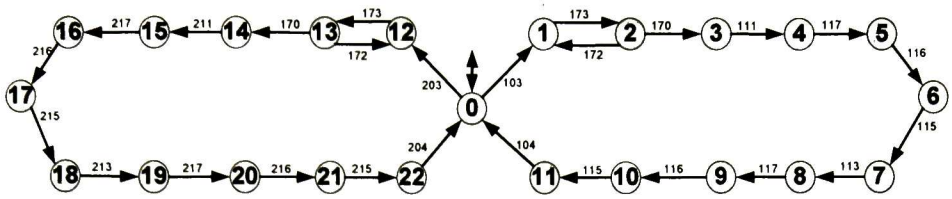


Figura 4.30: Supervisor de Fase: Despachar Materia Prima A o B (SD_L).

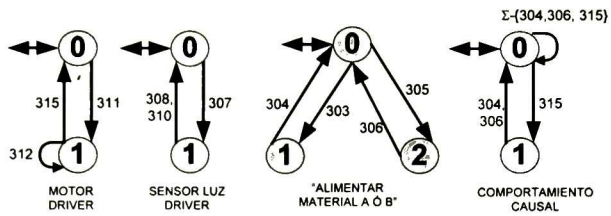


Figura 4.31: Componentes Elementales de Fase: Alimentar Materia Prima A o B.

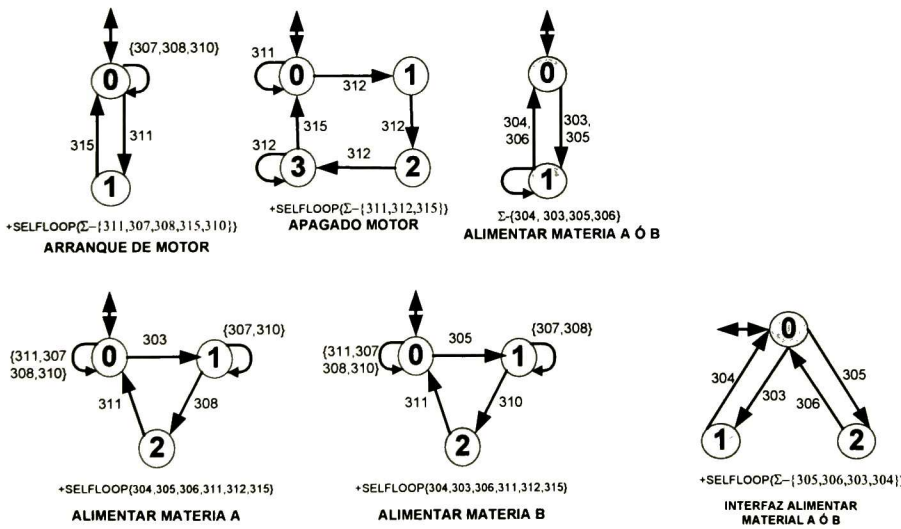


Figura 4.32: Especificaciones de Fase: Alimentar Materia Prima A o B.

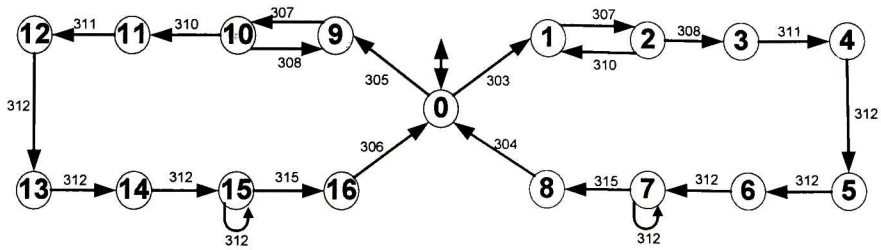


Figura 4.33: Supervisor de Fase: Alimentar Materia Prima A o B (SA_L).

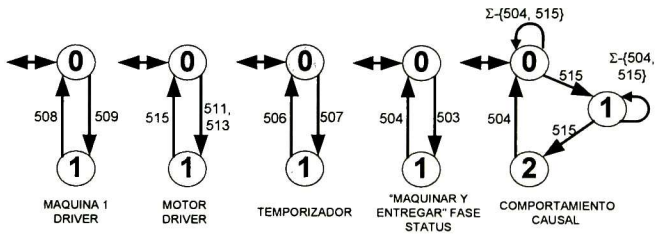


Figura 4.34: Componentes Elementales de Fase: Maquinado y Entrega.

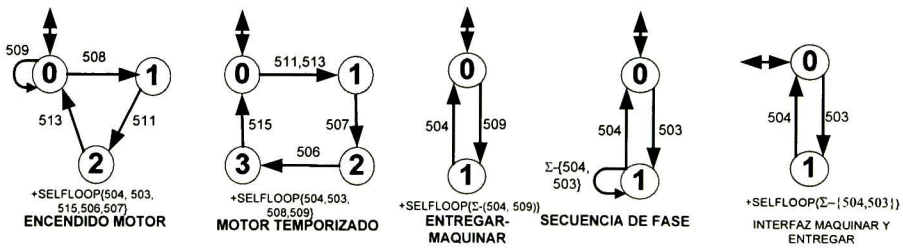


Figura 4.35: Especificaciones de Fase: Maquinado y Entrega.

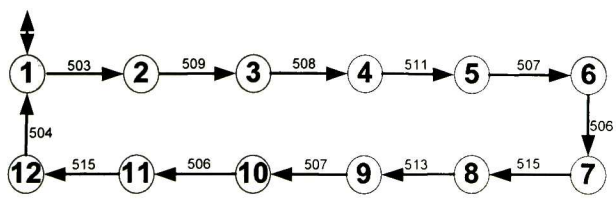


Figura 4.36: Supervisor de Fase: Maquinado y Entrega (SM_L).

En las figuras 4.37 y 4.38, se muestran los componentes elementales (interfaces) de cada módulo del Procedimiento Unitario. GD_H y $GPET1_H$ son el producto asíncrono de sus componentes elementales correspondiente a cada módulo.

En la figura 4.39 se muestran las especificaciones requeridas del módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario, y en la tabla 4.7 se describen estas especificaciones. En la figura 4.40 se muestra el supervisor $SPET1_H$ del módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

En el módulo Despachar Materia Prima del nivel Procedimiento Unitario no se tiene ninguna especificación, por lo tanto GD_H se toma como el supervisor SD_H de este módulo en este nivel.

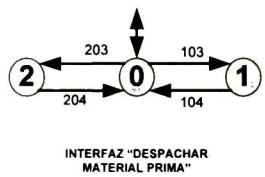


Figura 4.37: Componentes Elementales de Procedimiento Unitario: Despachar Materia Prima.

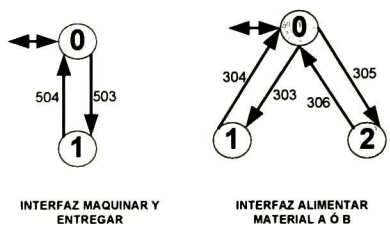


Figura 4.38: Componentes Elementales de Procedimiento Unitario: Procesar en Estación de Trabajo 1.

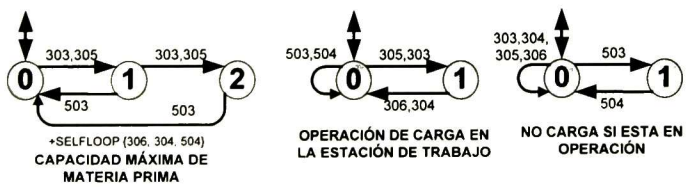


Figura 4.39: Especificaciones de Procedimiento Unitario: Procesar en Estación de Trabajo 1.

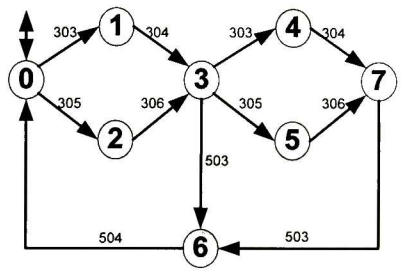


Figura 4.40: Supervisor de Procedimiento Unitario: Procesar en Estación de Trabajo 1 ($SPET1_H$).

4.2.3. Nivel Procedimiento

En este nivel de la estructura del caso de estudio se encuentran las recetas de producción de los módulo Despachar Materia Prima, Procesar en Estación de Trabajo 1 y Procesar en Estación de Trabajo 2 del nivel Procedimiento Unitario de acuerdo al estándar ISA88. En la figura 4.41 se presenta un ejemplo del procedimiento de la receta de control en los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

4.3. Estructura de Control Jerárquico III

En esta sección se presenta el control de supervisión jerárquico y modular basado en la proyección natural del caso de estudio, propuesto en [17]. Esta estructura de control jerárquico es mostrada en la figura 4.42, la cual está apegada al modelo híbrido del caso de estudio obtenida en el capítulo 3, sección 3.9.

El concepto de modularidad del caso de estudio del nivel fase fue tomado de [20, 21], para así tener módulos con alfabetos disjuntos controlados por un supervisor. En la construcción jerárquica de la estructura se utiliza la proyección natural, para que el comportamiento del sistema de los módulos del nivel Procedimiento Unitario sean las proyecciones naturales de los eventos con estatus operacional de los supervisores del nivel Fase que pertenecen a cada módulo en el nivel Procedimiento Unitario, y de esta manera generar una semántica clara y consistente entre los niveles, permitiendo así la comunicación entre ellos. Sin embargo, una justificación formal de la controlabilidad global no fue establecida. En este caso la modularidad de la estructura con alfabetos disjuntos controlados por un supervisor, y la manera de como se estableció la jerarquía de la estructura utilizando las proyecciones naturales de los eventos con estatus operacional de los supervisores del nivel Fase, es suficiente para garantizar

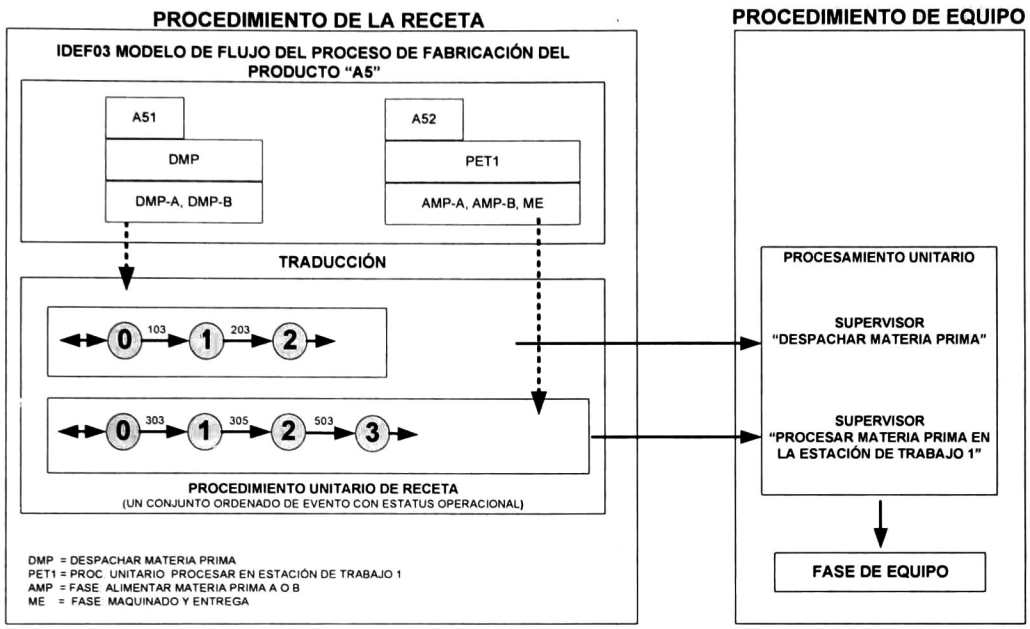


Figura 4.41: Procedimiento de la Receta de Control Jerárquico Basado en Interfaz

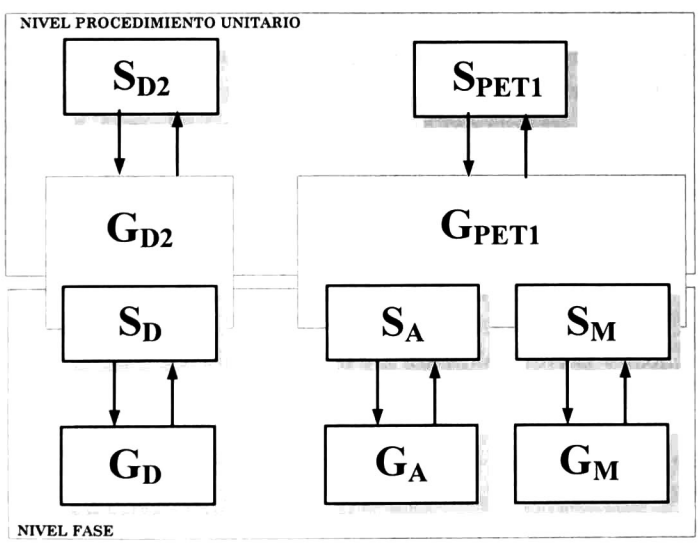


Figura 4.42: Control Jerárquico Basado en Proyecciones del Caso de Estudio.

la controlabilidad global.

