UNIVERSIDAD NACIONAL DE LA PLATA

Facultad de Informática

Magíster en Ingeniería de Software

Modelización y Simulación de un Sistema de Control para Ensayos de Motores de Combustión Interna

Tesis

***Autor*: Ing. Juan F. Giró**

***Director*: Ing. Oscar Sartori**

Mayo de 2005

A la memoria de mi padre Juan y mi abuelo Francisco,
de quienes aprendí que las cosas importantes,
solo se consiguen con imaginación,
esfuerzo y perseverancia.

A mi Esposa e Hijos.

Agradecimiento

A Oscar Sartori, Director de esta tesis,
y fuente inagotable de iniciativas por hacer cosas,
a quién conocí al ingresar a la facultad de ingeniería,
hace ya muchos años.

Modelización y Simulación de un Sistema de Control para Ensayos de Motores de Combustión Interna

índice de temas

índice de temas (cont.)

Modelización y Simulación de un Sistema de Control para Ensayos de Motores de Combustión Interna

índice de figuras, cuadros y tablas

índice de figuras, cuadros y tablas (cont.)

Modelización y Simulación de un Sistema de Control para Ensayos de Motores de Combustión Interna

Prólogo

Desde hace más de treinta años de vida profesional, me he dedicado a la resolución de problemas de ingeniería a través de sistemas de computación, y más de la mitad de ese tiempo estuve vinculado a los ensayos de motores de combustión interna. Estos sistemas para ensayos de motores están destinados a la realización de pruebas en forma completamente automática, incluyendo la adquisición de datos, monitoreo local y remoto, representación de valores en pantalla, y migración de resultados de ensayos a bases de datos. Como parte de estos sistemas, desarrollé una unidad digital de control que diseñé y perfeccioné progresivamente, alcanzando en todos los casos muy buenos resultados en lo que refiere a estabilidad y rapidez de respuesta para cumplir con las más variadas secuencias de ensayo. Así, he tenido la oportunidad de trabajar en más de cincuenta de estas instalaciones, ya sea en el diseño y construcción de sistemas de ensayo completamente nuevos, como también en la adecuación de instalaciones existentes o su mantenimiento.

Sin embargo, como contrapartida, mis unidades de control han requerido siempre de un ajuste muy laborioso para alcanzar estos buenos desempeños. Debo reconocer que nunca pude igualar la rapidez y facilidad con que son ajustadas las unidades de control analógicas, por lo general de marcas ampliamente reconocidas en el mercado. Ensayé diferentes teorías procurando justificar estas diferencias, aunque no obtuve de ninguna de ellas una respuesta concluyente. Estoy seguro de que mi unidad digital de control tiene margen para introducir variantes que posibiliten una calibración más amigable y la obtención de aun mejores desempeños. Para ello, disponer de un simulador que permita evaluar alternativas de control resultaría de un valor incalculable. En efecto, este simulador facilitaría la exploración de esas alternativas y permitiría, además, entender mejor algunos fenómenos de acoplamiento e interacción de difícil reproducción en instalaciones reales.

Mediante el desarrollo precedente, procuro justificar la elección del tema de la presente tesis, requisito para la finalización de la Maestría de Ingeniería de Software. En este sentido, encontré lógico encaminar mi trabajo de investigación en dirección a los ensayos de motores; con la seguridad de que dicho desarrollo contribuiría, a su vez, a una mejor comprensión de algunos fenómenos que merecen soluciones empíricas, dado que no parecen encontrar, en la teoría, una respuesta apropiada. Al mismo tiempo, el vínculo estrecho existente entre los temas propuestos para la presente tesis y mi actividad académica, profesional y de investigación, me permite confiar en que este esfuerzo contribuirá a un mejor desempeño de mi actividad cotidiana. En relación al área científica de mi especialidad, mantengo la esperanza de que estos aportes contribuyan a una mejor comprensión del fascinante mundo de los sistemas informáticos de tiempo real en general y de los destinados al ensayo de motores en particular.

Así, la *modelización y simulación de un sistema de control para el ensayo de motores de combustión interna* se convirtió en el objeto de este trabajo. Objeto que abordo desde la óptica de la *ingeniería de software.* Sin embargo, a poco de comenzar, descubrí que tal cuestión suponía una complejidad mayor de la esperada; dado que por tratarse de un sistema de tiempo real, las pautas para su tratamiento

no se presentan tan claramente establecidas como ocurre con los sistemas convencionales. En efecto, el análisis y diseño de los sistemas de tiempo real constituye actualmente el foco de numerosas líneas de investigación y promueve el desarrollo de diversas propuestas, algunas de ellas muy originales e inclusive controvertidas. Al mismo tiempo, pude comprobar que este mismo fenómeno se extiende al tratamiento de los sistemas híbridos y a sus aplicaciones en sistemas de control. Todo ello conformó un vasto campo de estudios que encontré apasionante, y por momentos también abrumador, al enfrentarme con tan diversos y variados puntos de vista.

Tal circunstancia me estimuló a ensayar e incorporar aquí mis propios puntos de vista; lo que, en alguna medida, alteró el rumbo que había trazado para este trabajo. Por lo tanto, a la idea inicial de implementar un simulador, fue necesario sumar otros esfuerzos que, en principio, eran insospechados.

En relación al aspecto conceptual, fue necesario proponer un tratamiento unificado del problema del control, integrando conceptos de sistemas híbridos, sistemas conmutados y de la propia teoría de control. En el aspecto metodológico, ensayé una forma de especificar los requerimientos a partir de una combinación de diferentes técnicas tradicionales. Este enfoque, a mi juicio original, demostró ser muy apropiado para tratar los sistemas de tiempo real. Por último, al considerar el propio objeto de estudio que es el ensayo de motores, me apoyé esencialmente en mis propias experiencias y sus resultados. Resulta quizás innecesario destacar que todos los aspectos aquí tratados se apoyaron en una minuciosa y laboriosa búsqueda bibliográfica, procurando el mejor conocimiento posible de los principales antecedentes disponibles en cada caso.

Otro aspecto a destacar es que la diversidad de temas que fueron quedando progresiva e inevitablemente involucrados me enfrentó con el desafío de lograr un adecuado equilibrio en su tratamiento. Por ello, procuré incorporarlos en la medida de lo necesario y, al mismo tiempo, evitar dispersarme en rumbos que, de haber profundizado, me hubiesen impedido delimitar el problema y completar este trabajo.

Considero necesario señalar que desarrollé la totalidad del software aquí utilizado sin recurrir a productos ni librerías de terceros. Tal decisión encuentra su justificación en la consideración no sólo de que es ésta la mejor forma, o quizás la única, de alcanzar un conocimiento profundo de los temas tratados, sino que además es la que me proporcionó mayor libertad para hacer todas las pruebas que parecieron interesantes. Por esta razón, evité recurrir a la utilización de simuladores o otros productos afines, y presentar aquí resultados que pudieron haber sido obtenidos con herramientas CASE. En este último aspecto, la razón que justificó esta decisión fue la necesidad de acotar el alcance del trabajo y darle una dimensión finita, teniendo en cuenta que las herramientas CASE constituyen todo un amplio mundo que merece un tratamiento especial y excede el alcance de este documento.

Este trabajo se integró con elementos provenientes de campos tan diversos como son la mecánica, electrónica, teoría de control, ciencia de la computación, inteligencia artificial, matemáticas, informática y metrología. Los resultados obtenidos, que son aquí reproducidos y comentados, parecerían indicar en primera instancia que se han alcanzado los objetivos planteados.

Juan F. Giró
Mayo de 2005

Modelización y Simulación de un Sistema de Control para Ensayos de Motores de Combustión Interna

1. Introducción

La finalidad de este trabajo es la modelización y simulación de un sistema de control para ensayos de motores de combustión interna. Este modelo computarizado de tiempo real estará destinado a predecir el desempeño de un conjunto motor-freno y a evaluar el comportamiento de su unidad de control. Éste permitirá estudiar condiciones de inestabilidad del sistema, facilitará el ajuste de los parámetros de la unidad de control y hará posible reproducir fenómenos colaterales o respuestas inesperadas. Más aún, la disponibilidad de este modelo permitirá también anticipar el comportamiento del motor en condiciones especiales, reemplazando así en algunos casos a los propios ensayos con un consiguiente ahorro de tiempo y costos.

Si bien la simulación de motores de combustión interna no es nueva, ya que desde los años ’70 se vienen registrando experiencias en este campo (Borman, 1971), es a partir de los ’90 que se produce un gran auge en este tipo de modelos, de la mano de los avances de la tecnología informática y del interés comercial que despiertan. Los objetivos perseguidos con estos modelos son muy diversos, pero tienen en común la necesidad de aportar información adicional a la obtenida con ensayos experimentales (Rubin, 1997). Entre otros beneficios, esto permite reducir el ciclo de desarrollo de nuevos productos, explorar nuevas soluciones y evaluar performances en dominios inaccesibles o en circunstancias en que los ensayos son muy costosos. Más recientemente, son los estudios orientados a la reducción del consumo y las emisiones nocivas de motores los que se han convertido en uno de los principales objetivos de estos modelos (Brace, 2001).

Se identifica así, en el dominio del problema planteado, a un sistema que involucra la respuesta dinámica continua de un conjunto mecánico y la acción de una unidad digital de control que opera a partir del muestreo discreto de los valores reales de velocidad y carga. Más adelante, se describirán las características de este sistema de control y será necesario evaluar un modelo capaz de representar el comportamiento dinámico de un motor acoplado a un freno dinamométrico.

Estos sistemas de control, que tienen la finalidad de operar sobre dominios continuos, dan lugar a un tipo particular de sistema denominado “híbrido”. Estos sistemas híbridos combinan, por lo tanto, componentes discretos y continuos que interactúan entre sí y responden a concepciones de naturaleza diferente, a pesar de que en última instancia el subsistema discreto no es más que una abstracción de la realidad. Estas dos dinámicas coexisten y es necesario representarlas con modelos que reflejen esta interacción, reconociéndose que las mayores dificultades se originan en la necesidad de sincronizar sus comportamientos. Podría afirmarse que en la actualidad los sistemas de control, en su mayoría, pueden ser reconocidos en alguna medida como híbridos ya que en realidad todo sistema de computación que interactúa con el mundo real queda encuadrado en esta categoría.

En este punto cabe destacar que los sistemas de control forman parte de las numerosas aplicaciones de tiempo real que son desarrolladas en la actualidad. Éstas abarcan un amplio espectro de destinos, que incluye desde la industria bélica hasta equipos de uso doméstico, y es indudable que la atención de dicha demanda tiene un fuerte impacto en todos los campos de la informática. Estas aplicaciones se caracterizan tanto por sus particulares exigencias de confiabilidad, eficiencia y rápida respuesta como por la diversidad de disciplinas que involucran. Entre estas últimas, se incluyen los lenguajes de programación, protocolos de comunicaciones, arquitectura del hardware, modelos matemáticos, sistemas operativos y teoría de control.

De esta forma, los sistemas de tiempo real se han incorporado a la creciente demanda de software especializado que se verifica en la actualidad. Además, el abaratamiento del hardware y el incremento de la densidad de integración hace propicia la realización de sistemas cada vez más complejos y es de preverse que esta tendencia se acentúe en el futuro próximo.

Así, la necesidad de dar respuesta a esta demanda no ha hecho más que incorporar nuevas exigencias al gran esfuerzo que se viene realizando para superar definitivamente la llamada "*crisis del software*" y que ha justificado una muy importante inversión de recursos, tanto intelectuales como económicos. Estos esfuerzos han estado orientados a establecer nuevas técnicas, métodos y herramientas que permitan tanto mejorar la calidad de los desarrollos de software como así también a aumentar la productividad de los procesos conducentes a tales desarrollos, aumentando eficiencia y eficacia.

Sin embargo, en la actualidad resulta preocupante comprobar que, en muchos casos, son los intereses comerciales los que orientan el mercado aun en direcciones que se contraponen con lo que indicarían otros puntos de vista, incluido el técnico. Se acentúa una visión simplista e ingenua de que el desarrollo de sistemas complejos se simplifica con la sola utilización de los métodos que han dado buenos resultados en otros campos de aplicación, o de que todo se resuelve incrementando los recursos de hardware.