La estructura está formada por los siguientes componentes: GD es la PLANTA que representa el comportamiento del sistema, y SD es el supervisor del módulo Despachar Materia Prima A y B en el nivel Fase. GA es la PLANTA que representa el comportamiento del sistema, y SA es el supervisor del módulo Alimentar Materia Prima A y B en el nivel Fase. GM es la PLANTA que representa el comportamiento del sistema, y SM es el supervisor del módulo Maquinar y Entregar en el nivel Fase. $GD2$ es la proyección natural de los eventos con estatus operacional del supervisor SD del nivel Fase, y $SD2$ es el supervisor del módulo Despachar Materia Prima en el nivel Procedimiento Unitario. $GPET1$ es producto asíncrono de las proyecciones naturales de los eventos con estatus operacional de los supervisores SA y SM del nivel Fase, y $SPET1$ es el supervisor del módulo Procesar en la Estación de Trabajo 1 en el nivel Procedimiento Unitario.

En la sección 4.3.1 se obtienen G_D, S_D, G_A, S_A, G_M y S_M del nivel Fase. En la sección 4.3.2 se obtienen G_{D2}, S_{D2}, G_{PET1} y S_{PET1} del nivel Procedimiento Unitario de la estructura del caso de estudio. Finalmente, en la sección (4.3.3) se da un ejemplo de las recetas de producción de los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

4.3.1. Síntesis de Supervisores de Fase

En esta sección se obtienen los componentes G_D y S_D del módulo Despachar Materia Prima A o B; los componentes G_A y S_A del módulo Alimentar Materia Prima A o B; y los componentes G_M y S_M del módulo Maquinar y Entregar del nivel Fase de la estructura del caso de estudio. Los módulos del nivel Fase están construidos con alfabetos disjuntos, por lo que la construcción del supervisor de cada modulo se realiza de una manera clara y estándar.

En las figuras 4.28, 4.31 y 4.34 se muestran los componentes elementales de cada módulo del nivel Fase, que son : los modelos de los equipos físicos que intervienen y cierto comportamiento causal requerido. Los componentes G_D, G_A y G_M son el producto asíncrono de sus componentes elementales correspondiente a cada módulo.

En la figura 4.29 se muestran las especificaciones (Banda no llena, Motor temporizado, Motor temporizado, Verificar banda, Verificar banda, Despachando materia A, Despachando materia B) requeridas en el módulo Despachar Materia Prima A y B en el nivel Fase, y en la tabla 4.4 se describen las especificaciones.

En la figura 4.32 se muestran las especificaciones (Arranque de motor, Apagado motor, Alimentar Materia A o B, Alimentar materia A, Alimentar materia B) requeridas en el

módulo Alimentar Materia Prima A y B en el nivel Fase, y en la tabla 4.5 se describen las especificaciones.

En la figura 4.35 se muestran las especificaciones (Encendido motor, Motor temporizado, Entregar-maquinar, Secuencia de fase) requeridas en el módulo Maquinar y Entregar en el nivel Fase, y en la tabla 4.6 se describen las especificaciones. De la misma manera, en las figuras 4.30, 4.33 y 4.36 se muestran los supervisores S_D, S_A y S_M del nivel Fase.

4.3.2. Síntesis de Supervisores de Procedimiento Unitario

En esta sección se obtienen los componentes G_{D2} y S_{D2} del módulo Despachar Materia Prima. También se obtienen G_{PET1} y S_{PET1} del módulo Procesar en Estación de Trabajo 1 del nivel procedimiento unitario de la arquitectura del caso de estudio.

En las figuras 4.43 y 4.44, se muestran los componentes elementales de cada módulo del Procedimiento Unitario, donde los componentes elementales de cada módulo son las proyecciones naturales de los eventos con estatus operacional de los supervisores del nivel Fase que pertenecen a cada módulo en el nivel Procedimiento Unitario. Los componentes G_{D2} y G_{PET1} son el producto asíncrono de sus componentes elementales correspondiente a cada módulo.

En la figura 4.39 se muestran las especificaciones (Capacidad máxima de materia prima, Operación de carga en la estación de trabajo, No carga si esta en operación) requeridas en el módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario, y en la tabla 4.7 se describen las especificaciones. En la figura 4.40 se muestra el supervisor S_{PET1} del módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

En el módulo Despachar Materia Prima del nivel Procedimiento Unitario no se tiene ninguna especificación, por lo que se toma G_{D2} como el supervisor S_{D2} de este módulo en este nivel.

La controlabilidad global se garantizó por: La modularidad de la estructura con alfabetos disjuntos del nivel fase controlados por un supervisor, y la manera de como se construyó la jerarquía de la estructura aplicando la proyección natural de los eventos con estatus operacional de los supervisores del nivel Fase.

4.3.3. Nivel Procedimiento

En este nivel de la estructura del caso de estudio se encuentran las recetas de producción de los módulo Despachar Materia Prima, Procesar en Estación de Trabajo 1 y Procesar en

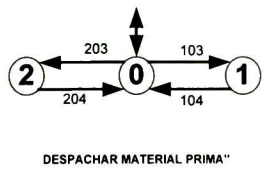


Figura 4.43: Componentes Elementales de Procedimiento Unitario: Despachar Materia Prima.

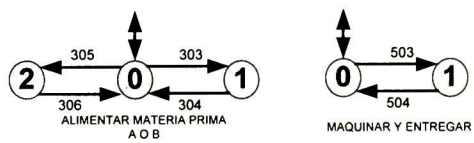


Figura 4.44: Componentes Elementales de Procedimiento Unitario: Procesar en Estación de Trabajo 1.

Estación de Trabajo 2 del nivel Procedimiento Unitario de acuerdo al estándar ISA88. En la figura 4.41 se presenta un ejemplo del procedimiento de la receta de control en los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

4.4. Estructura de Control Jerárquico IV

En esta sección se aplican los conceptos de sistema confiable, lenguaje realizable presentados en el capítulo 2, sección 2.2 (definidos por [9]) y de proyección natural en la construcción del control jerárquico y modular del caso de estudio.

La estructura de control jerárquico del caso de estudio es igual a la mostrada en la figura 4.42 de la sección 4.3.

La diferencia de esta estructura radica en el concepto de construcción, ya que la meta aquí es construir un control de supervisión jerárquico y modular que garantice la fabricación de un producto de una manera formal, es decir, construir un control modular que sea un sistema confiable en el nivel Fase, y que el control jerárquico en el nivel Procedimiento Unitario también sea un sistema confiable por construcción. para que así, la receta de producción sea realizable en el control jerárquico y modular, y de esta manera, se garantice la controlabilidad global.

En la sección 4.4.1 se obtienen: G_D, S_D, G_A, S_A, G_M y S_M del nivel Fase; en la sección 4.4.2 se obtienen G_{D2}, S_{D2}, G_{PET1} y S_{PET1} del nivel Procedimiento Unitario de la estructura del caso de estudio. Finalmente, en la sección 4.4.3 se da un ejemplo de las recetas de producción de los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

4.4.1. Síntesis de Supervisores de Fase

En esta sección se presentan en las figuras 4.30, 4.33 y 4.36 los supervisores S_D, S_A y S_M de los módulos del nivel Fase.

Estos supervisores se sintetizaron de tal forma que fueran sistemas confiables con las siguientes características: El estado q_0 es un estado de decisión y para toda $\sigma \in \Sigma_{Imp}$ y $\delta(q_0, \sigma)$ está definida, existe una palabra $\omega = \sigma s \tau$ tal que $\delta(q_0, \omega) = q_0$, donde $\tau \in \Sigma_{Inf}$ y $s \in \Sigma^*$

La síntesis de cada supervisor se realizó en la sección 4.3.1, del capítulo 4.3.

4.4.2. Síntesis de Supervisores de Procedimiento Unitario

En esta sección se obtienen los componentes G_{D2} y S_{D2} del módulo Despachar Materia Prima. También se obtienen G_{PET1} y S_{PET1} del módulo Procesar en Estación de Trabajo 1 del nivel procedimiento unitario de la estructura del caso de estudio.

En las figuras 4.43 y 4.44 se muestran los componentes elementales de cada módulo del nivel Procedimiento Unitario.

La proyección natural utilizada es de la siguiente manera $P : \Sigma^* \longrightarrow \Sigma_p^*$ donde Σ_p es igual a $\sigma \in \Sigma$ tal que $\sigma \in \Sigma_{Imp}$ y $\delta(q_0, \sigma)$ es definida, o $\tau \in \Sigma_{Inf}$ y $\delta(q_0, \sigma) = q_0$.

Estos componentes elementales son sistema confiable por que cumplen con el teorema 4.1 establecido en este trabajo de investigación, ya que este teorema establece que la proyección natural de un sistema confiable, es también un sistema confiable; y como todos los supervisores del nivel Fase son sistemas confiables entonces la proyección natural de estos es también un sistema confiable en este caso los componentes elementales.

Teorema 4.1. *Sea* $G := (Q, \Sigma, \delta, q_0, Q_m)$ *un generador con las siguientes características:*

1. *El estado* q_0 *es un estado de decisión.*
2. *Para toda* $\sigma \in \Sigma_{Imp}$ *con* $\delta(q_0, \sigma)$ *definida, existe una palabra* $\omega = \sigma s \tau$ *con* $\tau \in \Sigma_{Inf}$ *y* $s \in \Sigma^*$ *tal que* $\delta(q_0, \omega) = q_0$.

Sea $P : \Sigma \longrightarrow \Sigma_P$ la proyección natural, que se define por la regla $P(\sigma) = \sigma$ si $\sigma \in \Sigma_{Imp}$ y $\delta(q_0, \sigma)$ está definida, o $\sigma \in \Sigma_{Inf}$, $\delta(q_n, \sigma) = q_0$ está definida y $q_n \in Q$ $P(\sigma) = \epsilon$ de otro modo.

P es catenativa, de forma tal que para cualquier palabra $\omega = s_1 s_2 \dots s_n$, $P(\omega) = P(s_1)P(s_2)\dots P(s_n)$.

Si G es un sistema confiable, entonces la máquina generadora $(G' = (Q', \Sigma' \ \delta' \ q'_0, Q'_m))$ de $P(G)$ es un sistema confiable.

Demostración. *Supongo que G es un sistema confiable y $P(G)$ tiene las características enunciadas arriba.*

Pruebo que G' es un sistema confiable.

Sea $\omega \in L(G)$ donde $\omega = \sigma s \tau$, con $P(w) = \sigma s' \tau$, donde $s' \in \Sigma_p^*$.

Entonces tenemos que s' puede ser de dos casos.

Caso 1. *Si $s' = \epsilon$, entonces $\sigma \in \Sigma_{Imp}$ es una transición de salida solamente del estado q_0 en G. Por lo tanto σ es una salida solamente del estado q'_0 en (G').*

Caso 2. *Si $s' \neq \epsilon$.*

A) *Si $s' \in \Sigma^*_{Inf}$, entonces $\sigma \in \Sigma_{Imp}$ es una transición de salida solamente del estado q_0 en G. Por lo tanto σ es una salida solamente del estado q'_0 en (G').*

B) *Si s' contiene al menos una $\sigma \in \Sigma_{Imp}$, donde σ es una transición de salida del estado q_0 en G entonces, por hipótesis $\delta(q_1, s) = q_n$ y a si mismo $\tau \in \Sigma_{Inf}$ y $\delta(q_n, \tau) = q_0$ esta definida en G. Entonces existe un estado de decisión q_0 logrables desde q en G. Por lo tanto σ es una transición de salida del estado q'_0 en G' $\delta'(q'_1, s') = q'_n$ y así mismo $\tau \in \Sigma_{Inf}$ y $\delta'(q'_n, \tau) = q'_0$ esta definida en G' entonces existe un estado q'_0 logrables desde q' en G'*

Como $P(w)$ conserva el estado de decisión q'_0 y este es alcanzable por toda $q' \in Q'$ Por lo tanto G' es un sistema confiable.

G_{D2} y G_{PET1} son productos asíncronos de sus respectivos componentes elementales de cada módulo. Como el producto asíncrono de dos sistemas es un sistema confiable [9], se tiene que G_{D2} y G_{PET1} son sistemas confiables.

En la figura 4.39 se muestran las especificaciones (Capacidad máxima de materia prima, Operación de carga en la estación de trabajo, No carga si esta en operación) requeridas en el módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario, y en la tabla 4.7 se describen las especificaciones. En la figura 4.40 se muestra el supervisor S_{PET1} del módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

En el módulo Despachar Materia Prima del nivel Procedimiento Unitario no se tiene ninguna especificación; por lo tanto G_{D2} se toma como el supervisor S_{D2} de este módulo en este nivel.

4.4.3. Nivel Procedimiento

En este nivel de la estructura del caso de estudio se encuentran las recetas de producción de los módulo Despachar Materia Prima, Procesar en Estación de Trabajo 1 y Procesar en Estación de Trabajo 2 del nivel Procedimiento Unitario de acuerdo al estándar ISA88. En la figura 4.41 se presenta un ejemplo del procedimiento de la receta de control en los módulos Despachar Materia Prima y Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario.

La receta de cada módulo del nivel Procedimiento es un lenguaje realizable en los supervisores del módulo correspondiente del nivel Procedimiento Unitario, y también es realizable en el producto asíncrono de los supervisores de cada nivel Fase que pertenecen al modulo correspondiente del nivel Procedimiento Unitario de la receta. Por ejemplo, la receta del módulo Procesar en Estación de Trabajo 1 del nivel Procedimiento Unitario es realizable en el supervisor S_{PET1} y también en el producto asíncrono de S_A y S_B.

Si la receta de cada módulo del nivel Procedimiento es un lenguaje realizable en los supervisores del módulo correspondiente del nivel Procedimiento Unitario, entonces también es realizable en el producto asíncrono de los supervisores de cada nivel Fase que pertenecen al modulo de la receta del nivel Procedimiento Unitario, por el teorema 4.2 propuesto en este trabajo de investigación.

Teorema 4.2. *$L(R)$ es el lenguaje de receta, S_n es un sistema confiable y S_{n-1} es un sistema confiable.*

Si $L(R)$ es un lenguaje realizable en S_n, entonces $L(R)$ es un lenguaje realizable en S_{n-1}.

Demostración. *Suponemos que S_n es un sistema confiable, S_{n-1} es un sistema confiable y $L(R)$ es un lenguaje realizable en S_n.*

Como S_n es el comportamiento a lazo cerrado, entonces si $s\sigma \in L(S_n)$ también $s\sigma \in L(G_n)$.

También como $L(G_n)$ es la proyección de S_{n-1} por construcción de la jerárquica de control, entonces $P^{-1}(L(G_n)\|L(S_{n-1})) = L(S_{n-1})$ y como S_{n-1} es confiable, entonces las trayectorias realizables de $L(G_n)$ se pueden realizar también en S_{n-1}. Por lo tanto $L(R)$ es un lenguaje realizable en S_{n-1}.

Como el control jerárquico y modular es un sistema confiable, y la receta de producción es realizable en el control jerárquico y modular, entonces la controlabilidad global queda garantizada.

4.5. Conclusiones

En esta sección se muestra un análisis de las diferentes estructuras de control presentadas en las secciones 4.1, 4.2, 4.3 y 4.4.

De acuerdo a las características de las diferentes estructuras presentadas en la tabla 4.8: La construcción de las estructuras I, II y IV están respaldadas por una metodología que garantiza la controlabilidad global y el no bloqueo. Sin embargo, en las estructuras III y IV existe un método para construir los componentes de los módulos del nivel superior y métodos automatizados para la validez de la teoría; y requieren menos recursos en su implementación, así como son mas fáciles de construir. Por otra parte, en la estructura II solo si cualquier interfaz de un módulo del nivel inferior es cambiada entonces también es cambiada el nivel superior. En el caso de que otra cosa sea modificado en cualquier módulo del nivel inferior, solo se modifica el nivel inferior; y esta también requiere menos recursos en su implementación. Por lo tanto, se concluye que la mejor de las estructuras por sus características es la estructura IV, aún cuando todas cumplen con el mismo propósito de control.

Tabla 4.8: Características de las Estructuras

Características	Estructura I	II	III	IV
Metodología que garantiza el no bloqueo y la controlabilidad global	3	3	1	3
Validez de la teoría de una manera automatizada	1	1	3	3
Hay un método a seguir para construir: La abstracción del nivel inferior, las interfaces de cada modulo del nivel inferior o los componentes de los módulos del nivel superior; según el caso.	1	1	3	3
Si un módulo del nivel inferior es modificado, el nivel superior también es modificado.	3	2	3	3
La implementación de la estructura requiere menos recursos computacionales.	1	3	3	3
Fácil de construir	1	1	3	3
Orden en el que cumplen las características: 3, 2, 1				

Capítulo 5

Implementación del Control Jerárquico en Matlab-Simulink

Una vez obtenidas las diferentes estructuras de control jerárquico al caso de estudio en las secciones 4.1, 4.2, 4.3 y 4.4 del capítulo 4, el siguiente paso es la implementación.

La implementación de las diferentes estructuras de control jerárquico se realizó como sigue: para cada estructura se desarrolló un sistema en Simulink de Matlab, donde el sistema se comunica mediante una tarjeta de adquisición de datos Sensoray© modelo 626 con una interfaz electrónica del sistema automatizado de manufactura (SAM) implementado en *Lego*® Para más detalle del diseño de la interfaz consultar [5].

En la sección 5.1 se da una breve descripción de Matlab, Simulink y Stateflow. En la sección 5.2 se describe la manera en que se implementó la estructura de control jerárquico I, en esta estructura se aplico la metodología de supervisión jerárquica de SED citada en [20](capítulo 5). Finalmente, en la sección 5.3 se describe la manera en que se implementó la estructura de control jerárquico III, en esta estructura se presenta el control de supervisión jerárquico y modular basado en la proyección natural del caso de estudio, propuesto en [17]. No se describe la implementación de las estructuras de control jerárquico II y IV debido a que los supervisores obtenidos en las estructuras de control jerárquico II, III y IV resultaron iguales.

5.1. Matlab, Simulink y Stateflow

Matlab es una aplicacion de alto desempeño computacional para cálculos técnicos. Este integra cálculos, visualización y programación de un ambiente sencillo, donde los problemas y las soluciones se expresan en una notación matemática familiar y sencilla. Es posible integrar otros lenguajes y aplicaciones a Matlab, como C, C++, Fortran, Java, COM y Excel de

Microsoft.

Simulink es un paquete de software que se ejecuta acompañando a Matlab para modelar, simular y analizar sistemas dinámicos. Además posee una interfaz gráfica de usuario (GUI), con diagramas de bloques para construir los modelos y una amplia biblioteca de bloques, donde también podemos personalizar y crear nuestros propios bloques. Simulink es utilizado para la implementación de las metodologías de control jerárquico, debido a las características mencionadas. Los elementos de la estructura de control de cada metodología se programan como bloques de Simulink para aprovechar la interfaz gráfica que ofrece.

Stateflow es una herramienta interactiva concebida para el diseño y simulación de sistemas dirigidos por eventos y se basa en la teoría de autómatas finitos. Además forma parte de un entorno de simulación y está estrechamente integrado con Matlab y Simulink, lo que da lugar a un entorno eficiente para modelar, simular y analizar sistemas dinámicos complejos.

5.2. Implementación de la Estructura de Control Jerárquico I

En esta sección se implementa la estructura de control jerárquico I aplicada al caso de estudio de la sección 4.1 del capítulo 4. En la figura 5.1 se muestra la implementación de la estructura de control jerárquico I en las estaciones de trabajo. Este sistema se desarrolló en Simulink, utilizando el bloque Chart de Stateflow para crear los bloques del supervisor y del buffer. En la figura 5.2 se muestra la implementación de la estructura de control jerárquico I en la estación de trabajo I. Este sistema se desarrolló en Simulink, utilizando el bloque S-Function para crear el bloque del supervisor y el bloque del buffer.

El diagrama del sistema de la figura 5.1 está formado por los siguientes bloques:

- Los bloques blancos con bordes azul cielo en la parte superior representan las recetas de producción y su funcionamiento es igual al descrito en la sección 5.3.
- Los bloques azul cielo representan los supervisores SA_{lo}, SM_{lo}, $SPET1_{lo}$ y $SPET1_{hi}$ que contienen los autómatas finitos de GA_{lo}, GM_{lo} y $GPET1_{lo}$, $SPET1_{hi}$ de cada estación de trabajo. Estos bloques son descritos en la subsección 5.2.1.
- Los bloques grises representan los buffers y su función es similar a la de los bloques fifo descritos en la sección 5.3. La diferencia es que si sucede más de una señal en un mismo instante en el nivel inferior, estas son almacenadas en una cola tal y como suceden. En cambio con el componente fifo se supone que solo una señal sucede en un instante.

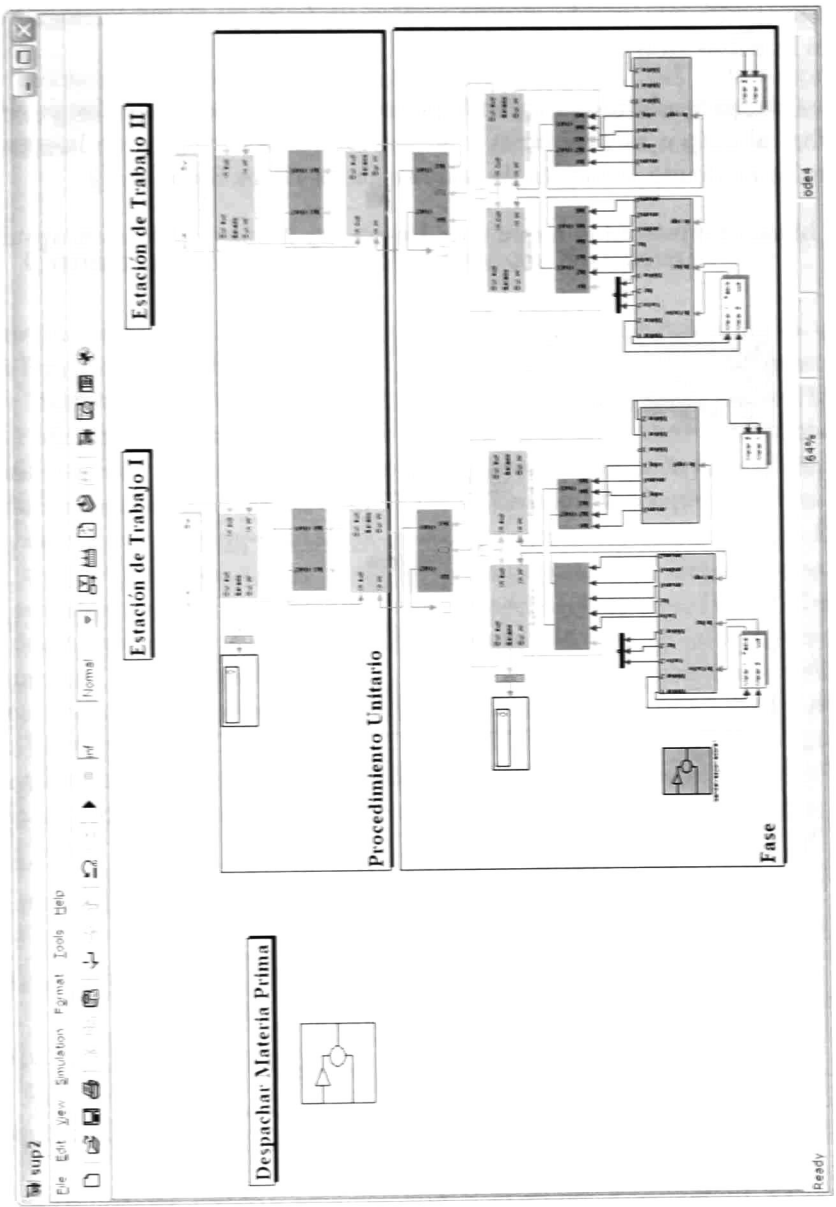


Figura 5.1: Implementación de la Metodología de Control Jerárquico I.