Coexiste, felizmente, otra realidad determinada por el ya mencionado amplio espectro de buenas técnicas y herramientas de análisis y diseño de sistemas que han sido desarrolladas y están disponibles. El esfuerzo debe ahora orientarse a hacer que trasciendan los ámbitos de los centros de investigación y se incorporen al mercado cotidiano.

De una manera indudablemente muy modesta, este trabajo espera hacer su aporte en este sentido, demostrando que es posible desarrollar e implementar un simulador de tiempo real de una unidad de control a través de un procedimiento sistemático, racional y riguroso, a los fines de asegurar un modelo que sea a la vez correcto, completo y consistente.

A partir de todo lo expuesto, quedan planteados los objetivos del presente trabajo, que son formalmente enunciados a continuación:

1.1 Objetivo general

Desarrollar, implementar y evaluar un simulador de tiempo real de una unidad de control destinada a conducir ensayos de motores de combustión interna en forma completamente automática.

1.2 Objetivos particulares

A partir del objetivo general se enuncian los siguientes cuatro objetivos particulares:

1.2.1. Marco teórico

Al comprobarse la existencia de diversas teorías y/o enfoques que son aplicables al problema planteado, se establece que el primer objetivo particular debe necesariamente ser el siguiente:

Definir un marco teórico apropiado que sirva de respaldo conceptual al modelo y de soporte numérico a su implementación.

1.2.2. Marco metodológico

Habiéndose reconocido que, por sus particularidades, los sistemas de tiempo real en general y los sistemas híbridos de control en particular requieren un tratamiento especial por parte de la ingeniería de software, se establece el segundo objetivo particular de este trabajo:

Definir el marco metodológico necesario que permita modelar e implementar en forma sistemática un modelo correcto, completo y consistente.

1.2.3. Definición e implementación

En concordancia con el objetivo general enunciado y a partir de los resultados de los objetivos anteriores, se establece el tercer objetivo particular:

Diseñar e implementar un modelo apto para simular en tiempo real la acción de un sistema de control sobre el ensayo de un motor.

1.2.4. Revisión

De acuerdo con la última fase del desarrollo de todo sistema, que establece la necesidad de comprobar que éste responde a sus especificaciones y éstas a su vez son correctas, se enuncia el último objetivo particular de este trabajo:

Comprobar las aptitudes y limitaciones del simulador implementado, así como su capacidad para reproducir correctamente fenómenos habituales en ensayos reales de motores.

2. Marco teórico

Se presenta a continuación una formulación teórica de los sistemas híbridos de control que respaldará el modelo a ser desarrollado y se comprueba la necesidad de otras herramientas numéricas, según el siguiente detalle:

- El fenómeno continuo será representado por un sistema de ecuaciones diferenciales ordinarias que debe ser resuelto en el tiempo, para lo cual se selecciona el método de integración numérica apropiado.
- Estas ecuaciones diferenciales incluyen entre sus términos las fuerzas que resultan de las acciones del motor y freno dinamométrico. Para su representación se recurre a redes neuronales artificiales de perceptrones multicapa.
- El sistema de control recibirá del sistema continuo un muestreo discreto y no sincronizado de las variables de estado. Para cumplir las acciones de control será necesario extrapolar y derivar numéricamente estos valores.

Por lo tanto, los *sistemas híbridos de control*, la *integración numérica de ecuaciones diferenciales, la aproximación de funciones con redes neuronales artificiales* y la *diferenciación e interpolación numérica* conforman el marco teórico de este trabajo.

2.1 Sistemas híbridos de control

Los sistemas dinámicos híbridos se caracterizan por la interacción entre procesos continuos y discretos. Se trata de una combinación de sistemas dinámicos continuos o discretos en el tiempo (time-driven) y sistemas dinámicos de eventos discretos SED's (*event-driven*), donde la característica distintiva está dada por su vector de estado conjunto $\boldsymbol{w}$, que contiene variables tanto discretas como continuas. Un sistema será entonces denominado híbrido si su vector de estado contiene al menos una de cada tipo de estas variables, es decir que $\boldsymbol{w} \subset \boldsymbol{R}^n \times \boldsymbol{Z}^m$ donde $n > 0$ y $m > 0$.

2.1.1 Aspectos generales

La necesidad de dar a estos sistemas un tratamiento especial fue reconocida hace más de treinta años por Fahrland, quien es considerado un pionero en este campo (Fahrland, 1970). Sin embargo, por mucho tiempo los sistemas híbridos no ocuparon la atención de los investigadores, hasta que en los últimos diez años estos sistemas comenzaron a ser el objeto de un estudio intensivo, tanto por parte de las comunidades de investigadores de Ciencia de la Computación como por las de la de Teoría del Control. En particular, puede comprobarse que se otorga un particular énfasis al objetivo de alcanzar una representación unificada de los sistemas híbridos que se apoye en fundamentos matemáticos rigurosos (Balluchi, 2000). Sin embargo, debe advertirse que el campo de los sistemas híbridos es extremadamente amplio, incluyendo los problemas de control de sistemas continuos y de control de sistemas discretos como casos particulares. Por lo tanto, es de prever que no será fácil que se llegue a establecer una estrategia común para abordar todos estos casos.

Tradicionalmente, los sistemas híbridos han sido representados por diferentes formas de autómatas, cuyos estados son asociados con ciertas ecuaciones diferenciales y las transiciones son definidas a partir de condiciones impuestas sobre las variables del vector de estado. Las dos principales corrientes para estudiarlos se desarrollaron en torno a las *Redes de Petri* y a los *Autómatas Finitos*.

La teoría de la *Red de Petri* fue desarrollada por Carl Adam Petri, Anatol Holt y otros, siendo el primero de ellos quién posteriormente la presentó en su tesis doctoral. Se trataba de un nuevo modelo, destinado al estudio de flujos de información en sistemas, que es aún hoy exitosamente utilizado para representar fenómenos asincrónicos, concurrencia, paralelismo y sincronización de actividades. Estas *Redes de Petri* fueron luego extendidas para adecuar sus modelos a la representación de fenómenos de *Tiempo Real*, dando lugar a numerosas variantes que se designan bajo la denominación general de *Redes de Petri Temporales* (Murata, 1989). Aproximadamente en esa misma época se introdujeron los modelos denominados *Continuos* (Alla y David, 1987), que forman parte de una evolución en la que se amplió el formalismo de las Redes de Petri para adecuarlas a la representación de Sistemas Híbridos. Se presentaron luego las *Redes de Petri Híbridas* (Le Bail, 1991) y las *Redes de Petri Diferenciales* (Demongodin, 1996 y Champagnat, 1998).

El enfoque alternativo para la representación de modelos de tiempo real surgió con la presentación de los *Autómatas Híbridos* (Alur, 1993 y 1995). Numerosas corrientes, que pueden ser agrupadas en dos clases, se derivaron de aquel trabajo de Alur. En la primera se encuentran los métodos que responden a un formalismo integrador que extiende uno de los dominios -discreto o continuo- con la finalidad de incorporar al otro. En la segunda clase, los que preservan la identidad de cada dominio y concentran la atención en la coordinación entre ellos, conservando el potencial propio de cada subsistema.

La referencia específica al tratamiento de los sistemas híbridos en aplicaciones de control requiere la mención de los trabajos de Vadim Utkin (Utkin, 1977), Raymond DeCarlo (Decarlo, 1988 y 2000), Michael Branicky (Branicky, 1998 y 2000), Panos Antsaklis (Antsaklis, 2000) y Xenofon Koutsoukos (Koutsoukos, 1999 y 2000) por sus importantes aportes en este campo.

En éstos y otros trabajos recientes se ha advertido que las interacciones entre los subsistemas continuos y discretos no es trivial. Más aún, estas interacciones deben ser cuidadosamente estudiadas y representadas para no distorsionar la respuesta del modelo. En efecto, tal como se señaló, estas interfaces posibilitan la comunicación entre los diferentes dominios, generándose eventos a partir de las señales de los procesos continuos y alterándose estos procesos continuos a partir de eventos originados en el controlador. Así, estas interfaces definen particiones en el espacio de estados al dividirlo en regiones que constituyen estados cualitativos del proceso.

Un análisis más específico permite distinguir los siguientes dos casos: 1) Sistemas continuos que muestran cambios abruptos en sus respuestas o en los parámetros involucrados en su comportamiento dinámico, incluyéndose el caso en que comportamientos no lineales son representados por una sucesión de funciones

lineales en intervalos claramente establecidos y 2) Controladores discretos conmutables destinados a operar sobre los subsistemas continuos.

En el primer caso, se presentan sistemas no lineales que son modelados por conjuntos de ecuaciones diferenciales, cada una de las cuales cubre su comportamiento en una cierta región de su espacio de estados. El modelo se completa con las expresiones que definen los límites de estas regiones, es decir, los subdominios en que las ecuaciones diferenciales son aplicables.

El segundo caso corresponde al de controladores conmutables que operan sobre sistemas continuos. La principal razón para optar por controladores de este tipo es la obtención de la mejor performance posible en todo su rango de utilización; lo que implica buena respuesta en todos los regímenes estacionarios y una rápida transición al cambiar de una condición a otra. Además, es esencial un apropiado reconocimiento de su naturaleza híbrida en la obtención de un grado importante de autonomía. Estos sistemas conmutados son sistemas dinámicos híbridos que constan de una familia de subsistemas continuos o discretos en el tiempo y un conjunto de reglas que determinan la conmutación entre ellos. Cabe señalar que, en algunos casos, la conmutación entre controladores es necesaria para asegurar condiciones de estabilidad y que, en otros casos, se utiliza para cumplir eficientemente la secuencia de operaciones requeridas. En estos sistemas, un elemento supervisor selecciona el tipo de control apropiado a partir de la información recibida del proceso y de las señales de control requeridas para atender cada caso. En aquellos casos donde intervienen distintos subsistemas suele diseñarse un autómata híbrido para cada subsistema, incluido el controlador y se obtiene luego mediante una composición paralela un autómata híbrido para el sistema completo.

Obsérvese que los sistemas híbridos de control aparecen con la incorporación de las computadoras en la supervisión de procesos continuos, como es el caso, por ejemplo, de los procesos químicos y de fabricación.

La consideración del problema desde el punto de vista de la modelización, permite señalar que, en muchos sistemas reales, sus principales componentes son claramente identificables. En estos casos la unidad de control, las interfaces y el subsistema controlado encuentran en los sistemas híbridos una representación funcional adecuada para su estudio formal, ya que cada componente está representado de manera natural. Desde ésta óptica puede ser interpretado el esquema de la Figura 1.

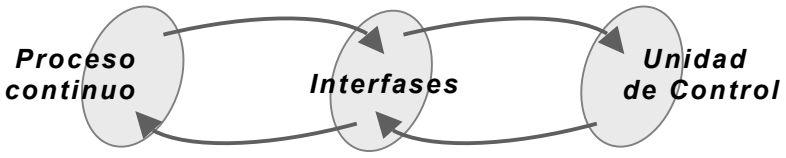


Figura 1 : Sistema híbrido de control

Se presentan otros casos en los que tal separación no es tan clara y esta representación es una abstracción. En estos, el objetivo no suele ser la implementación de una estrategia de control, sino más bien la identificación de sus propiedades y la facilitación de su estudio. Aun en estos casos, una descripción de

este tipo puede ser útil para una mejor comprensión de sus propiedades esenciales, las que están relacionadas con la interacción de las dinámicas continuas y discretas.

El desarrollo presentado permite orientar el estudio hacia el problema planteado, *el control del ensayo de un motor de combustión interna*, concentrándose la atención en la representación del objeto controlado, la unidad de control y sus interfases.

Para ello, se presenta a continuación la formulación teórica que respalda el estudio de estos elementos, para lo cual se tomaron como referencia los lineamientos propuestos en numerosos trabajos por los ya mencionados Michael Branicky (Branicky,1998), Raymond Decarlo (Decarlo,2000) y Xenofon Koutsoukos (Koutsoukos, 1999 y 2000) referidos a la supervisión y control de sistemas hibridos.

2.1.2 Objeto controlado

Puede comprobarse que la tendencia de la ingeniería actual se orienta hacia una creciente complejidad de los sistemas, caracterizada tanto por el elevado número de señales de entrada y salida que deben ser controladas, como así también por la necesidad de contemplar desempeños más exigentes.

En respuesta a estos requerimientos, la teoría de control moderna se está encaminando hacia un tratamiento eficiente de estos sistemas no lineales, que muestran variaciones en el tiempo y disponen de múltiples entradas y salidas. Para ello se utiliza un enfoque que está esencialmente orientado a tratar estos problemas en el dominio temporal.

Estos sistemas de control de múltiples entradas y salidas son denominados *MIMO* (*Multi Input-Multi Output*) y son resueltos por medio de computadoras a través de un modelo que estudia su estabilidad y comportamiento en su espacio de estados. Para ello, se describe el problema en términos de un sistema de ecuaciones diferenciales que es resuelto mediante un enfoque matricial que brinda generalidad a la solución, permitiendo su aplicación a problemas de diversa complejidad.

Esta técnica contrasta con la teoría de control convencional o clásica, que proponía una formulación que sólo era aplicable a sistemas con una única entrada-salida, invariantes en el tiempo y lineales, que era resuelta a través de un tratamiento particular en el dominio de la frecuencia compleja.

Así, el objeto controlado, también denominado “planta” en obvia referencia a los procesos químicos e industriales, es representado por una o varias ecuaciones diferenciales que toman la siguiente forma general:

$$d\boldsymbol{y}(t)/dt = f(\boldsymbol{y}(t), \boldsymbol{z}(t)) \qquad (1)$$

donde :

- $\boldsymbol{y}(t)$ representa el vector de estado de la planta; $\boldsymbol{y}(t) \in \boldsymbol{Y}$ y $\boldsymbol{Y} \subset \boldsymbol{R}^n$
- $\boldsymbol{z}(t)$ es el vector de entrada o de señales de control; $\boldsymbol{z}(t) \in \boldsymbol{Z}$ y $\boldsymbol{Z} \subset \boldsymbol{R}^m$.
- t está definido en cierto intervalo de tiempo; $t \in [a, b]$

donde $\boldsymbol{R}$ denota a los números reales y $\boldsymbol{R}^n$ representa el espacio vectorial de un vector real de *n* dimensiones.

Por cada valor de $\boldsymbol{z}(t)$ y para cada condición inicial $\boldsymbol{y}_0$, la función:

$$f(\bullet, \boldsymbol{z}(t)): \boldsymbol{Y} \rightarrow \boldsymbol{Y} \qquad (2)$$

es continua en $\boldsymbol{Y}$, tiene solución y esta solución es única. Obsérvese que en general esta función es no lineal y es además invariante en el tiempo.

La expresión (1) puede ser rescrita, para un cierto conjunto de valores de las señales de entrada $\boldsymbol{z}(t)$, en la forma general:

$$d\boldsymbol{y}(t)/dt = \mathbf{A}\, \boldsymbol{y}(t) \qquad (3)$$

dando lugar a las siguientes definiciones de estabilidad:

- *Estabilidad en el sentido de Lyapunov*: cuando toda condición inicial finita origina una trayectoria acotada.
- *Estabilidad asintótica*: cuando toda condición inicial finita origina una trayectoria acotada y se cumple además que $lim\ \boldsymbol{y}(t) \rightarrow 0$ para $t \rightarrow \infty$. Esto significa que todos los autovalores de la matriz **A** tienen su parte real negativa y en este caso se dice que la matriz **A** es Hurwitz.
- *Teorema de Lyapunov*: una matriz **A** es Hurwitz si y sólo si, dada cualquier matriz simétrica y definida positiva **W**, la ecuación de Lyapunov:

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{W} \qquad (4)$$

  tiene una solución única **P**, que es también simétrica y definida positiva.

Considerando el vector **y** en un cierto instante t, puede definirse un escalar V tal que:

$$V(\mathbf{y}) = \mathbf{y}^T\mathbf{P}\mathbf{y} \qquad (5)$$

$$V'(\mathbf{y}) \leq \mathbf{y}^T\mathbf{W}\mathbf{y} \qquad (6)$$

Este escalar V, así definido, está asociado a la energía del sistema y es denominado función de Lyapunov, representando una condición necesaria pero no suficiente de estabilidad.

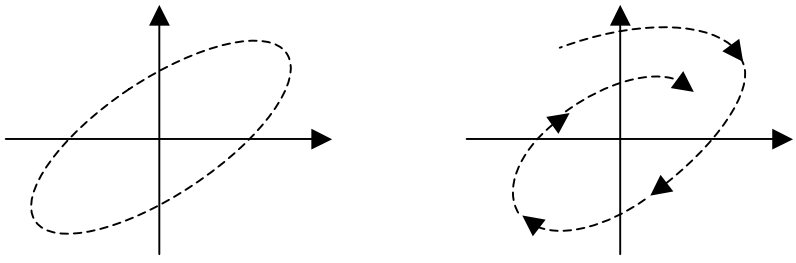


Figura 2: Trayectoria estable de Lyapunov y trayectoria estable asintótica

Hay algunos casos en que no hay una única función continua capaz de representar el comportamiento del sistema en toda la región de interés y debe recurrirse a varias expresiones tales como la siguiente:

$$d\boldsymbol{y}(t)/dt = \mathbf{A}_q\, \boldsymbol{y}(t) \qquad (7)$$

es decir que:

$$d\boldsymbol{y}(t)/dt = \begin{cases} \mathbf{A}_1\ \boldsymbol{y}(t) & ;\ t_a \le t < t_1 \\ \mathbf{A}_2\ \boldsymbol{y}(t) & ;\ t_1 \le t < t_2 \\ \cdots\cdots\cdots & \\ \mathbf{A}_m\ \boldsymbol{y}(t) & ;\ t_{m-1} \le t < t_b \end{cases} \qquad (8)$$

donde el rango de aplicación de cada matriz $\mathbf{A}_q$ debe estar perfectamente definido.

Se dice que se "*conmuta"* de una expresión a otra para representar la función en cada intervalo y a partir de esta composición de funciones surgen naturalmente los siguientes dos interrogantes:

a) ¿Qué sistemas admiten cualquier secuencia de conmutación, manteniendo una trayectoria estable?

b) ¿Qué secuencias de conmutación dan siempre lugar a trayectorias estables?

Las respuestas a estas preguntas están relacionadas con que cada matriz $\mathbf{A}_q$ cumpla individualmente con las condiciones de estabilidad y que las conmutaciones en los límites de cada intervalo sean suficientemente lentas. De esta manera, la garantía de estabilidad del sistema queda supeditada a las condiciones de conmutación, lo que puede ser demostrado a partir del comportamiento de la función de Lyapunov y otras consideraciones energéticas (Decarlo, 2000).

Considerando ahora los factores que intervienen en la selección de $\mathbf{A}_q$, se dice que un sistema es "*autónomo"* si la secuencia de conmutación sólo depende del tiempo y/o del propio proceso. En estos casos:

$$d\boldsymbol{y}(t)/dt = f(\boldsymbol{y}(t), q(t)) = \mathbf{A}_q\ \boldsymbol{y}(t) \qquad (9)$$

y

$$q(t+1) = \xi(\boldsymbol{y}(t), q(t)) \qquad (10)$$

donde $q(t) \in \mathbf{Q}$ es un conjunto finito que identifica las opciones de conmutación disponibles y ξ representa una función de transición, tal que

$$\xi : \boldsymbol{Y} \times \mathbf{Q} \rightarrow \mathbf{Q} \qquad (11)$$

Por el contrario, al sistema se lo denomina "*controlado*" si en la selección de $\mathbf{A}_q$ interviene también algún proceso externo representado por $\boldsymbol{z}(t)$. En estos casos:

$$d\boldsymbol{y}(t)/dt = f(\boldsymbol{y}(t), \boldsymbol{z}(t), q(t)) = \mathbf{A}_q\ \boldsymbol{y}(t) \qquad (12)$$

y

$$q(t+1) = \xi(\boldsymbol{y}(t), \boldsymbol{z}(t), q(t)) \qquad (13)$$

donde

$$\xi : \boldsymbol{Y} \times \boldsymbol{Z} \times \mathbf{Q} \rightarrow \mathbf{Q} \qquad (14)$$

Considerando que un cierto vector **x** representa los valores que deben ser alcanzados por el vector de estado **y**, se definirá al sistema como estable si

$$\mathbf{y}(t) \rightarrow \mathbf{x} \quad \text{para} \quad t \rightarrow \infty \qquad (15)$$

y en cada instante el error queda definido como:

$$\mathbf{e}(t) = \mathbf{y}(t) - \mathbf{x} \qquad (16)$$

Luego, se denomina "*realimentado*" a todo sistema "*controlado*" tal que

$$\boldsymbol{z}(t) = g(\boldsymbol{y}(t), \boldsymbol{e}(t)) \qquad (17)$$

donde *g* representa la acción de un sistema externo que tiene la finalidad de hacer mínimo el error ($\boldsymbol{e}(t) \rightarrow 0$).

Obsérvese que el caso presentado inicialmente en la ecuación (1) es un caso particular de sistema "controlado", que al quedar definido en todo el intervalo por una única expresión, la acción de control está sólo a cargo de un proceso externo que define al vector $\boldsymbol{z}$. Estos sistemas son denominados "*directamente controlados*".

Como se observará más adelante, el problema del control del conjunto motor-freno quedará encuadrado en lo que fue denominado un "sistema realimentado", pero con características muy especiales. En efecto, éste será representado por un sistema de ecuaciones diferenciales con términos que varían en forma continua en función tanto del vector de estado como del vector de señales de control. Se asume así:

$$d\boldsymbol{y}(t)/dt = \mathbf{A}\ \boldsymbol{y}(t) \qquad (18)$$

donde $$\mathbf{A} = h(\boldsymbol{y}(t), \boldsymbol{z}(t)) \qquad (19)$$

y la función *h* es intrínseca al propio sistema, por lo que debe notarse que no se cumple aquí la premisa de que el movimiento queda definido por una función invariante en el tiempo.

2.1.3 Control conmutado

Los sistemas conmutados de control tienen un importante antecedente en el trabajo de Vadim Utkin (Utkin, 1977), en el que presenta un detallado análisis de lo que denomina "*sistemas de estructura variable*". La propuesta consiste en alterar la estructura de la lógica de control de manera de asegurar, en el espacio de estados, una trayectoria compatible con ciertas condiciones que aseguren su precisión y estabilidad. Las hipersuperficies definidas en el espacio de estados por estas condiciones son denominadas "*superficies de deslizamiento*" y la intersección de todas estas superficies definen una hipertrayectoria, denominada "*trayectoria de deslizamiento*", presentada en la Figura 3. Posteriormente, Raymond DeCarlo (DeCarlo, 1988) realiza importantes aportes a estos "sistemas de estructura variable", que son los antecesores de los hoy denominados "*sistemas conmutados*" (DeCarlo, 2000).

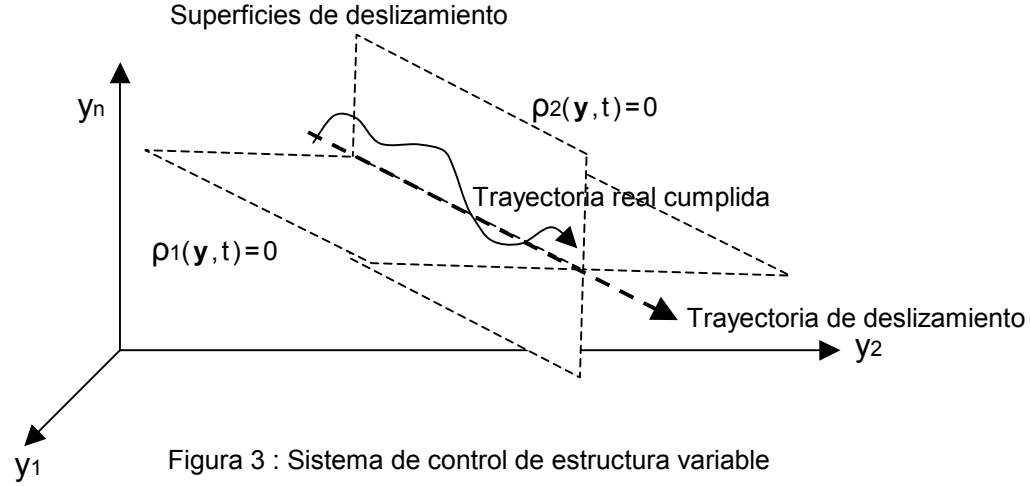


Figura 3 : Sistema de control de estructura variable

Una unidad de control o "*supervisor*" conmutado cumple las siguientes dos funciones principales:

a. *Definición de las señales de control*: A partir de los valores deseados **x** y de las condiciones de la Planta, representadas por su vector de estado **y**, deben determinarse las señales de control apropiadas **z**, tal como lo expresa la ecuación (17). El objetivo es que el vector error **e** alcance un valor mínimo con la mayor rapidez posible y éste error se mantenga estacionario.

b. *Selección de la lógica de control*: Para cumplir este objetivo se dispone de diversas opciones y en cada caso debe seleccionarse la más conveniente. Se trata por ello de un *sistema conmutado de control*.