- Los bloques azul verde son una encapsulación de los bloques azules que representan los autómatas finitos que modelan los componentes elementales descritos en la sección 5.3 y otro nuevo bloque creado para la implementación de la metodología, el cual se describe en la subsección 5.2.1.
- Los bloques blanco con bordes rojos en la parte inferior son la encapsulación de los componentes digital output, digital input y analog input, requeridos para la comunicación con la interfaz mediante la tarjeta de adquisición de datos Sensoray©.
- El bloque blanco representa el módulo de Despachar Materia Prima encapsulado.

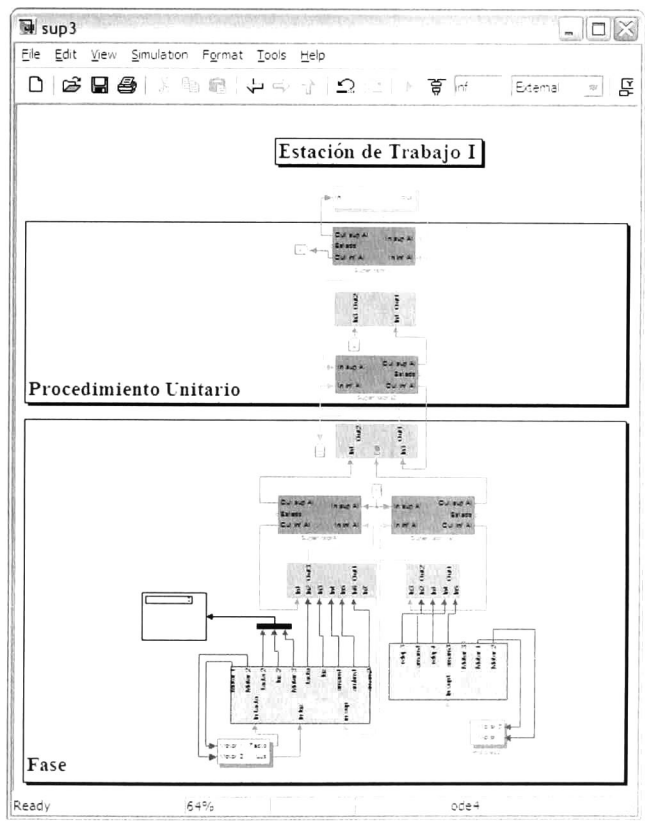


Figura 5.2: Implementación de la Metodología de Control Jerárquico I.

El diagrama del sistema de la figura 5.2 está formado por los mismos bloques de la figura 5.2; los únicos bloques diferentes son los siguientes:

Los bloques azul agua representan los supervisores SA_{lo}, SM_{lo}, $SPET1_{lo}$ y $SPET1_{hi}$ que contienen los autómatas finitos de GA_{lo}, GM_{lo},$GPET1_{lo}$ y $SPET1_{hi}$ de la Estación de Trabajo 1. Estos bloques son descritos en la subsección 5.2.1

- Los bloques café representa los buffers, semejante a la del buffer representado por el bloque gris; la diferencia es que este bloque es creado con bloques de la biblioteca de Simulink.

5.2.1. Construcción de los bloques en Simulink

La construcción de los bloques en Simulink se lleva a cabo mediante la utilización de los bloques S-Function y Chart, entre otros. El bloque S-function es un programa que puede ser escrito en Matlab, C, C++, Ada, o Fortran, el cual permite crear nuevos bloques en Simulink con el número de entradas y salidas requeridas para un caso determinado. Los bloques construidos con S-Function se programaron en el lenguaje C con el propósito de facilitar su comprensión y mantenimiento, en caso de ser requerido. El bloque Chart es un diagrama de Stateflow, el cual es una representación gráfica de una máquina de estados finitos, donde los estados y transiciones forman los bloques de construcción del sistema. Aunque también se puede representar un diagrama de flujo en Stateflow. El diagrama de Stateflow consiste de un conjunto de objetos gráficos (state, box, truth table function, graphical function, Embedded Matlab function, transitions, default transition, connective junctions y history junctions) y no gráficos (events, data y targets). El bloque Chart puede ser incluido en un modelo en Simulink. El diagrama de los bloques construidos con Chart está constituido por un estado, dos transiciones y funciones embebidas de Matlab. Estas funciones permiten introducir programas en código Matlab. Los bloques que se construyeron son los siguientes:

- Bloque de Receta
- Bloque Buffer
- Bloque Buffer Stateflow
- Bloque Supervisor C
- Bloque Supervisor Stateflow

En esta sección se presenta una breve descripción de la construcción de cada bloque.

Bloque de Receta

El bloque de receta implementa la receta de producción en el sistema. Este bloque, que se presenta en la figura 5.3, contiene la secuencia de transiciones que deben ser ejecutadas por el supervisor de Procedimiento Unitario para la elaboración del producto.

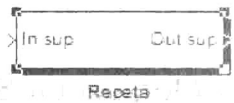


Figura 5.3: Bloque de Receta.

El bloque cuenta con una entrada y una salida. En la salida se envía hacia el supervisor de Procedimiento Unitario la transición correspondiente para la elaboración del producto y en la entrada se recibe información sobre la ejecución de la transición enviada al supervisor de Procedimiento Unitario.

Este bloque requiere de 2 datos. El dato 1 corresponde a una matriz de n elementos, que son las etiquetas de la secuencia de transiciones para la elaboración de un producto. El dato 2 es el tamaño de la matriz. Estos datos son introducidos en el cuadro de diálogo del bloque que se presenta en la figura 5.4.

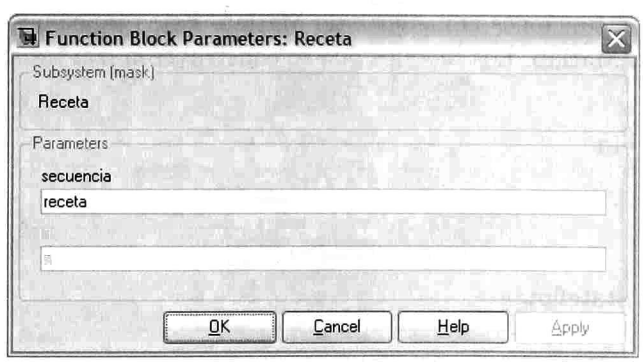


Figura 5.4: Parámetros: Bloque de Receta.

Este bloque funciona como sigue: el bloque de la receta envía hacia el supervisor de Procedimiento Unitario la transición correspondiente para la elaboración del producto. Cuando el supervisor de Procedimiento Unitario ejecuta la transición recibida del bloque de receta,

el supervisor envía la transición recibida. El bloque de receta al recibir la transición enviada recorre la lista de transiciones para enviar la siguiente de la lista hacia el supervisor de Procedimiento Unitario.

En el sistema se plantean tres bloques de receta, uno por cada supervisor de Procedimiento Unitario, que es el nivel en donde se implementa la receta de producción.

Bloque Buffer

El buffer tiene n entradas y 2 salidas. Las entradas de 2 a n son las transiciones recibidas del nivel inferior que son ordenadas en una cola por cada transición, donde cada cola arroja un escalar por instante de tiempo en forma secuencial, es decir: primero arroja la cola 1, cola 2, ... cola n y después se reinicia de nuevo. Este escalar se almacena en otra cola, desde la cual se envían las transiciones al supervisor por la salida "Sup". La entrada "In Sup" recibe las transiciones enviadas del supervisor, para enviarlas al nivel inferior por la salida "Inf". La figura 5.5 presenta el bloque buffer.

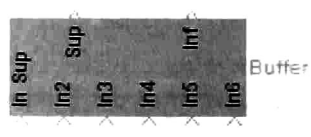


Figura 5.5: Bloque Buffer.

El bloque buffer esta formado por los subsistemas Queue y Mux Queue, como se muestra en la figura 5.6. Mux Queue recibe n-1 entradas, donde cada entrada tiene una cola (Queue), y arroja un escalar por instante de tiempo; este se muestra en la figura 5.7. Queue recibe los elementos a encolar en la entrada 2 y el vector con las instrucciones para el nivel inferior, así como las órdenes para desencolar en la entrada 1. Además envía los elementos desencolados en la salida 1, las órdenes para el nivel inferior en la salida 2. El bloque Queue este se muestra en la figura 5.8.

Al bloque buffer se le pueden eliminar o agregar entradas según lo requiera. El mínimo número de entradas es 2, donde la primera es la entrada del nivel superior y la otra entrada es del nivel inferior.

La manera como se agregan entradas al bloque buffer es la siguente:

1. Se selecciona el bloque buffer y se le da click derecho, posteriormente le aparece un menú en el cual se selecciona la opción "Look Under Mask". Posteriormente se abre la

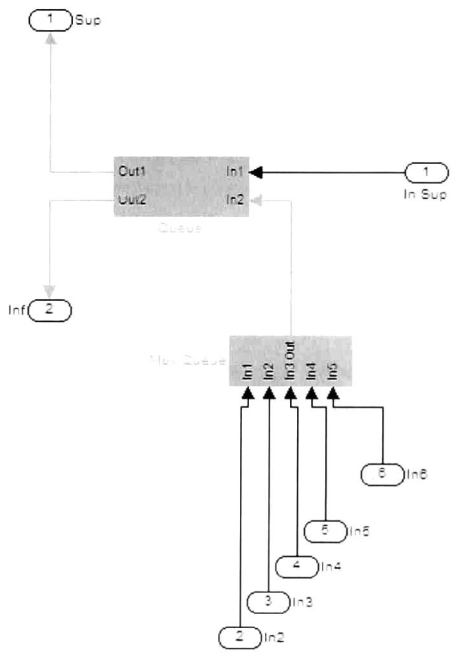


Figura 5.6: Elementos del Bloque Buffer.

ventana en la que se muestra la figura 5.6, donde el bloque que se va a modificar es Mux Queue.

2. Se selecciona el bloque Mux Queue y se le da doble click; después de esta acción, se abre la ventana en la cual se muestra la figura 5.7.

3. Una vez abierta esta ventana, se agregan las entradas requeridas.

4. Para agregar una entrada, se selecciona, copia y pega la entrada 5 y los bloques y conexiones de color gris.

5. Posteriormente, en la nueva entrada 6, se modifica el dato "Constant value" de los dos bloques "Compare to Constant" con los valores (número de entrada x 2)-2, (número de entrada x 2)-1. En este caso, si la entrada es 6 , los valores deberán ser 10, 11.

6. También se modifican los datos de los componentes de color café claro. En el bloque "Counter Limited" se modifica el "Upper Limit", donde por cada entrada agregada se le suma dos al número existente. En el bloque "Index Vector" se modifica "Number of Inputs" de la misma manera: por cada entrada agregada se le suma dos al número existente.

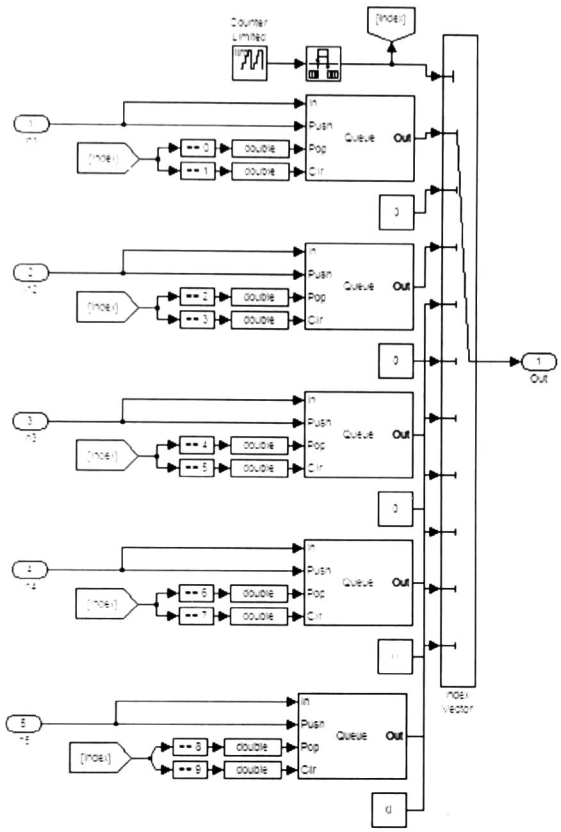


Figura 5.7: Bloque Buffer: Mux Queue.