En la Figura 4, se representan la unidad conmutada de control y el objeto controlado o "planta", que en este caso quedará representado por el conjunto motor-freno:

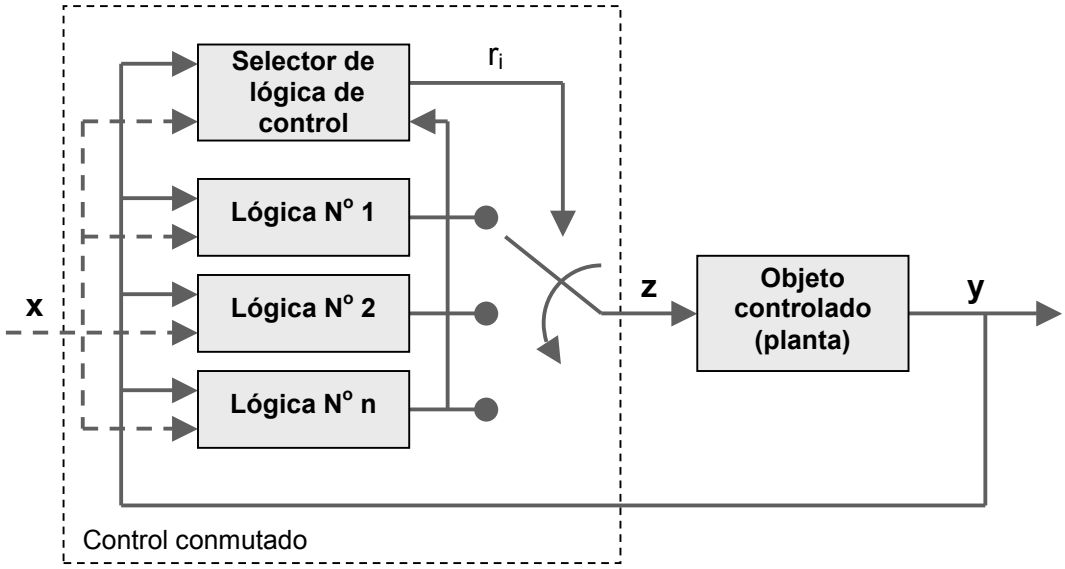


Figura 4: Control conmutado y objeto (planta)

El selector de lógica de control es un sistema de eventos discretos, asincrónico, representado por una *máquina secuencial de tipo traductora* que es definida como una quíntupla a través de sus componentes:

$$Uc = (\boldsymbol{S}, \boldsymbol{\Sigma}, \boldsymbol{R}, \delta, \phi) \qquad (20)$$

donde :

- $\boldsymbol{S}$: Conjunto finito de estados
- $\boldsymbol{\Sigma}$: Alfabeto de símbolos de entrada, que representan las condiciones de la "planta".
- $\boldsymbol{R}$: Alfabeto de símbolos de salida, que representan las opciones de selección de las lógicas de control disponibles.
- δ : Función de transición, definida como $\delta: \boldsymbol{S} \times \boldsymbol{\Sigma} \rightarrow \boldsymbol{S}$
- ϕ : Función de salida, que puede quedar definida como $\phi: \boldsymbol{S} \times \boldsymbol{\Sigma} \rightarrow \boldsymbol{R}$ o $\phi: \boldsymbol{S} \rightarrow \boldsymbol{R}$, según las exigencias de la aplicación.

Obsérvese que en algunos casos la literatura atribuye a está máquina la condición de Autómata Finito, cuando en realidad se trata de una máquina secuencial de Mealy o de Moore, según la función de salida adoptada. Sin embargo, estas máquinas secuenciales se convierten en Autómatas Finitos cuando es necesario tener identificadas en el sistema de control las condiciones de puesta en marcha y detención de la Planta. Estas condiciones son representadas por los estados S_o y $\mathbf{S}_F$ de la unidad de control, que a su vez tendrán una relación biunívoca con los vectores de estado de la planta en esas condiciones. El autómata finito quedará ahora expresado como una séptupla:

$$Uc = (\boldsymbol{S} , \boldsymbol{\Sigma} , \boldsymbol{R} , S_o , \mathbf{S}_F , \delta , \phi) \qquad (21)$$

donde se incorporan los componentes :

S_o : Estado inicial, donde $S_o \in \boldsymbol{S}$

$\boldsymbol{S}_F$: Condiciones de detención, donde $\boldsymbol{S}_F \subset \boldsymbol{S}$

En un cierto intervalo discreto de tiempo “t” las acciones del controlador quedan entonces definidas por las expresiones:

$$r[t] = \phi (s[t], \sigma[t]) \qquad (22)$$

o por

$$r[t] = \phi (s[t]) \qquad (23)$$

según sea que la salida del autómata finito corresponda a un modelo de Mealy o Moore. En cualquier caso, la conducta del autómata queda definida por su función de transición, que permite determinar su próximo estado:

$$s[t+1] = \delta (s[t], \sigma[t]) \qquad (24)$$

donde $s[t], s[t+1] \in \boldsymbol{S}$, $\sigma[t] \in \boldsymbol{\Sigma}$ y $r[t] \in \boldsymbol{R}$.

El determinismo de la máquina secuencial o del autómata finito que queda definido por δ es un aspecto que debe ser considerado, ya que siempre existirá la posibilidad de que en cierta condición la unidad de control admita más de un próximo estado.

En este caso δ representa una relación de transición tal como la siguiente:

$$\delta : \boldsymbol{S} \times \boldsymbol{\Sigma} \rightarrow \boldsymbol{P} (\boldsymbol{S}) \qquad (25)$$

donde $\boldsymbol{P}$ representa subconjuntos de estados o elementos de $\boldsymbol{S}$. Para el caso estudiado se asumirá que, asignando niveles de prioridad a cada posible condición de transición, puede siempre ser evitado el no determinismo del autómata.

2.1.4 Interfases

El controlador y el objeto controlado (Planta) pertenecen a diferentes dominios; lo que impide su comunicación en forma directa y constituye la esencia de los sistemas híbridos. Para superar esta incomunicación y tal como será presentado en detalle más adelante, debe recurrirse a interfases que se materializan a través de sensores, actuadores, conversores A/D y conversores D/A. También debe anticiparse que, en el sistema real, ciertas condiciones del vector de estado **y** del

objeto controlado serán reconocidos como *eventos* por la unidad de control, es decir que:

$$\chi^{y} = \mathbf{f}^{y}{}_{\chi}(\mathbf{y}) \qquad (26)$$

donde χ^{y} representa el conjunto de todos los eventos reconocidos. Resulta obvio que debe haber una función biunívoca entre estos eventos y el alfabeto de los símbolos de entrada $\boldsymbol{\Sigma}$ del selector de lógicas de control, por lo que se obtiene que:

$$\sigma[t] = \psi(\mathbf{f}^{y}{}_{\chi}(\mathbf{y}), t) \qquad (27)$$

Es muy importante advertir que, en el modelo, en idénticas condiciones deben generarse los mismos eventos que se generarían en el sistema real, de manera de dar lugar a los mismos símbolos de entrada del alfabeto $\boldsymbol{\Sigma}$. Esto significa que:

$$\sigma[t] = \psi(\mathbf{f}^{u}{}_{\chi}(\mathbf{u}), t) \qquad (28)$$

donde **u** es el vector de estados que en el modelo discreto representa al vector de estados **y** del sistema real.

Las funciones $\mathbf{f}_{\chi}$ pueden ser representadas como hipersuperficies que dividen el espacio de estados en regiones disjuntas y los eventos se originan cada vez que la trayectoria definida por el vector de estado cruza una de estas hipersuperficies (Figura 5). La región central representa todas las condiciones de operación normal del conjunto motor-freno, que es cruzada por las trayectorias de deslizamiento ya presentadas en la Figura 3. Obsérvese que esta región de operación normal es redefinida cuando se verifiquen cambios en las condiciones de operación del sistema controlado.

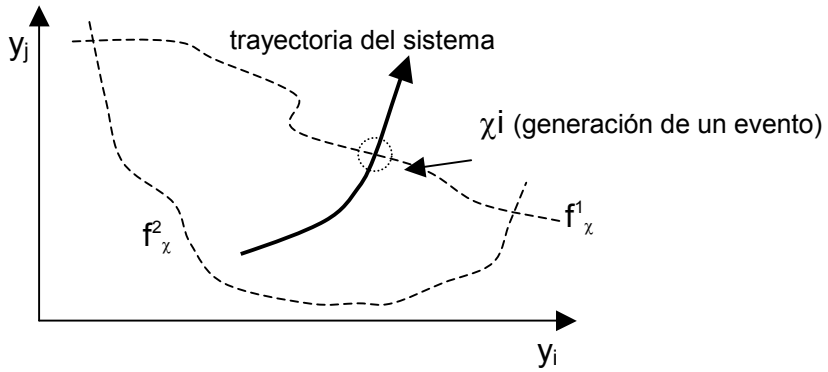


Figura 5: Región de operación normal y generación de un evento

2.1.5 Lógicas de control

Se concentrará ahora la atención en las propias lógicas de control, cuya finalidad es generar las señales **z** destinadas a reducir y mantener estable el error representado por el vector **e**, tal como fue expresado por la Ecuación 17.

Con este fin, en la mayoría de los sistemas industriales de control, y en particular en el ensayo de motores, se implementan lógicas denominadas *PID* - Proporcional, Integral y Derivativa. Los conceptos asociados a este tipo de control tienen antecedentes muy remotos, pero no fue hasta el trabajo de Minorsky (1922) referido a la conducción de barcos, que el control *PID* cobró verdadera importancia teórica. En la actualidad, el controlador *PID* es aún el más utilizado en la industria moderna, a pesar de la abundancia de sofisticadas herramientas y métodos

avanzados alternativos de control. Es probable que esta gran difusión y popularidad en aplicaciones industriales, de la más variada naturaleza, se deba a la simplicidad, versatilidad y eficacia que las unidades de control *PID* han demostrado (Creus, 1987 y Goodwin, 2001).

En la Figura 6, se representa un circuito realimentado básico que corresponde a una de las opciones de conmutación del sistema ya presentado en la Figura 4:

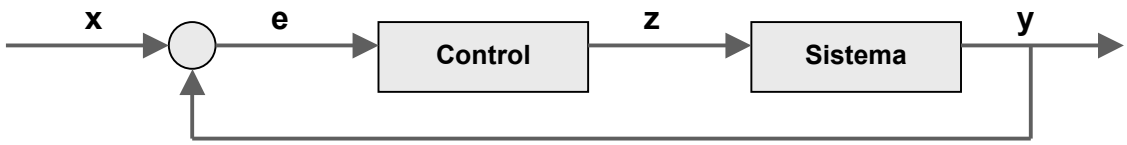


Figura 6: Esquema general de Control

La función de transferencia G queda definida como

$$G(t) = y(t) / z(t) \quad (29)$$

y el error es $$e(t) = x(t) - y(t) \quad (30)$$

luego $$e(t) = x(t) - G(t).z(t) \quad (31)$$

donde $$z(t) = Kp\ e(t) + (Kp / Tr) \int_t e(t)\,dt + Kp\ Td\ de/dt \quad (32)$$

En la expresión (32), se observan tres términos que hacen que la señal de control “z” incluya componentes proporcionales al error (P), proporcionales a la integración del error en el tiempo (I) y finalmente proporcionales a su derivada (D), que se definen a continuación:

Acción proporcional (P): la acción de control está directamente relacionada con el error instantáneo medido. Esta acción proporcional puede controlar cualquier sistema estable pero no es capaz de asegurar que la respuesta estará libre de errores estacionarios.

Acción integradora (I): la acción de control es proporcional al error acumulado en un cierto intervalo de tiempo, lo que implica una acción relativamente lenta. Como contrapartida, tiene la ventaja de asegurar un error estacionario nulo.

Acción derivativa (D): su acción es proporcional a la rapidez con que cambia el error y por esta razón su efecto suele ser definido como predictivo. En efecto, esta acción modera o revierte la acción proporcional cuando todavía no se alcanzó el nivel de referencia deseado pero el ritmo de aproximación es demasiado rápido. De esta forma, se "adelanta" la acción de la variable de control para obtener así una variable de proceso más estable. Esta acción derivativa es deno,imada a veces "rate action" por algunos fabricantes de controles porque considera la "razón de cambio" en la variable de proceso. Su mayor limitación es su tendencia a generar ajustes de elevada magnitud en respuesta a errores que cambian con mucha rapidez (Voda y Landau, 1995).

La lógica de control PID puede ser representada con un esquema como el presentado a continuación en la Figura 7:

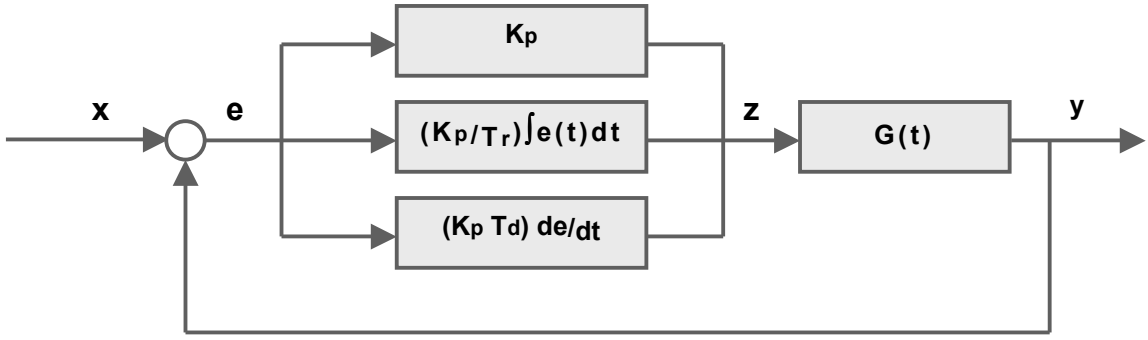


Figura 7: Esquema de Control PID

El ya señalado inconveniente que presenta la acción derivativa por su tendencia a generar ajustes de elevada magnitud se pone de manifiesto ante los cambios bruscos en los elementos del vector **x** (set-point), casos que pueden ser asimilados a una entrada en escalón.

Una solución interesante para este problema se obtiene al trasladar el componente derivativo, de manera que sea calculado sobre la variable de salida del proceso (realimentada) y no sobre su error. Esta lógica de control es denominada PI-D y es una alternativa cierta cuando los cambios requeridos en las condiciones de operación son muy bruscos. En la Figura 8, se presenta un esquema de un sistema de este tipo.

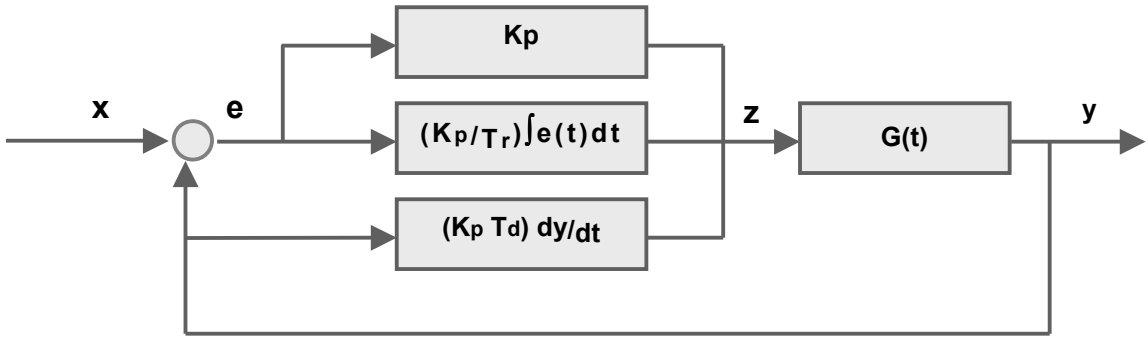


Figura 8 : Esquema de Control PI-D

En base a ambas soluciones, surgió la idea de una unidad de control más general que incluya ambas componentes derivativos, es decir, que ajuste la variable de control tanto a partir de la realimentación de la variable de proceso (salida del objeto controlado) como también a partir de la realimentación de su error.

Este tipo de solución ofrece la posibilidad de regular en forma dinámica la incidencia de uno y otro componente de control, con el fin de lograr transiciones suaves y condiciones estacionarias más precisas. Obsérvese que el selector de lógicas de control (Figura 4) es el encargado de conmutar entre las diferentes opciones disponibles, incluyendo las opciones de variables de control sobre las que se actúa en cada instante y las realimentaciones que se consideran en cada caso. Un esquema de un sistema de este tipo se presenta en la Figura 9.

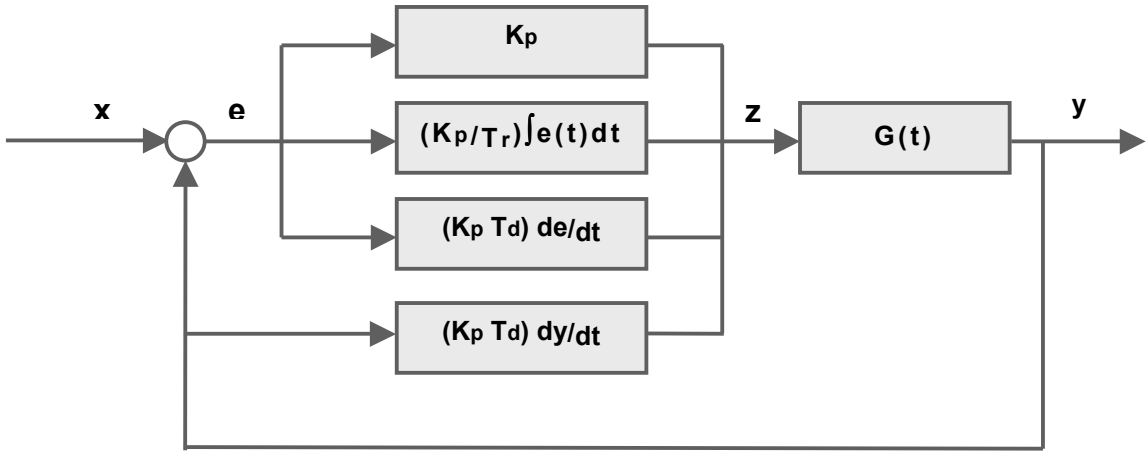


Figura 9 : Esquema de Control PID-D

En particular, tanto los controles *PID* como sus variantes son apropiados para los casos en que se desconoce la función de transferencia de la Planta, ya que pueden ser ajustados a partir de técnicas que se apoyan en resultados experimentales. Los métodos más usados son los de Ziegler-Nichols (Ziegler,1942) y de Cohen-Coon. El primero permite ajustar los parámetros de la unidad de control a partir de la respuesta del sistema a lazo abierto y el segundo lo hace considerando la respuesta a una función escalón.

2.2 Integración de las ecuaciones del movimiento

Los modelos matemáticos apropiados para representar el comportamiento dinámico de elementos mecánicos quedan expresados por sistemas de ecuaciones diferenciales, denominadas *ODE*. En efecto, la teoría de las ecuaciones diferenciales permite estudiar los más diversos procesos dinámicos que cumplen con tres condiciones que forman la base de la mecánica de los sistemas discretos y son las siguientes:

a. *Que estén determinados*: un proceso se denomina “determinado” si su estado pasado y futuro puede obtenerse a partir de su estado presente. Aquí cobran importancia los teoremas de existencia y unicidad, que postulan la existencia de una solución y que tal solución es única.

b. *Que tengan dimensión finita*: el conjunto de todos los estados del proceso conforma su espacio de estados y se denomina de dimensión finita si lo es su espacio de estados.

c. *Que sean diferenciables*: un proceso se llama diferenciable si sus estados tienen estructura de variedad diferencial.

A este problema se lo denominado "problema de Cauchy" o de “valor inicial” y es habitualmente resuelto en forma numérica, ya que estas ecuaciones pueden no tener una solución analítica conocida o no resultar práctica su determinación. Más aún, en muchos casos los métodos numéricos representan la única alternativa posible para la resolución de ciertos problemas analíticamente intratables, que

normalmente representan fenómenos no-lineales.

Se presenta además otra razón que justifica la amplia difusión que han alcanzado estos métodos numéricos y está referida a la disponibilidad de medios apropiados de cálculo. En efecto, es necesario reconocer que, como ocurrió en todos los campos del conocimiento, los métodos de integración numérica han estado por mucho tiempo postergados hasta que el advenimiento del computador digital y su rápido desarrollo los trajo nuevamente al centro de la escena.

En su forma básica más general, el problema de valor inicial ya fue expresado en la Ecuación 3 y puede ser planteado como:

$$y' = f(y,t) \quad ; \quad \text{donde } y(t_o) = y_o \qquad (33)$$

donde $f(y,t)$ es una función no lineal cualquiera y el objetivo es encontrar los valores de la variable dependiente "y" en una secuencia finita de valores de la variable independiente "t", dentro de un intervalo $[t_o, t_F]$ de interés de la solución. El intervalo de tiempo entre dos instantes consecutivos de la solución, denominado usualmente "*paso de integración Δt*", puede permanecer constante sobre un determinado número de intervalos de tiempo, o ser variado cuando consideraciones referidas al error obtenido lo hagan aconsejable.

Para resolver ecuaciones diferenciales existe una gran cantidad de métodos numéricos (Chapra,2004) que permiten obtener soluciones aproximadas de buena calidad. Por ello, la elección del algoritmo apropiado queda determinada por varios factores, entre los que pueden mencionarse los requerimientos de precisión, estabilidad y velocidad de cálculo, como así también la capacidad de proceso disponible. Esta elección requiere entonces de la evaluación del algoritmo y se apoyará en aspectos tales como:

a. La magnitud del error cometido en cada paso del cálculo y la forma en que este error se propaga a los pasos siguientes.
b. La habilidad del método para estimar el error en una etapa de cálculo, en función de los resultados obtenidos.

2.2.1 Sistemas rígidos

Sin embargo, el problema a ser resuelto en este trabajo posee una característica particular que debe ser considerada al momento de realizarse la selección del algoritmo. Las ecuaciones diferenciales a ser integradas corresponden a un sistema dinámico que exhibe tanto una respuesta de cuerpo rígido como también oscilaciones que son consecuencia de sus propiedades elásticas. Debe observarse que mientras el movimiento principal puede ser estacionario u oscilar a baja frecuencia, las vibraciones pueden incluir armónicas de frecuencias elevadas. Inclusive, puede no sólo presentarse el caso en que ambas dinámicas estén presentes en forma simultánea, sino también en que en ciertos instantes de tiempo actúe sólo la componente rápida y en ciertos instantes de tiempo la componente lenta.

Estos sistemas cuya respuesta incluye componentes con muy diversas constantes de tiempo son denominados "*rígidos*" (*stiff systems*) por los matemáticos y su estudio ha despertado mucho interés en los últimos años por las dificultades

que se presentan en los procesos de integración numérica. En efecto, si se considera la componente dinámica rápida se requiere un paso de integración pequeño para garantizar la estabilidad y exactitud de la solución de las ecuaciones diferenciales. Sin embargo, este paso puede ser demasiado pequeño para la solución de la componente dinámica lenta, y si se lo mantiene constante, llevará a extender innecesariamente la obtención de la solución en el intervalo de interés.

Para estos casos en que se presentan constantes de tiempo muy diferentes, son recomendables los métodos de paso variable, los cuales pueden ajustar el paso de integración de acuerdo a los requerimientos del sistema. Sin embargo, debe observarse que esta idea de utilizar un paso diferente según la naturaleza de la respuesta del sistema tiene un costo extra por la necesidad de efectuar evaluaciones y comparaciones adicionales para determinar su valor apropiado.