7. Si se requiere otra entrada se regresa al paso 4.

8. Una vez que se terminó de agregar las entradas, se guardan los datos de la ventana que se muestra en la figura 5.7 y se regresa a la ventana en la que se muestra la figura 5.6, excepto que ahora el bloque Mux Queue ya tiene más entradas.

   a) En esta ventana se agregan los bloques "In" necesarios, según las entradas que se hayan agregado, y se conectan los bloques "In" a las nuevas entradas en el bloque Mux Queue.

   b) Se guardan las modificaciones de esta ventana.

   c) Se cierra esta ventana y se finaliza con un buffer con las entradas requeridas.

La manera en que se eliminan las entradas al bloque buffer es como sigue:

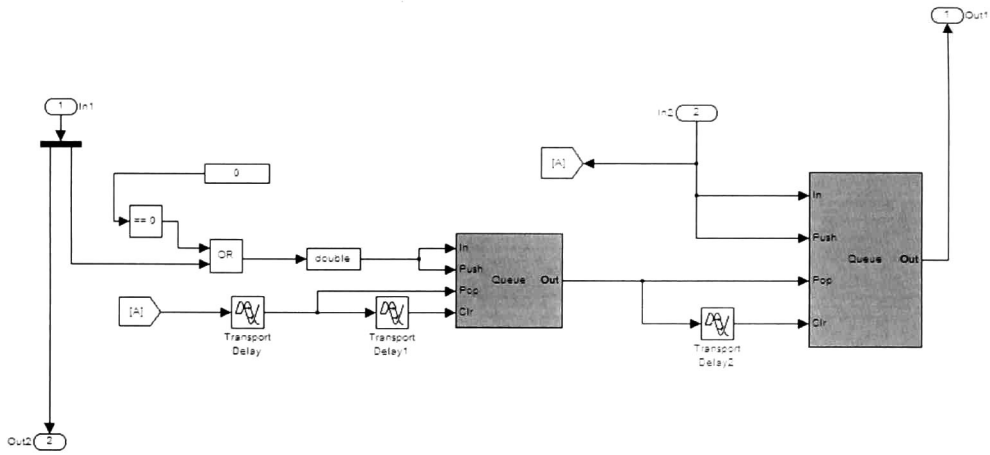


Figura 5.8: Bloque Buffer: Queue.

1. Se selecciona el bloque buffer y se le da click derecho; posteriormente aparece un menú en el cual se selecciona la opción "Look Under Mask". Posteriormente se abre la ventana en la que se muestra la figura 5.6, donde el bloque que se va a modificar es Mux Queue.

2. Se selecciona el bloque Mux Queue y se le da doble click, con lo cual se abre la ventana en la que se muestra la figura 5.7.

3. Una vez abierta esta ventana, se eliminan las entradas requeridas.

4. Para eliminar una entrada, se selecciona y elimina la entrada 5 y los bloques y conexiones que están de color gris.

5. También se modifican los datos de los componentes de color café claro. En el bloque "Counter Limited" se modifica el "Upper Limit", donde por cada entrada que se elimina se le resta dos al número existente. En el bloque "Index Vector" se modifica "Number of Inputs" de la misma manera: por cada entrada que se elimina se le resta dos al número existente.

6. Si se requiere eliminar otra entrada se regresa al paso 4.

7. Una vez que se terminó de eliminar las entradas, se guardan los datos de esa ventana y se regresa a la ventana en la que se muestra la figura 5.6, excepto que ahora el bloque Mux Queue tiene menos entradas.

a) En esta ventana se eliminan los bloques "In" innecesarios, según las entradas que se hayan eliminado.

b) Se guardan las modificaciones de esta ventana.

c) Se cierra esta ventana y se finaliza con un buffer que tiene las entradas requeridas.

Bloque Buffer Stateflow

Las figuras 5.9 y 5.10 muestran el bloque buffer stateflow y los componentes que lo conforman. Este bloque tiene la misma función que el bloque buffer. La diferencia radica en los bloques Queue y Mux Queue que lo conforman, ya que estos son construidos con el bloque Chart de Stateflow de Simulink.

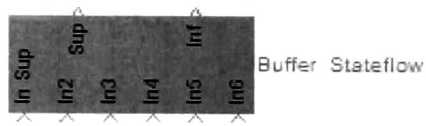


Figura 5.9: Bloque Buffer Stateflow.

La barra negra que se muestra en la figura 5.7 recibe n elementos y envía un vector de tamaño n. El bloque Mux Queue tiene una entrada y una salida. En la entrada se recibe un vector de longitud n, y por cada elemento del vector se maneja una cola. Por lo que el funcionamiento de este bloque es encolar cada elemento del vector entrante y desencolar un elemento por ejecución (unidad de tiempo), el cual se envía por la salida. En las figuras 5.11 y 5.12 se muestra el bloque Chart de Mux Queue y el diagrama del bloque Mux Queue en Stateflow.

El bloque Queue recibe los elementos a encolar en la entrada "In info", el vector con las instrucciones para el nivel inferior así como las órdenes para desencolar en la entrada "In sup". Después envía los elementos desencolados por la salida "Out sup" y las ordenes para el nivel inferior por la salida "Out inf". En las figuras 5.13 y 5.14 se muestra el bloque Chart de Queue y el diagrama del bloque Queue en Stateflow.

Al bloque buffer stateflow se le pueden eliminar o agregar entradas según se requiera. El mínimo número de entrados es 2, donde la primera es la entrada del nivel superior y la otra entrada es del nivel inferior.

La manera en que se agregan entradas al bloque buffer stateflow es como sigue:

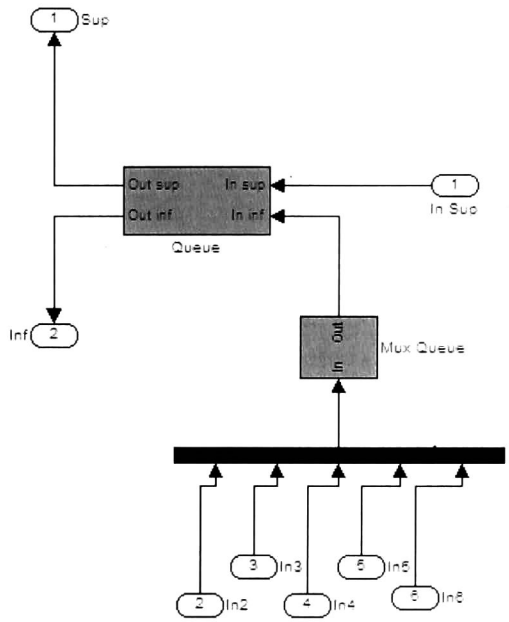


Figura 5.10: Componentes del Bloque Buffer Stateflow.

1. Se selecciona el bloque buffer stateflow y se le da click derecho; posteriormente aparece un menú en el cual se deberá seleccionar la opción "Look Under Mask" Posteriormente se abre la ventana en la que se muestra la figura 5.10.

2. En esta ventana se modifica el bloque "Mux" en el dato "Number of Inputs" con el número de entradas requeridas. Después se agregan los bloques "In" necesarios, según las entradas que se agregaron, y se conectan los bloques "In" a las nuevas entradas del bloque Mux. Posteriormente el bloque que se va a modificar es Mux Queue.

3. Se selecciona el bloque Mux Queue y se le da doble click; después de esta acción, se abre la ventana en la cual se muestra la figura 5.11, donde el bloque que se va a modificar es Chart.

4. Se selecciona el bloque Chart y se le da doble click; después de esta acción, se abre la ventana en la cual se muestra la figura 5.12.

5. Una vez abierta esta ventana, se va al menú "Tools⟶Explore"; después de esta acción, se abre la ventana en la cual se muestra la figura 5.15.

6. Una vez abierta esta ventana, se modifican los datos: "cola" en el dato "Size[num. entradas, 20]" "in" en el dato "Size[num. entradas]" y "tope" en el dato "Size[num.

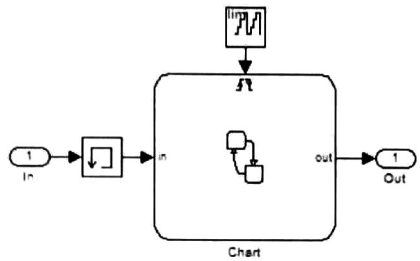


Figura 5.11: Bloque Buffer Stateflow: Chart de Mux Queue.

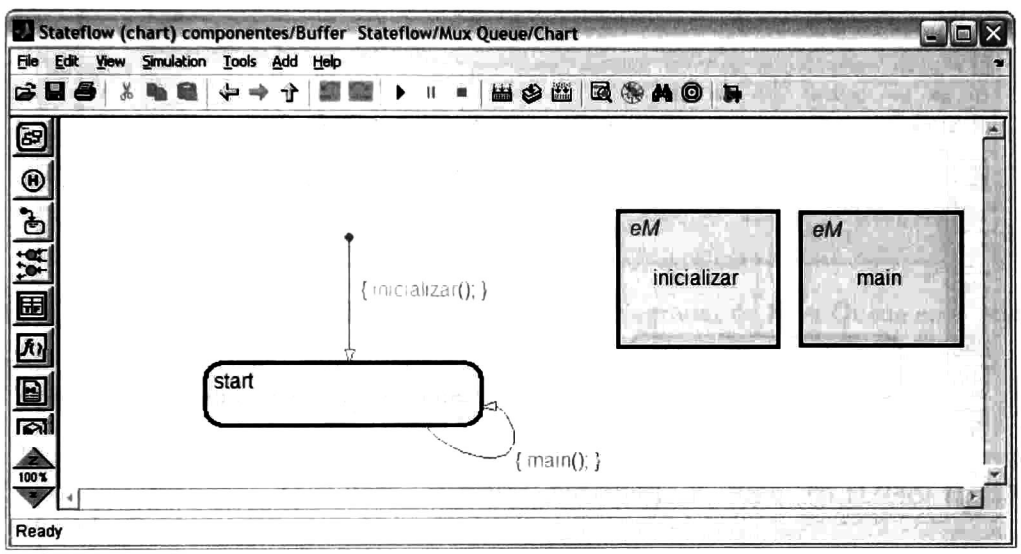


Figura 5.12: Bloque Buffer Stateflow: Diagrama de Mux Queue en Stateflow.

entradas]"

7. Una vez que se terminó de agregar las entradas, se guardan los datos de esa ventana y se cierran las ventanas.

8. Se finaliza con un buffer stateflow con las entradas requeridas.

La manera en que se eliminan entradas en el bloque buffer stateflow es como sigue:

1. Se selecciona el bloque buffer stateflow y se le da click derecho, posteriormente aparece un menú en el cual se habrá de seleccionar la opción "Look Under Mask". Posteriormente se abre la ventana en la que se muestra la figura 5.10.

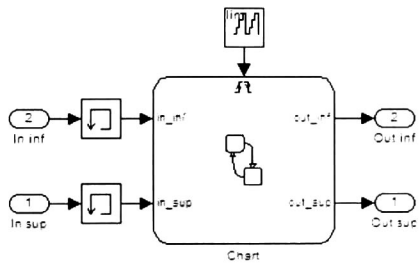


Figura 5.13: Bloque Buffer Stateflow: Chart de Queue.

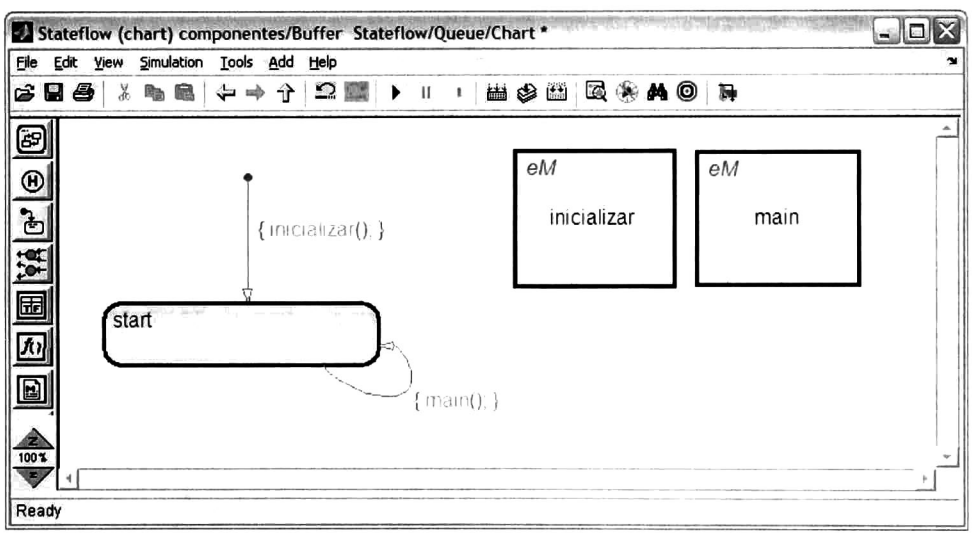


Figura 5.14: Bloque Buffer Stateflow: Diagrama de Queue en Stateflow.