Métodos muy usados en la práctica para tratar estos casos son aquellos de pasos múltiples o multipaso de orden superior, tales como es el caso del método predictor-corrector de Milne o el método de Adams. Sin embargo, se ha comprobado que aún estas fórmulas tienen sus limitaciones y no son adecuadas cuando se trata de integrar algunos problemas particulares. Entre éstos, puede citarse el caso en que la matriz Jacobiana tenga algún autovalor con parte imaginaria grande (Alvarez Lopez,2002). En un intento de superar estas dificultades en los últimos años se han introducido algunos métodos más generales que combinan técnicas multipaso y de Runge-Kutta.

No obstante todo lo expuesto, la alternativa de implementar en base al método Runge-Kutta tanto procedimientos de paso variable como de paso fijo demostraron en general muy buenos resultados y son las opciones seleccionadas para este trabajo.

2.2.2 Métodos de paso fijo

Los métodos más utilizados para la integración numérica de ecuaciones diferenciales son los denominados métodos de *Runge-Kutta* de un paso*,* que se caracterizan por (Alfaro,2004)*:*

a. Ser auto-iniciables
b. Requerir sólo información del punto anterior en el proceso de integración.
c. Evaluar en cada iteración la función tantas veces como sea el orden del método.
d. No poseer la forma de estimar el error cometido.

La base de todos los métodos del tipo Runge-Kutta es expresar la diferencia entre los valores de ***y*** evaluados en los puntos t_{n+1} y t_n como:

$$y_{n+1} - y_n = \sum_{i=1}^{m} \gamma_i k_i \tag{34}$$

donde

$$k_i = \Delta t\, f\left(y_n + \sum_{j=1}^{i-1} \beta_{ij} k_j,\ t_n + \alpha_i \Delta t \right) \tag{35}$$

en las que las constantes **α, β y δ** deben ser determinadas para cada caso. Obsérvese que estas constantes se obtienen al igualar las expresiones con los primeros términos del desarrollo en serie de Taylor de la función integrada.

Estos métodos son una evolución del método de Euler, al que Heun (1900) le incorporó dos pasos adicionales y luego Kutta (1910) formuló el esquema general tal como hoy se conoce. Sin embargo, Kutta consideró que las expresiones de orden superior eran demasiado complicadas y no las incluyó al publicar sus trabajos.

El método de 4to orden, que es denominado *Runge-Kutta clásico*, brindará resultados con un error que resultará del orden de Δt^5 con un paso de integración Δt fijo, y sus expresiones son las siguientes:

$$k_1 = \Delta t\, f(y_n, t_n)$$

$$k_2 = \Delta t\, f\left(y_n + \frac{1}{2}k_1,\ t_n + \frac{1}{2}\Delta t\right)$$

$$k_3 = \Delta t\, f\left(y_n + \frac{1}{2}k_2,\ t_n + \frac{1}{2}\Delta t\right) \qquad (36)$$

$$k_4 = \Delta t\, f(y_n + k_3,\ t_n + \Delta t)$$

$$y_{n+1} = y_n + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

Una variante es el método de *Runge-Kutta-Simpson* que, siendo también de 4to orden y con un error igualmente del orden de Δt^5, responde a estas expresiones:

$$k_1 = \Delta t\, f(y_n, t_n)$$

$$k_2 = \Delta t\, f\left(y_n + \frac{1}{3}k_1,\ t_n + \frac{1}{3}\Delta t\right)$$

$$k_3 = \Delta t\, f\left(y_n - \frac{1}{3}k_1 + k_2,\ t_n + \frac{2}{3}\Delta t\right) \qquad (37)$$

$$k_4 = \Delta t\, f(y_n + k_1 - k_2 + k_3,\ t_n + \Delta t)$$

$$y_{n+1} = y_n + \frac{1}{8}[k_1 + 3k_2 + 3k_3 + k_4]$$

Se dispone de otras opciones, como son las implementaciones del método de Runge-Kutta de 5to orden. Entre ellas se destaca una que es atribuida a Butcher, de muy buen comportamiento, que presenta la formulación siguiente :

$$y_{n+1} = y_n + (7.k_1 + 32.k_3 + 12.k_4 + 32.k_5 + 7.k_6) / 90 \qquad (38)$$

donde:

$$k_1 = \Delta t.f(y_m, t_m)$$

$$k_2 = \Delta t.f(y_m + (1/4).k_1 , t_m + (1/4). \Delta t)$$

$$k_3 = \Delta t.f(y_m + (1/8).k_1 + (1/8).k_2 , t_m + (1/4). \Delta t)$$

$$k_4 = \Delta t.f(y_m - (1/2).k_2 + k_3, t_m + (1/2). \Delta t) \quad (39)$$

$$k_5 = \Delta t.f(y_m + (3/16).k_1 + (9/16).k_4, t_m + (3/4). \Delta t)$$

$$k_6 = \Delta t.f(y_m - (3/7).k_1 + (2/7).k_2 + (12/7).k_3 - (12/7).k_4 + (8/7).k_5, t_m + \Delta t)$$

2.2.3 Métodos de paso variable

Tal como se mencionó anteriormente, los métodos del tipo Runge-Kutta no tienen forma de evaluar el error cometido en cada iteración. Para superar esta limitación se deben combinar dos métodos de orden diferente o utilizar un mismo método con dos pasos de integración diferentes. De este modo, se puede estimar el error, y en base a su valor, decidir si es necesario o no cambiar el tamaño del paso de integración (reducirlo o incrementarlo), obteniéndose así un método de paso variable.

En el presente trabajo se adoptó el llamado método de Runge-Kutta-Fehlberg, en el que se combinan dos métodos de diferente orden que permiten estimar el error cometido en cada paso de integración. En este caso se usa un Runge-Kutta de 4^{to} y otro de 5^{to} orden, con lo que se obtiene el llamado método de *paso variable de orden 4-5*, que se desarrolla a continuación:

$$k_1 = \Delta t.f(y_m, t_m)$$

$$k_2 = \Delta t.f(y_m + a_{21}.k_1 , t_m + b_2. \Delta t)$$

$$k_3 = \Delta t.f(y_m + a_{31}.k_1 + a_{32}.k_2 , t_m + b_3. \Delta t)$$

$$k_4 = \Delta t.f(y_m + a_{41}.k_1 + a_{42}.k_2 + a_{43}.k_3, t_m + b_4. \Delta t) \quad (40)$$

$$k_5 = \Delta t.f(y_m + a_{51}.k_1 + a_{52}.k_2 + a_{53}.k_3 + a_{54}.k_4, t_m + b_5. \Delta t)$$

$$k_6 = \Delta t.f(y_m + a_{61}.k_1 + a_{62}.k_2 + a_{63}.k_3 + a_{64}.k_4 + a_{65}.k_5, t_m + b_6. \Delta t)$$

A partir de estas constantes, puede plantearse una expresión de 4^{to} orden para evaluar la función, que resulta:

$$y^W_{n+1} = y^W_n + (c_1.k_1 + c_2.k_2 + c_3.k_3 + c_4.k_4 + c_5.k_5) \quad (41)$$

y la expresión de 5^{to} orden para evaluar la misma función toma la forma:

$$y^Z_{n+1} = y^Z_n + (d_1.k_1 + d_2.k_2 + d_3.k_3 + d_4.k_4 + d_5.k_5 + d_6.k_6) \quad (42)$$

El cálculo del error se realiza a partir de las dos expresiones anteriores:

$$\varepsilon = y^Z - y^W = (e_1.k_1 + e_2.k_2 + e_3.k_3 + e_4.k_4 + e_5.k_5 + e_6.k_6) \quad (43)$$

Puede considerarse que este error es proporcional al orden del método más bajo, luego:

$$\varepsilon = K.\ (\Delta t)^5 \qquad (44)$$

y para estar dentro del error máximo admitido (ε_{max}) es necesario introducir un ajuste en el paso de cálculo que queda definido por el factor η, luego:

$$\varepsilon_{max} = K.\ (\eta\ \Delta t)^5 \qquad (44')$$

A partir de estas expresiones y de criterios empíricos destinados a tener en cuenta que el valor de referencia proveniente de un proceso de integración de Runge-Kutta de 5to orden tiene sus propios errores, se arriba a que:

$$\eta = 0{,}84\ (\Delta t\ \varepsilon_{max} / \varepsilon)^{1/4} \qquad (45)$$

Finalmente, en el proceso de integración se varía el paso de cálculo con el siguiente criterio:

- Si $\varepsilon > \varepsilon_{max}$; se recalcula el valor de la función en el mismo punto "n" usando un paso de cálculo ajustado con el factor η.

- Si $\varepsilon < \varepsilon_{max}$; se corrige el paso de cálculo y se lo utiliza en la evaluación de la función en el próximo intervalo "n+1".

Este criterio básico de selección del paso de integración más conveniente puede ser ajustado según las experiencias que se obtengan para cada caso en particular. Obsérvese además que el esfuerzo extra de cálculo es el que corresponde a la evaluación de la Ecuación 43, ya que las seis evaluaciones de la función de las Ecuaciones 40 deben ser realizadas como parte del proceso normal de integración.

Las constantes que definen todas las expresiones anteriores quedan resumidos en la siguiente tabla :

	x = 1	**2**	**3**	**4**	**5**	**6**
$\mathbf{a_{2x}}$	1 / 4					
$\mathbf{a_{3x}}$	3 / 32	9/32				
$\mathbf{a_{4x}}$	1932 /2197	-7200/2197	7296 / 2197			
$\mathbf{a_{5x}}$	439 / 216	-8	3680 / 513	-845 / 4104		
$\mathbf{a_{6x}}$	-8 / 27	2	3544 / 2565	1859 / 4104	-11 / 40	
$\mathbf{b_x}$		1 / 4	3 / 8	12 / 13	1	1 / 2
$\mathbf{c_x}$	25 / 216	0	1408 / 2565	2197 / 4104	-1 / 5	
$\mathbf{d_x}$	16 / 135	0	6656 / 12825	28561/56430	-9 / 50	2 / 55
$\mathbf{e_x}$	1 / 360	0	128 / 4275	2197 / 75240	1 / 50	2 / 55

Tabla 1 : Constantes de expresiones de Runge-Kutta de 4º y 5º orden

Como contrapartida a las innegables ventajas que ofrecen los métodos de pasos variables para tratar sistemas rígidos, debe advertirse las dificultades que presentan su implementación cuando no se dispone de libertad para alterar el período final de integración, como es el caso de las aplicaciones de tiempo real.

2.3 Redes neuronales artificiales

El interés que despiertan aquí las redes neuronales artificiales tiene dos razones, que son enunciadas a continuación.

En primer lugar, se reitera la necesidad de representar el comportamiento del motor y freno dinamométrico, que responden a funciones altamente no lineales y de las cuales sólo se conocen sus evaluaciones experimentales. En ambos casos, se trata de modelos que deben predecir el valor de funciones a partir de dos parámetros de entrada y que deben cumplir las siguientes condiciones adicionales:

a. Capacidad para reproducir el comportamiento observado de motores y frenos reales a través del ajuste de sus constantes.
b. Posibilidad de extrapolar dentro de ciertos límites, en regiones donde se carece de lecturas.
c. Rápida respuesta para hacer posible su implementación en simulaciones en Tiempo Real.
d. Posibilidad de incorporar en el futuro nuevas entradas para contemplar otros parámetros (como es el caso de la temperatura del aire admisión).
e. Tolerancia a cierta dispersión y algunas inconsistencias en los datos, es decir, diferentes valores para las mismas condiciones de entrada.

Esta última condición es particularmente importante ya que los datos disponibles provendrán de mediciones efectuadas en distintos ensayos, y por lo tanto, es natural esperar que contengan ciertas discrepancias y componentes de error. Es necesario evitar que estos defectos en los datos produzcan oscilaciones o indeterminaciones en la respuesta de los modelos.

Para implementar estos modelos se dispone de varias alternativas, tales como la interpolación sobre superficies biliniales, las superficies bicúbicas de Bézier, las superficies B-Spline o los mínimos cuadrados (Rogers y Adams, 1990). Más recientemente se vienen utilizando redes neuronales multicapa de perceptrones, por sus interesantes propiedades como aproximadores universales de funciones. En efecto, estos modelos, que han demostrado sus ventajas en la representación de fenómenos no-lineales y donde los datos disponibles presentan un nivel de incerteza importante, son utilizados exitosamente en la Universidad de Salerno para representar el comportamiento de motores (Arsie, 1998 y 2001) y en otros muchos centros de investigación (Xiao, 1996 y Yin, 2001).