2. En esta ventana se modifica el bloque "Mux" en el dato "Number of Inputs" con el número de entradas requeridas. Después se eliminan los bloques "In" innecesarios. Posteriormente el bloque que se va a modificar es Mux Queue.

3. Se selecciona el bloque Mux Queue y se le da doble click; después de esta acción, se abre la ventana en la cual se muestra la figura 5.11, donde el bloque que se va a modificar es Chart.

4. Se selecciona el bloque Chart y se le da doble click; después de esta acción, se abre la ventana en la cual se muestra la figura 5.12.

5. Una vez abierta esta ventana, se va al menú "Tools—→Explore" después de esta acción,

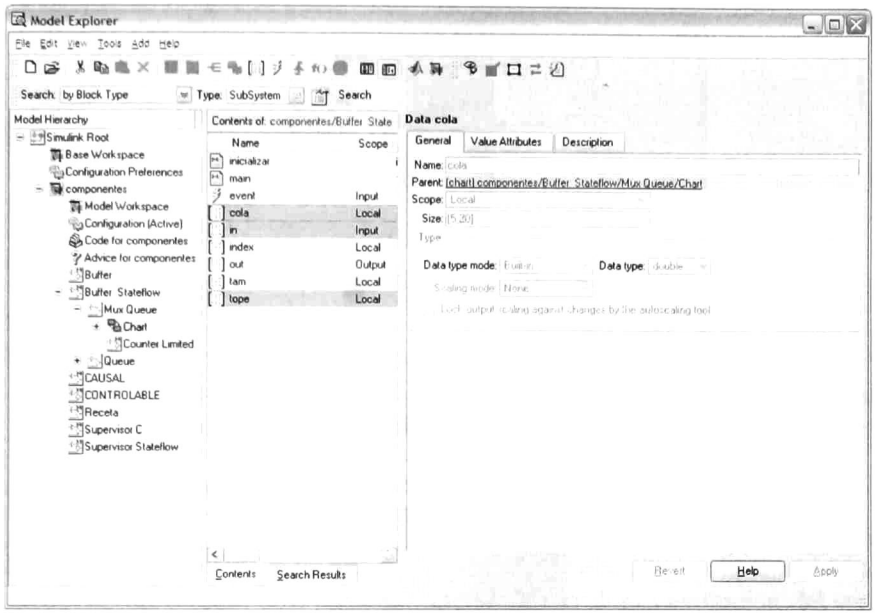


Figura 5.15: Bloque Buffer Stateflow: Parámetros del Diagrama de Mux Queue en Stateflow.

se abre la ventana en la cual se muestra la figura 5.15.

6. Una vez abierta esta ventana, se modifican los datos: "cola" en el dato "Size[num. entradas, 20]", "in" en el dato "Size[num. entradas]" y "tope" en el dato "Size[num. entradas]".

7. Una vez que se terminó de agregar las entradas, se guardan los datos de esa ventana y se cierran las ventanas.

8. Se finaliza con un buffer stateflow con las entradas requeridas.

Bloque Supervisor C

Este bloque es construido con una S-function, la cual cuenta con tres entradas y tres salidas. La primera entrada recibe la señal del reloj, la segunda recibe transiciones enviadas por el supervisor del nivel superior y la tercera recibe la transición enviada por el nivel inferior. En la salida 1 se envían las transiciones recibidas del nivel superior y en la salida 3 se envían las transiciones del supervisor, que se requiere que se ejecuten en el nivel inferior, y la orden para desencolar el buffer. La salida 2 muestra el estado del generador requerido. La figura 5.16 presenta el bloque supervisor C.

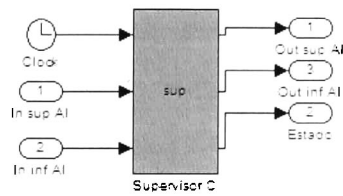


Figura 5.16: Bloque Supervisor C.

Se crea un subsistema con el bloque supervisor C y el bloque del reloj del sistema que es enmascarado, como se presenta en la figura 5.17, para facilitar su conexión. El bloque resultante cuenta solo con dos entradas y tres salidas.

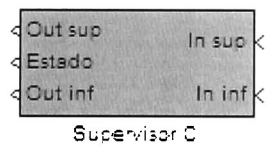


Figura 5.17: Bloque Supervisor C.

Este bloque requiere de 5 datos, los cuales son: m es una matriz que representa la función de transición de un generador requerido, mv es la matriz que representa los estados vocalizados del generador, tm son los renglones de la matriz m, tmv son los renglones de la matriz mv, y nivel, cuyo valor es "1" para el supervisor del nivel 1 y "0" para los supervisor del nivel 2-n. Estos datos son introducidos en el cuadro de diálogo del bloque que se presenta en la figura 5.18.

Los parámetros mn3, mvn3, tmn3 y tmvn3 se definen en un archivo n3.m, como se muestra en la figura 5.19.

Este bloque representa el comportamiento del supervisor de la metodología citada en [20](capítulo 5). Si el supervisor que se quiere representar es del nivel superior, la matriz m corresponde al supervisor S_{hi} y la matriz mv no tiene nada. Pero si el supervisor es de nivel inferior, las matrices m y mv corresponden a G_{lo}.

Este bloque funciona como sigue: se inicia en el estado actual de la matriz m, en espera de un transición. Cuando se recibe una transición por la entrada "In sup" del nivel superior, se comprueba si esta transición es igual a la vocalización de un estado de la matriz m. Esto se realiza en la matriz mv, ya que está contiene el estado y la vocalización. Si la transición

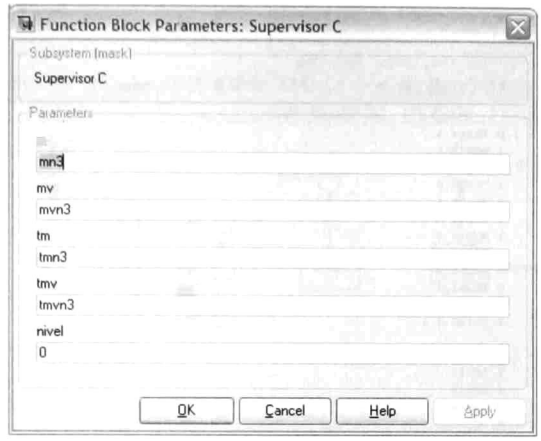


Figura 5.18: Parámetros: Bloque Supervisor C.

resultó no ser una vocalización de un estado, no se hace nada. Pero si la transición resultó ser una vocalización de un estado, ésto da la pauta para controlar la evolución del supervisor, ya que sólo las secuencias de transiciones que llevan del estado actual al estado vocalizado son posibles. La secuencia ejecutada es de acuerdo a la prioridad de ejecución de las transiciones en cada estado. En cada estado existe una lista de transiciones habilitadas por el supervisor que son ejecutadas en el siguiente orden de acuerdo a su prioridad: las transiciones incontrolables tienen la prioridad más alta. Si una transición incontrolable se encuentra habilitada en el supervisor sola o con alguna transición controlable, esta última es ejecutada inmediatamente (siempre y cuando esta transición haya sido recibida por la entrada "In inf"). Las transiciones controlables tienen la prioridad más baja y se pueden presentar varias de ellas habilitadas al mismo tiempo en algún estado, en este caso cualquiera de ellas se puede ejecutar (esta transición es enviada por la salida "Out inf").

Después de que se ejecuten ciertas transiciones y se llegue a un estado vocalizado, el bloque supervisor envía la vocalizacion de este estado al nivel superior por la salida "Out sup".

Bloque Supervisor Stateflow

La figura 5.20 presenta el bloque supervisor stateflow, el cual tiene dos entradas y tres salidas. Este bloque tiene la misma función que el bloque supervisor C, la diferencia radica en que el bloque supervisor stateflow fue construido con el bloque Chart. El bloque Chart con el que se construyó el bloque supervisor stateflow se muestra en la figura 5.21.

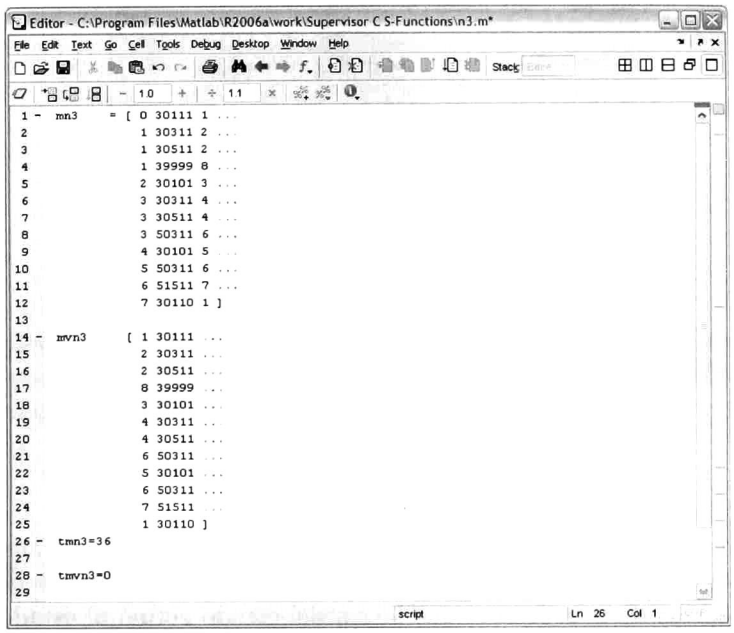


Figura 5.19: Definicion de mn3, mvn3, tmn3 y tmvn3 en el archivo n3.m

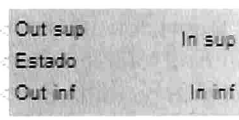


Figura 5.20: Bloque Supervisor Stateflow.

El bloque Chart con el que se construyó el bloque supervisor requiere de 3 parámetros, los cuales son: m es una matriz que representa la función de transiciones de un generador requerido, mv es la matriz que representa los estados vocalizados del generador requerido, nivel indica si el supervisor es S_{lo} del nivel fase. Los nombres de las matrices se introduce en los bloques de color amarillo y verde que se muestran en la figura 5.21; y el nivel se introduce en el bloque de color rojo.

Si con este bloque se construye un supervisor(S_{hi}), lo que se modifica es el dato "Data" con el nombre de matriz m definida para ese bloque supervisor stateflow del bloque "From Workspace" de color amarillo. En el bloque rojo se modifica el dato "Constant value" con el valor de 0.

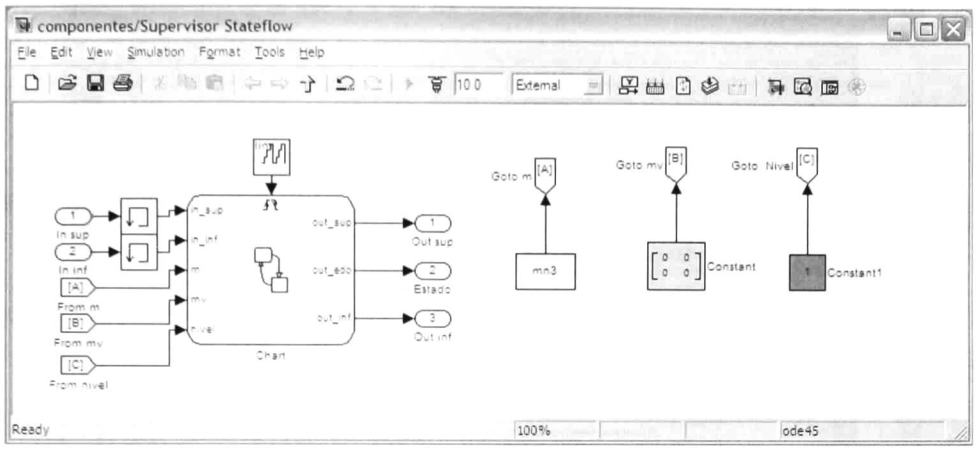


Figura 5.21: Bloque Chart del Bloque Supervisor Stateflow.

En caso de que con este bloque se construya un supervisor(S_{lo}), se elimina el bloque verde y se copia el bloque amarillo, el que pasará a ocupar el lugar del bloque verde eliminado. Posteriormente se modifica el dato "Data" con el nombre de matriz m o mv, según sea el caso para este bloque supervisor stateflow, de los bloques "From Workspace" de color amarillo. Si S_{lo} está en el nivel fase, el bloque de color rojo queda como está; en caso contrario en el bloque rojo el dato "Constant value" se modifica con el valor de 0.

Las matrices m y mv requeridas sea el caso, se definen en un archivo.m, como se muestra en la figura 5.22.

El diagrama del bloque Chart en Stateflow con el que se construyó el bloque supervisor stateflow se muestra en la figura 5.23.

5.2.2. Tarjeta de Adquisición de Datos

Para la implementación del control se emplea una tarjeta de adquisición de datos Marca Sensoray, modelo 626.

La tarjeta Sensoray modelo 626 es una tarjeta de bajo costo y se conecta al exterior mediante 5 cables planos.

Esta tarjeta cuenta con 48 señales digitales bidireccionales. Las entradas digitales se emplean para la lectura del sensor de tacto. Cada sensor de tacto requiere de una entrada digital.

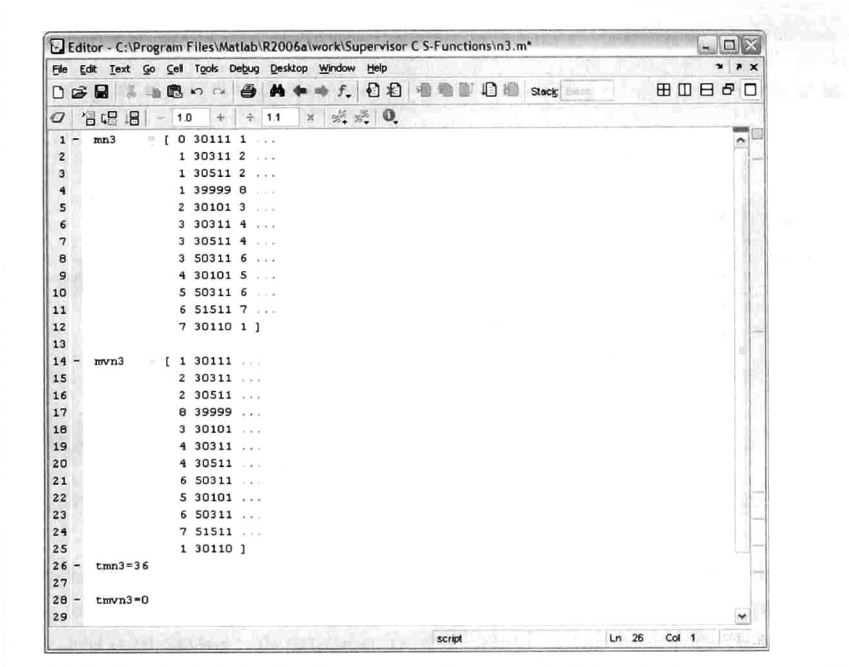


Figura 5.22: Definición de mn3 y mvn3 en el archivo n3.m

Las salidas digitales se emplean para el control de los motores. Cada motor requiere de dos señales digitales para su funcionamiento, dependiendo del sentido de giro requerido.

También cuenta con dieciséis entradas analógicas diferenciales de 16 bits que pueden configurarse para 5 ó 10 volts. Estas entradas son empleadas para las lecturas de voltaje provenientes de los sensores de luz. Cada sensor de luz requiere de una entrada analógica.

Además cuenta con tres contadores de 24 bits, que pueden ser configurados para 6 encoders incrementales de 24 bits de resolución, cuatro salidas de voltaje (salidas analógicas), que pueden ser programadas para 10 volts y un paro de emergencia, que puede ser activado por un relay mecánico o programado.

Una de las características más importantes de esta tarjeta es que permite una interacción en el ambiente de Simulink, en donde fue programado el control, aunque es posible trabajar en diferentes sistemas operativos.

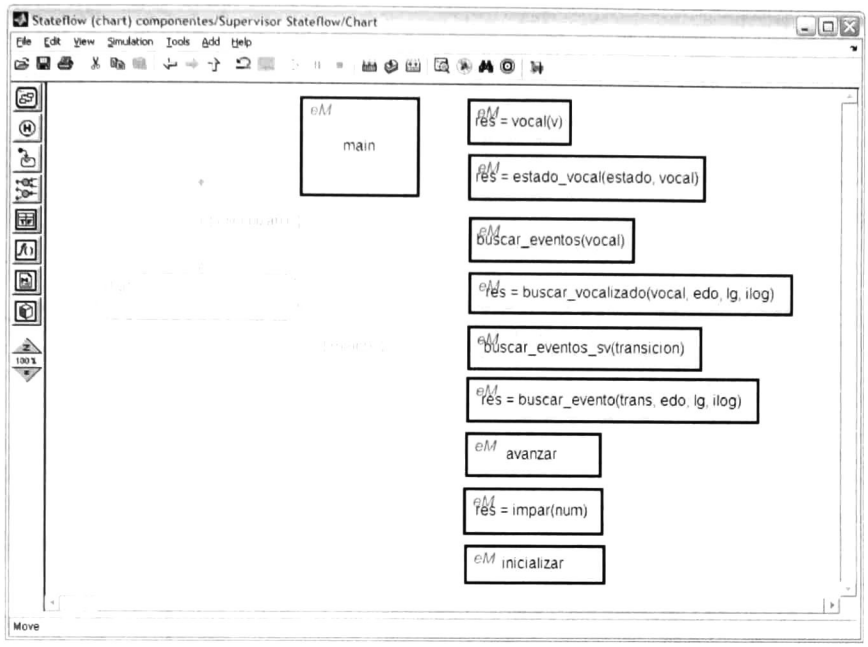


Figura 5.23: Diagrama del Bloque Chart en Stateflow.

5.2.3. Construcción de un Esquema de Control con la Estructura de Control Jerárquico I

La construcción de un esquema de control basada en la estructura de control jerárquico I se realiza utilizando dos bibliotecas de Simulink existentes en el Cinvestav, Unidad Guadalajara. La biblioteca "Control Supervisor Lego" fue desarrollada por [17]; y la otra biblioteca "Control Supervisor (Wonham)" se desarrolló en este trabajo y se muestra en la siguiente figura 5.24.

Las instrucciones para construir un esquema de control son las siguientes:

En un diagrama de Simulink nuevo se colocan los componentes del sistema que se ha construido y que se desea controlar.

La biblioteca "Control Supervisor Lego" contiene alguno de los componentes elementales empleados en este esquema: motor, temporizador, sensor de luz sensor de tacto y comportamiento causal.

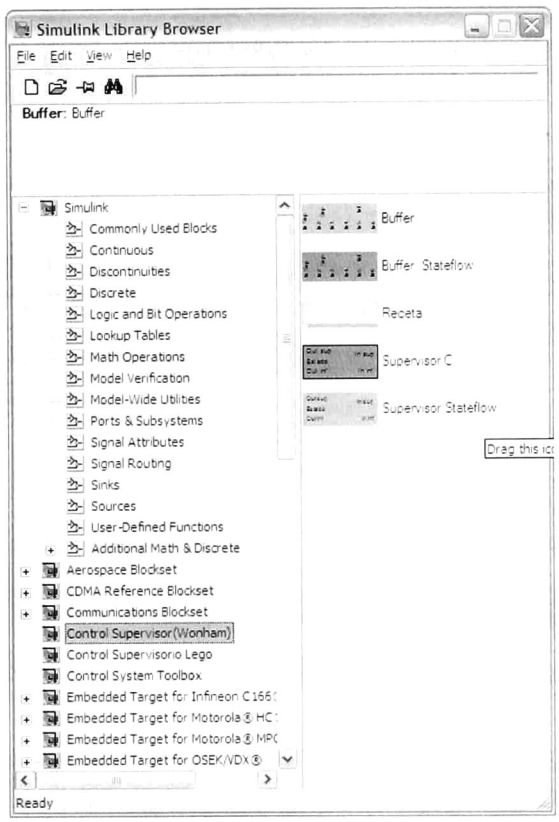


Figura 5.24: Biblioteca "Control Supervisor (Wonham)"

Para agregar un componente elemental al sistema, éste se arrastra de la biblioteca, se coloca en el diagrama de Simulink y se llenan sus parámetros; este paso se repite para cada uno de los componentes elementales del sistema.

Una vez que se han colocado todos los componentes elementales del sistema se procede a colocar los bloques buffer y supervisor que se encuentran en la biblioteca "Control Supervisor (Wonham)"

Si se decide colocar los bloques buffer y supervisor C, éstos se arrastran de la biblioteca y se colocan en el diagrama de Simulink. Estos bloques se configuran según sea el caso, como se muestra en las subsecciones 5.2.1, 5.2.1.

El otro caso es si se decide colocar los bloques buffer stateflow y supervisor stateflow.

Se arrastran los bloques de la biblioteca y se colocan en el diagrama de Simulink. Luego se configuran según sea el caso, como se muestra en las subsecciones 5.2.1, 5.2.1.

Finalmente se colocan los bloques de receta para cada supervisor del nivel superior. Estos bloques se arrastran de la biblioteca "Control Supervisor (Wonham)" se colocan en el diagrama de Simulink y se agregan sus parámetros.

5.3. Implementación de la Estructura de Control Jerárquico III

En esta sección se muestra la implementación de la estructura de control jerárquico III, que fue aplicada al caso de estudio en la sección 4.3 del capítulo 4. Esta implementación es mostrada en la figura 5.25, donde se presenta el diagrama del sistema en Simulink construido con los bloques que forman la biblioteca "Control Supervisor Lego" de Simulink.

El diagrama del sistema está formado por los siguientes bloques:

- Los bloques naranja en la parte superior son las recetas de producción, que contienen la secuencia de transiciones necesarias para la elaboración de un producto. Cada bloque contiene una salida que está conectada al supervisor del nivel superior. Mediante esta salida se envía información acerca de la transición a ejecutar por el supervisor. Asimismo, cada bloque contiene una entrada en la que se recibe información acerca de la ejecución de una transición en el supervisor del nivel inferior al de la receta.

- Los bloques rojos representan los supervisores que contienen el autómata finito del supervisor sintetizado y el orden de ejecución de las transiciones. Contienen dos entradas: la entrada 1 recibe las transiciones controlables que deben ser ejecutadas del nivel superior; la entrada 2 recibe las transiciones incontrolables generadas en el nivel inferior. También contienen dos salidas: la salida 1 envía información hacia el nivel superior sobre la ocurrencia de transiciones incontrolables que son de relevacia para el nivel superior y envía información sobre la ocurrencia de transiciones controlables recibidas del nivel superior; la salida 2 envía las transiciones controlables que deben ser ejecutadas por algún elemento del nivel inferior y señales sobre la evolución mediante una transición incontrolable para que sea modificada la lista de transiciones del bloque fifo.