La segunda razón para justifica el interés por las redes neuronales excede los límites previstos para este trabajo. Se refiere al potencial que estas redes ofrecen para implementar sistemas de *control adaptativos*, que si bien no serán implementados aquí, ofrecen una promisoria alternativa que merece ser explorada en próximos trabajos.

El esquema clásico de control adaptativo, capaz de ajustar sus parámetros durante la operación del sistema (*en línea*) para brindar una mejor respuesta, se apoya en la hipótesis del comportamiento lineal del objeto controlado. Aquí las redes neuronales han demostrado ser una opción muy apropiada que permiten superar esta limitación y extender el alcance de estos sistemas a casos severamente no-lineales (Tan, 1997). Estas ventajas son aún más evidentes cuando se trata de

sistemas MIMO (*Multi Input-Multi-Output*), que como se observará es el caso del control de ensayos de motores (Pham, 1995).

Un controlador neuronal de este tipo invariablemente involucra una red que debe ser entrenada para reproducir la dinámica inversa de un proceso (Isasi, 2004). Retornando a la Ecuación 29, puede expresarse la salida de un proceso dinámico a partir de la señal de control z(t) y de la función de transferencia G(t):

$$y(t+1) = G(t).z(t) \qquad (46)$$

Luego, la relación inversa es una expresión de la forma:

$$z(t) = G^{-1}(t)\ y(t+1) \qquad (47)$$

Por lo tanto, la red neuronal se utiliza para implementar una estrategia de control inverso que al aproximar la relación G^{-1} puede conocerse la señal de control necesaria para alcanzar cierta salida en el sistema controlado.

2.3.1 Red multicapa de perceptrones

La ya mencionada propiedad de "aproximador universal de funciones" de estas redes las hacen capaces de encontrar cualquier tipo de relación no lineal múltiple entre las variables de un vector de entrada y los elementos de un vector de salida.

Esta propiedad que exhiben las redes de perceptrones, con estructuras de por lo menos tres capas, de ser capaces de reproducir cualquier función genérica continua fue formalmente demostrada por Kolmogorov y otros varios investigadores. Más aún, este autor estableció que una función entre dos vectores de dimensiones *n* (entrada) y *m* (salida) puede ser apropiadamente representada por una red con *n* unidades en la capa de entrada, un mínimo de 2*n*+1 unidades en la capa oculta y *m* unidades en la capa de salida. Cabe acotar que la denominación de "*oculta"* obedece a que en esta capa sus unidades no están en contacto directo con las entradas ni con las salidas. Una de estas redes es presentada esquemáticamente a continuación en la Figura 10, donde se representa una arquitectura con dos unidades en la capa de entrada, una unidad en la capa de salida y dos capas ocultas. En lo sucesivo se denominarán N y M a las cantidades de unidades en estas capas.

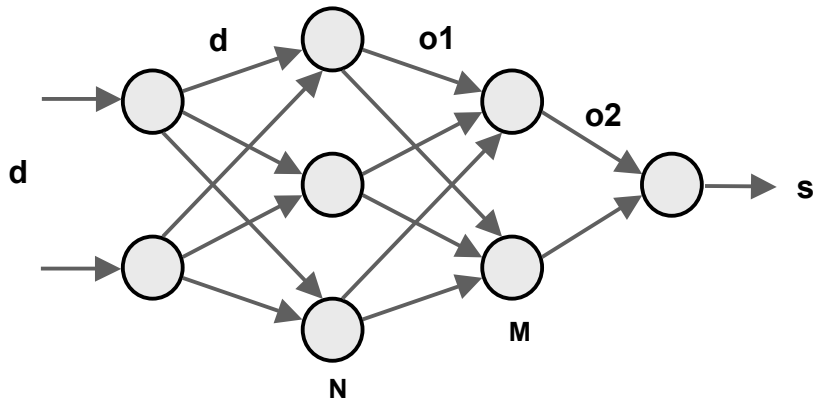


Figura 10 : Red de perceptrones con dos capas ocultas

Dentro del marco de las redes neuronales, el perceptron multicapa es en la actualidad una de las arquitecturas más utilizadas para la resolución de problemas, las que han sido adoptadas en este trabajo para representar el desempeño del motor y freno dinamométrico

La función desconocida *F* que se desea representar es expresada como:

$$\{s\} = F(\{d\}) \qquad (48)$$

y el proceso de aproximación de esta función por una red de perceptrones puede ser esquematizado según se indica:

$$\{d\} \longrightarrow \{d\} \xrightarrow{[W_{do}],\{\theta_d\}} \{o_1\} \xrightarrow{[W_{oo}],\{\theta_o\}} \{o_2\} \xrightarrow{[W_{os}],\{\theta_s\}} \{s\} \qquad (49)$$

donde las matrices [W] representan los pesos sinápticos o peso de la conexión y los vectores {θ} representan los umbrales de activación, según se describe en la simbología de este trabajo. Tantos los elementos de [W] como los de {θ} son números reales.

Resulta obvio que el resultado que se obtiene al evaluar la función depende de los pesos sinápticos y umbrales utilizados. El vector de entrada se proyecta hacia el de salida, avanzando de una capa hacia la siguiente, y para el caso de la red indicada en la Figura 10, este proceso puede expresarse como:

$$\begin{aligned} \{o_1'\} &= [W_{do}]\{d\} \quad \text{y} \quad \{o_1\} = \{\mathbf{f}(o_1')\} \\ \{o_2'\} &= [W_{oo}]\{o_1\} \quad \text{y} \quad \{o_2\} = \{\mathbf{f}(o_2')\} \\ \{s'\} &= [W_{os}]\{o_2\} \quad \text{y} \quad \{s\} = \{\mathbf{f}(s')\} \end{aligned} \qquad (50)$$

donde **f** representa una función bipolar denominada "*función de activación*", que debe ser derivable y puede implementarse de formas diversas. En estos casos, se adoptan funciones signoidales que responden a la expresión:

$$\mathbf{f}(a) = 1 / (1 - e^{-a}) \qquad (51)$$

o tangentes hiperbólicas cuya ecuación es:

$$\mathbf{f}(a) = (1 - e^{-a}) / (1 + e^{-a}) \qquad (52)$$

donde en ambos casos se verifica que $0 \leq \mathbf{f}(a) \leq 1$.

2.3.2 Arquitectura

La estructura mínima propuesta por Kolmogorov debe normalmente ser incrementada a efectos de lograr un proceso de convergencia más rápido para el entrenamiento de la red. No se presentan, sin embargo, para ello criterios formales ni prácticos de aplicación general y la estructura más conveniente para cada caso debe ser obtenida a partir de sucesivas pruebas.

Debe observarse que en las redes de perceptrones las conexiones siempre están dirigidas hacia adelante y todas las unidades de una capa están conectadas a todas las neuronas de la capa siguiente, prosiguiendo así hasta la capa de salida. No es posible demostrar formalmente que si se utilizan estructuras que escapen a esta regla se pueden obtener mejores resultados, aunque en algunos casos esto ha

podido ser comprobado. Estas variantes se obtienen eliminando conexiones o incorporando conexiones que vinculen capas no contiguas. Por el contrario, nunca pueden implementarse vínculos hacia atrás o de realimentación ya que éstos desvirtuarían la naturaleza de este tipo de redes.

2.3.3 Entrenamiento de la red

Para lograr que la red de perceptrones produzca los resultados correctos es necesario que los pesos sinápticos tengan los valores apropiados y éstos deben ser obtenidos a través de un proceso de ajustes sucesivos, denominado "*entrenamiento"*. Si bien desde los primeros años en que se presentaron estas redes se propusieron diversas variantes para ello, la verdadera revolución en el campo de las redes neuronales se produjo a mediados de los años '80 con la presentación que hizo Rumelhart del "*Método de propagación hacia atrás" o "Backpropagation" (*Rumelhart, .Hinton & Williams, 1986). Aquí es necesario señalar que, al margen de los méritos de Rumelhart, la aplicación de este tipo de algoritmos se hizo posible recién cuando las computadoras alcanzaron la capacidad y rapidez necesaria.

El método de propagación hacia atrás prevé un proceso iterativo donde la diferencia entre la salida esperada (dato experimental) y la obtenida de la red es reconocida como un error que pone de manifiesto que los pesos sinápticos no son los apropiados para representar la función y que estos pesos deben ser ajustados. Este ajuste progresivo es realizado calculando el error a la salida y proyectando ese error hacia las capas anteriores, pudiendo ser ejemplificado en el diagrama de la Figura 11 presentada a continuación:

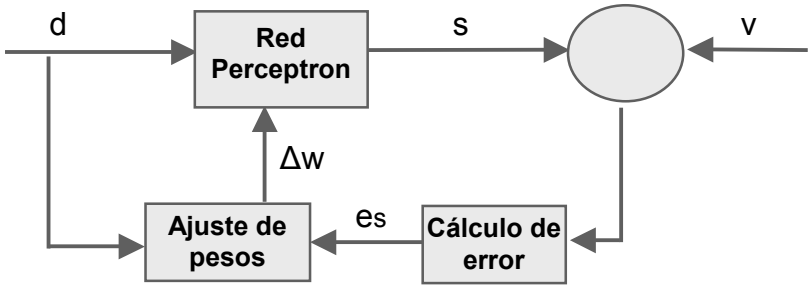


Figura 11 : Ajuste de pesos de la red

La técnica clásica de "backpropagation" no es más que una implementación del método del gradiente descendente, en el que se procura encontrar el mínimo de una función error en el hiperespacio de los pesos sinápticos de la red, como se representa en la Figura 12.

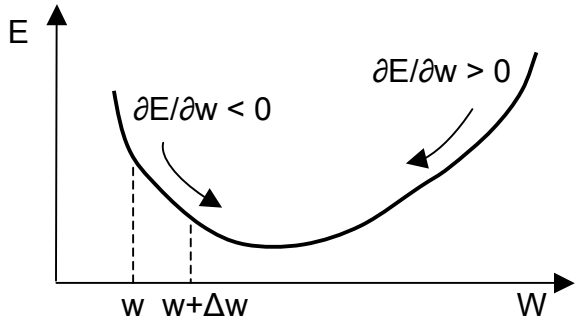


Figura 12 : Método del gradiente descendente

Para ello se determina en primer lugar el error en la salida de la red, que a partir de la correspondiente formulación diferencial (Nilsson, 2001) queda expresada por:

$$\{e_s\}=([v]-[s]).[s].(\{1\}-\{s\}) \quad (53)$$

y se determinan luego los errores de los vectores salida de las sucesivas capas ocultas, desde la última hasta la primera. Volviendo al ejemplo de la Figura 10:

$$\{u_2\}= [W_{os}]^T\{e_s\}$$

y

$$\{e_{o2}\}=[u_2].[o_2].(\{1\} - \{o_2\}) \quad (54)$$

$$\{u_1\}= [W_{oo}]^T\{e_{o2}\}$$

y

$$\{e_{o1}\}=[u_1].[o_1].(\{1\} - \{o_1\}) \quad (55)$$

Una vez determinados los errores de los vectores de salida, se ajustan las matrices de pesos sinápticos según las expresiones:

$$[\Delta W_{os}] = \eta\{e_s\}.\{o_2\}^T$$

y

$$[W_{os}] = [W_{os}] - [\Delta W_{os}] \quad (56)$$

$$[\Delta W_{oo}] = \eta\{e_{o2}\}.\{o1\}^T$$

y

$$[W_{oo}] = [W_{oo}] - [\Delta W_{oo}] \quad (57)$$

$$[\Delta W_{do}] = \eta\{e_{o1}\}.\{d\}^T$$

y

$$[W_{do}] = [W_{do}] - [\Delta W_{do}] \quad (58)$$

donde **η**, denominado "*factor de aprendizaje*", es un parámetro que determina la magnitud del desplazamiento sobre la superficie del error y define por lo tanto la velocidad de convergencia del algoritmo. Se presenta aquí uno de los puntos débiles más importantes de este método de aproximación de funciones. En efecto, para alcanzar la necesaria precisión durante el ajuste de los pesos sinápticos es necesario asignar valores bajos al factor de aprendizaje **η**. Esto permite evitar oscilaciones, pero conduce a procesos de convergencia muy lentos.