- Los bloques verdes son las fifos inferior y superior, que se encargan de ordenar de acuerdo al orden en que se presentan las transiciones incontrolables provenientes del nivel inferior para enviarlas al supervisor. Este bloque cuenta con 2 salidas: la salida 1

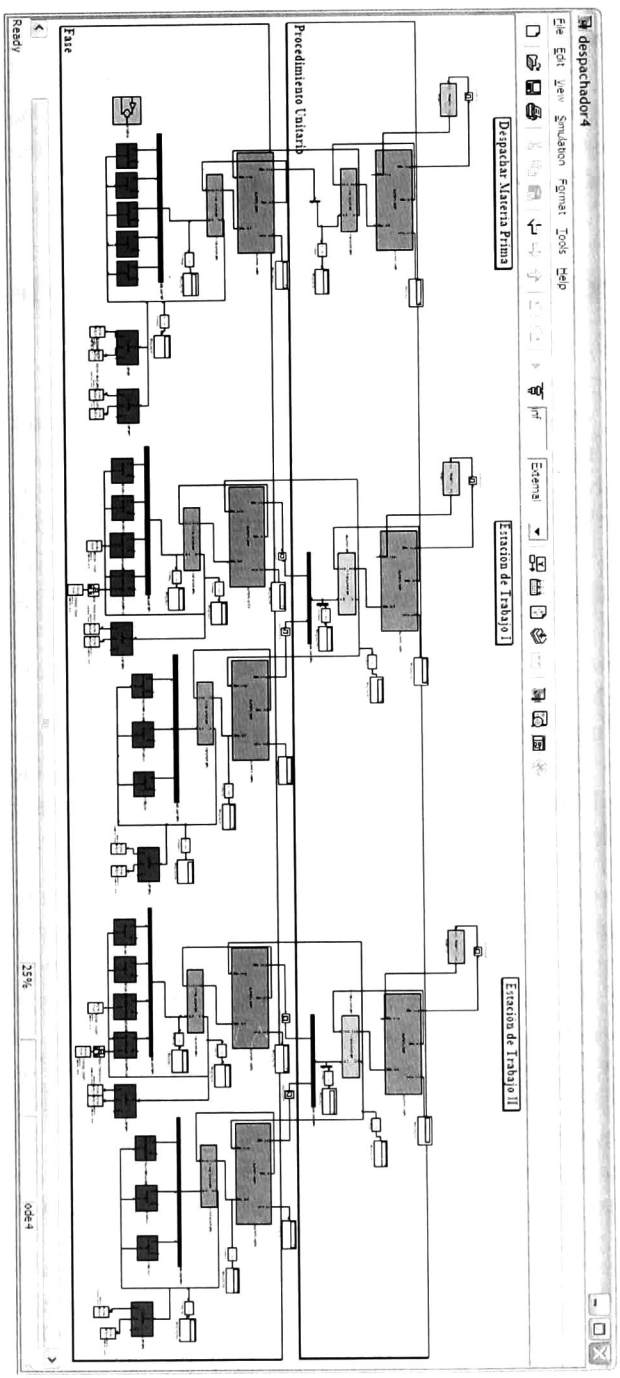


Figura 5.25: Implementación de la Metodología de Control Jerárquico III.

envía al supervisor la primera transición de la lista de transiciones incontrolables que posee; la salida 2 envía al nivel inferior las transiciones controlables que se presentan en el supervisor para que sean ejecutadas. También cuenta con dos entradas: la entrada 1 recibe información del supervisor sobre la evolución de éste empleando la primera transición de la lista enviada por este bloque. Cuando esta información es recibida, la lista es modificada y los elementos de la lista son recorridos. Esta entrada también recibe información sobre la ejecución de una transición controlable en el supervisor superior, que a su vez envía a los elementos del nivel inferior a través de la salida 2. La entrada 2 recibe las transiciones de nivel inferior que son ordenadas en la lista de transiciones incontrolables.

Los bloques negros son los colectores, que proporcionan el único canal para la transmisión de las transiciones incontrolables hacia el nivel superior. Este bloque colector se hace necesario cuando un supervisor coordina más de un elemento del nivel inferior; en caso contrario este bloque puede ser omitido.

- Los bloques azules son los autómatas finitos que modelan los componentes elementales. Estos bloques envían las transiciones incontrolables que suceden en cada uno de ellos al bloque colector. Algunas de estas señales son recibidas por el sistema automatizado de manufactura (SAM), implementado en *Lego*®, a través de la tarjeta de adquisición de datos; y otras son señales de software generadas directamente por los bloques de los componentes.

  El bloque naranja de la parte inferior representa la banda, que controla el motor que mueve a la banda transportadora.

Para más detalles referentes a la programación de los bloques de la utilería consultar [17](capítulo 4),[11], [12].

La estructura del diagrama del sistema está apegada al modelo híbrido del caso de estudio obtenida en el capítulo 3, sección 3.9. Esta estructura se muestra claramente, ya que en el nivel fase se encuentran los componentes elementales y sus supervisores de los módulos: Despachar Materia Prima A o B, Alimentar Materia Prima A o B y Maquinar y Entregar en la Estación de Trabajo 1, e igualmente para la Estación 2. En el nivel de Procedimiento Unitario se encuentran los supervisores de los módulos: Despachar materia prima, Procesar en Estación de Trabajo 1 y Procesar en Estación de Trabajo 2 de ese nivel.

Capítulo 6

Conclusiones y Trabajo Futuro

En esta tesis, se presenta el diseño e implementación de diferentes estructuras de control jerárquicas y modulares de un prototipo de un sistema automatizado de manufactura flexible hecho en *Lego*®

El prototipo cuenta con una interfaz electrónica destinada al acondicionamiento de señales y la comunicación entre el prototipo y una PC. La estructura y su funcionamiento del SAM exhibe un comportamiento con cierto grado de complejidad, representativo del comportamiento en sistemas mas grandes donde si se presenta, durante la síntesis, el problema de explosión de estados. Por lo tanto, en esta tesis se ilustra como atacar este problema, diseñando una estructura jerárquica y modular que permita controlar el prototipo.

La metodología de modelado de las estructuras del prototipo se basó en el modelo híbrido construido con los estándares industriales ISA [2, 1], permitiendo así separar las actividades de control de equipo de las actividades propias de producción. Por lo tanto, no es necesario modificar los supervisores que controlan los equipos para cambiar las órdenes de producción.

La construcción en general de las estructuras de control del prototipo se realizó aplicando la teoría de control supervisor (TSC). La 1^{ra} estructura se construyó aplicando la metodología (supervisión jerárquica de SED citada en [20] en el capítulo 5) que garantiza la controlabilidad global y el no bloqueo. Sin embargo, no existen métodos para construir las abstracciones del nivel inferior (es a prueba. error y experiencia adquirida) y métodos automatizados para la validez de la teoría, si un módulo del nivel inferior es modificado, el nivel superior también es modificado; esta estructura es difícil de construir y requieren más recursos computacionales en su implementación.

La 2^{da} estructura se construyó aplicando la metodología (control jerárquico basada en interfaz citado en [10]) que garantiza la controlabilidad global y el no bloqueo. Sin embargo,

no existen métodos para construir las interfaces de cada modulo del nivel inferior y métodos automatizados para la validez de la teoría, solo si cualquier interfaz de un módulo del nivel inferior es modificada entonces también es modificada el nivel superior. En el caso de que otra cosa sea modificada en cualquier módulo del nivel inferior, solo se modifica ese módulo del nivel inferior; esta estructura es difícil de construir, pero requieren menos recursos computacionales en su implementación.

La 3^{ra} estructura de control jerárquico fue propuesta en [17], donde la modularidad de la estructura con alfabetos disjuntos del nivel fase del caso de estudio fue tomado de [20, 21] y la jerarquía de la estructura se construyó aplicando la proyección natural de los eventos con estatus operacional de los supervisores del nivel Fase. Sin embargo, existen métodos para construir los componentes de los módulos del nivel superior y métodos automatizados para la validez de la teoría, si un módulo del nivel inferior es modificado, el nivel superior también es modificado; esta estructura es fácil de construir y requieren menos recursos computacionales en su implementación.

La 4^{ta} estructura se construyó aplicando los conceptos de sistema confiable, lenguaje realizable presentados en el capítulo 2, sección 2.2; proyección natural confiable presentada en el teorema 4.1 y lenguaje realizable por contención presentado en el teorema 4.2 establecido en este trabajo de investigación. Todo esto permite garantizar la controlabilidad global y el no bloqueo. Sin embargo, existen métodos para construir los componentes de los módulos del nivel superior y métodos automatizados para la validez de la teoría, si un módulo del nivel inferior es modificado, el nivel superior también es modificado; esta estructura es fácil de construir y requieren menos recursos computacionales en su implementación. Por lo tanto, se concluye que la mejor de las estructuras por sus características es la 4^{ta} estructura, aún cuando todas cumplen con el mismo propósito de control.

Se implementaron la 1^{ra} y 3^{ra} estructuras de control jerárquico mediante Simulink de Matlab en una PC, utilizando la biblioteca "Control Supervisor Lego" de Simulink desarrollada por [17], y la otra biblioteca "Control Supervisor (Wonham)" desarrollada en este trabajo de investigación; estas bibliotecas permiten la construcción y el análisis de las estructuras de control jerárquico y modular de forma gráfica. No obstante, no se implementaron la 2^{da} y 4^{ta} estructuras de control jerárquico, debido a que los supervisores sintetizados son similares a la 3^{ra} estructura de control jerárquico.

El formalismo existente para la construcción de la 4^{ta} estructura de control jerárquico consiste en: validar que un supervisor sea un sistema confiable en la estructura, cuando la receta de producción es un lenguaje realizable en cada nivel [9]; conservar la propiedad de sistema confiable entre los niveles mediante la proyección natural, como establece el teorema 4.1 citado en el capítulo 4, sección 4.4; garantizar que si la la receta es un lenguaje realizable

en supervisor del nivel n, entonces la receta es un lenguaje realizable en supervisor del nivel n-1 dado por el teorema 4.2 citado en el capítulo 1, sección 4.4. Por lo tanto, lo que falta para un trabajo futuro en esta formalización es buscar una clasificación de las especificaciones que permiten la obtención de un supervisor que sea un sistema confiable.

Bibliografía

[1] ANSI/ISA-88.01-1995. *Batch Control: Models and Terminology.* ISA, 1995.

[2] ANSI/ISA-95.00.01-200. *Enterprise: Control System Integration.* ISA, 2000.

[3] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic, 1999.

[4] S. Chafik and E. Niel. Hierarchical-decentralized solutions of supervisory control. *3rd International Symposium on Mathematical Modelling*, 2000.

[5] F A. García. Desarrollo formal de controladores lógicos. Aplicación a un caso de estudio en sistemas de manufactura. Tesis de maestría, Departamento de Control Automático, CINVESTAV IPN Unidad Guadalajara, 2007.

[6] P. Gohari and W. M. Wonham. A linguistic framework for controlled hierarchical des. *Proceedings of the 4th IEE Workshop on Discrete Event Systems*, 1998.

[7] D. Gouyon, J. F. Petin, and A. Gouin. Pragmatic approach for modular control synthesis and implementation. *International Journal of Production Research*, 2004.

[8] E. Hernandez. Modelado y control-jerárquico de la coordinación de recursos de sistemas de manufactura flexible. Tesis de maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2005.

[9] F Jaimes. Modelado y control de sistemas automatizados de manufactura. Tesis de maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2004.

[10] R. Leduc, M. Lawford, and P. Dai. Hierarchical interface-based supervisory control of a flexible manufacturing system. *IEEE Transactions on Control Systems Technology*, 14(4), July 2006.

[11] L. Llamas and A. Sánchez. *Biblioteca de Simulink: Control Supervisor Lego Manual De Usuario*, 2008.

[12] L. Llamas and A. Sánchez. *Biblioteca de Simulink: Control Supervisor Lego Manual Técnico*, 2008.

[13] J. A. Nava. Arquitectura basada en isa95 e isa88 para el control de sistemas continuos, discretos y por lotes. Tesis de maestría, Departamento de Control Automático CINVESTAV IPN Unidad Guadalajara, 2005.

[14] M. Nourelfath and E. Niel. Modular supervisory control of an experimental automated manufacturing system. *Control Engineering Practice*, 12, 2004.

[15] L. F. Parra, R. Baird A. Sánchez, and S. Macchietto. Hybrid modeling and dynamic simulation of automated batch plants. *ISA Transactions*, 42(3), July 2003.

[16] S. Perk. Hierarchical design of discrete event controllers: An automated manufacturing system case study. Tesis de maestría, Friedrich-Alexander-Universität Erlangen-Nürnberg Lehrstuhl für Regelungstechnik, 2004.

[17] H. Rivera. Estudio comparativo de estrategias de control jerárquico-modular para sistemas automatizados de manufactura. Tesis de maestría, Departamento de Ingeniería Eléctrica CINVESTAV IPN, 2007.

[18] T. J. Williams. *A Reference Model for Computer Integrated Manufacturing.* Instrument Society of America, 1991.

[19] T. J. Williams and H. Li G. A. Rathwell. *A Handbook on Master Planning and Implementation for Enterprise Integration Programs.* Instrument Society of America, 1991.

[20] W.M. Wonham. *Notes: Supervisory Control of Discrete-Event System.* University of Toronto, 2006.

[21] T.S. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 12, 2002.

[22] H. Zhong and W. M. Wonham. On the consistency of hierarchical supervision in discrete event systems. *IEEE Transactions on Automatic Control*, 35, 1990.

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Síntesis de Arquitecturas de Coordinación en Sistemas Automatizados de Manufactura

del (la) C.

Liz Eréndira LLAMAS LÓPEZ

el día 31 de Marzo de 2008.

Dr. Arturo del Sagrado Corazón Sánchez Carmona
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Raúl Ernesto González Torres
Investigador CINVESTAV 2C
CINVESTAV Unidad Guadalajara

Dr. Mario Angel Siller González Pico
Investigador CINVESTAV 2A
CINVESTAV Unidad Guadalajara