Procurando superar estas dificultades, se incorpora en cada ajuste de pesos un porcentaje del ajuste del ciclo anterior. Esta corrección incorpora cierta inercia en el proceso de ajuste y a este factor de incidencia se lo denomina "*momentum"* **β**. Sin embargo, aún así y dependiendo de las características de la superficie que representa la función error, el proceso de aprendizaje puede quedar atrapado en mínimo locales o planicies.

Además, la necesidad de asignar valores apropiados a los factores **η** y **β** incorpora nuevos parámetros que deben ser definidos mediante un proceso exploratorio, tal como ocurre con la selección de la arquitectura más conveniente. Esto dificulta el proceso de entrenamiento y hace imposible asegurar que se ha encontrado la solución óptima.

2.3.4 Evaluación del error total

Para un conjunto de *n* pares de vectores de entrenamiento ({e},{v}) el error total *E* de la red es obtenido a partir de la contribución de los errores que se observan en cada uno de estos pares, donde:

$$E^2 = (1/n) \sum_{1}^{n} (\{v\} - \{s\})^2 \qquad (59)$$

2.3.5 Otros métodos de entrenamiento

En los últimos años se vienen aplicado diversas técnicas del análisis numérico (Valishevsky,1998) con el fin de superar las dificultades señaladas y acelerar los procesos de convergencia. Algunas de ellas, como el método "*QuickProp"*, recurren a la curvatura de la función error determinando su segunda derivada. En este caso, las ventajas son discutibles, ya que no es posible asegurar que el esfuerzo de cálculo extra que implica la determinación del ajuste de pesos será compensado con una reducción en los ciclos de cálculo que son necesarios para entrenar la red.

Otro método destacable es el algoritmo "*Rprop"* cuyo nombre es un acrónimo de "**R**esilient back**prop**agation" (Riedmiller, 1994). Este método difiere de la técnica de propagación hacia atrás clásica en que las derivadas parciales de la función error sólo son usadas para determinar el sentido en que deben ser corregidos los pesos de la red pero no las magnitudes de los ajustes. A estos ajustes se les asigna un valor inicial que es luego corregido en cada ciclo de cálculo con el siguiente criterio:

el ajuste de pesos en cada paso de cálculo "p" es: $\Delta w^{p} = \mu \, . \, \Delta w^{p-1}$ (60)

$$\text{donde si } \frac{\partial E^{p-1}}{\partial w} \frac{\partial E^{p}}{\partial w} < 0 \Rightarrow \mu = 0{,}5 \qquad \text{y si } \frac{\partial E^{p-1}}{\partial w} \frac{\partial E^{p}}{\partial w} > 0 \Rightarrow \mu = 1{,}2 \qquad (61)$$

lo que significa que se reduce el tamaño del ajuste en caso de encontrarse un mínimo de la función error y aumentado en caso contrario (Figura 12).

Se presentan algunos métodos más recientes que proponen estrategias de ajuste no monótonas que permiten que la función error se incremente en ciertos puntos, conduciendo a procesos que, aunque moderadamente oscilantes, siempre son estables, confiables y normalmente más rápidos (Plagianacos, 2002). En estas técnicas, se determina en forma dinámica el valor del factor de aprendizaje **η** más apropiado en cada caso, presentándose propuestas tales como el "*BPVS"* (**B**ack **P**rop **V**ariable **S**tepsize):

$$\boldsymbol{\eta} = ½ . \, | w^{p} - w^{p-1} | \; / \; | \Delta E(w^{p}) - \Delta E(w^{p-1}) | \qquad (62)$$

o el "*ABP"* (**A**daptive **B**ack **P**ropagation) donde :

$$\boldsymbol{\eta}^{p} = \begin{cases} \boldsymbol{\mu}.\boldsymbol{\eta}^{p-1} & \text{si } E(w^{p}) < E(w^{p-1}) \;\; ; \;\; \boldsymbol{\mu} > 1 \\ \boldsymbol{\delta}.\boldsymbol{\eta}^{p-1} & \text{si } E(w^{p}) > E(w^{p-1}) \;\; ; \;\; 0 < \boldsymbol{\delta} < 1 \\ \boldsymbol{\eta}^{p-1} & \text{si } E(w^{p}) = E(w^{p-1}) \end{cases} \qquad (63)$$

Por último, deben mencionarse las técnicas que trabajan sobre la definición de los pesos iniciales. Para la mayoría de los casos anteriores se deben adoptar valores aleatorios pequeños, mientras que una alternativa es encontrar la mejor distribución de pesos iniciales, según la naturaleza de la función reproducida y la arquitectura de la red. Estos pesos iniciales procuran que cada unidad tenga el mejor desempeño en el proceso de entrenamiento, acelerando la convergencia y evitando fenómenos de saturación prematura (Wlodzislaw, 1997). En algunos casos, se propone un incremento progresivo en las unidades de la red con determinación de los pesos iniciales de las nuevas unidades a partir de los pesos en la red anterior (Delashmit, 2002 y 2003).

2.3.6 Conducción del proceso de entrenamiento

Al evaluar el comportamiento de una red de perceptrones no sólo es necesario saber si la red fue exitosamente entrenada, también es indispensable comprobar el comportamiento de la red ante patrones no utilizados en el proceso de aprendizaje. Esto significa que, durante el entrenamiento, la red debe extraer de las muestras las características de la función a representar y no memorizar los vectores utilizados en este proceso, como se ilustra en la Figura 13.

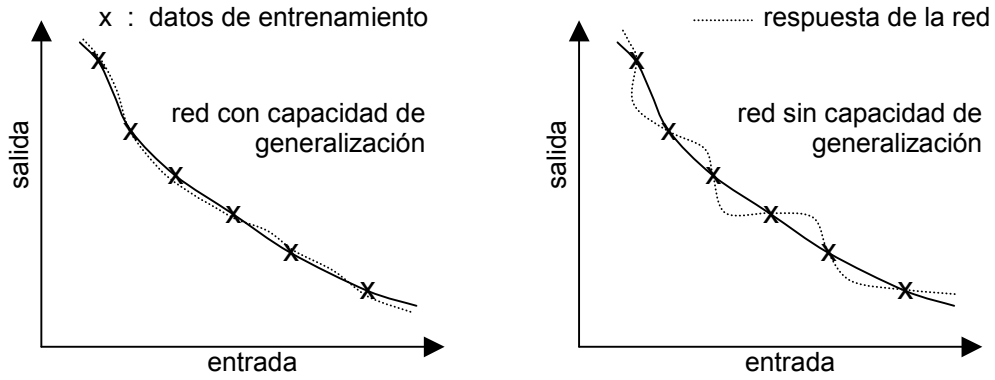


Figura 13: Capacidad de generalización de las redes neuronales

Esta propiedad de la red es denominada "*capacidad de generalización*" y debe ser comprobada en todos los casos para asegurar que la red brindará la respuesta esperada.

Por lo tanto, cuando se realiza el proceso de aprendizaje, es muy importante comprobar paralelamente su capacidad de generalización. Para ello es necesario dividir al conjunto de pares de entrenamiento en dos grupos, uno de ellos para ajustar los pesos de la red y el otro de validación para comprobar su generalización. Más aún, es recomendable repetir el proceso de entrenamiento intercambiando el rol de ambos grupos de datos o inclusive formar nuevos grupos fraccionando los anteriores. En la Figura 14 se representan las dos posibilidades que presenta la evolución de los errores en el proceso de aprendizaje.

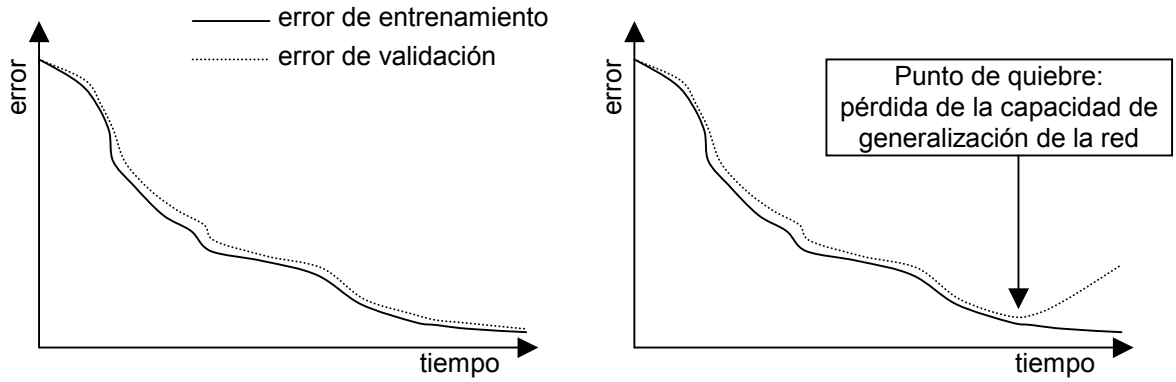


Figura 14: Evolución del error en el proceso de entrenamiento

Una grave equivocación, frecuentemente cometida, es la de evaluar solamente el error de entrenamiento, con lo cual puede ocurrir que no se advierta que la red ha perdido generalidad y se continúe el proceso de ajuste de pesos.

2.4 Aproximación de funciones

Tal como fue reseñado en el punto 2.1.4, el proceso y la unidad de control pertenecen a diferentes dominios. Por este motivo, los valores del vector de estado **y** del proceso sólo pueden ser conocidos desde el sistema de control a través de interfases, que se materializan mediante sensores y conversores A/D. Si bien la conversión de señales será desarrollada más adelante en el punto 4.5, es fácil advertir que difícilmente habrá simultaneidad en las mediciones de las diferentes variables, ya que sus frecuencias de muestreo serán distintas, o como mínimo, estarán desfasadas en el tiempo.

Sin embargo, en la unidad de control se deben conocer los valores de todos los componentes del vector de estado para un mismo instante de tiempo; a su vez esta unidad operará con su propia frecuencia, por lo que es necesario prever un procedimiento de extrapolación de estas funciones. Es necesario conocerlas en cierto instante y además, para algunos de los criterios incorporados en la unidad de control, es necesario conocer también las derivadas de estas variables.

Todo esto justifica la necesidad de disponer de un método eficiente para extrapolar y derivar funciones a partir de un muestreo discreto, tal como se presenta en la Figura 15; y para ello se analizaron diferentes posibilidades.

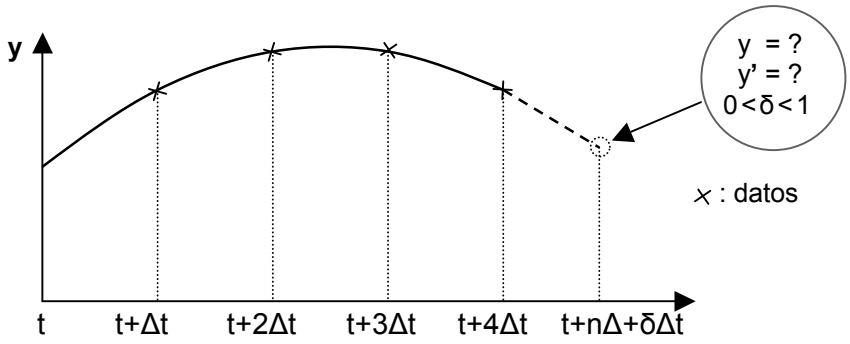


Figura 15 : Aproximación de funciones

Tras evaluar las opciones disponibles, se seleccionó una aproximación por mínimos cuadrados con un polinomio de tercer grado como el método más conveniente. Aquí se propone un polinomio de grado bajo por ser compatible con la evolución que se espera de la función, y para el caso de la derivada, también se espera acotar el riesgo de error como consecuencia de alguna oscilación en el extremo extrapolado.

En realidad, la regresión por mínimos cuadrados es utilizada cuando los datos incluyen errores importantes e inclusive se dispone de numerosas lecturas para un cierto instante de tiempo. Aquí se presenta la primera de estas condiciones cuando el sistema de control trabaja en una instalación real, pero no es el caso del simulador, en que los elementos del vector de estado son generados por otro proceso. Sin embargo, aún así ésta es una buena elección para extrapolar y obtener la derivada de las funciones.

Asumiendo entonces que la función será aproximada por una ecuación de tercer grado:

$$y = a_o + a_1.t + a_2.t^2 + a_3.t^3 \qquad (64)$$

puede determinarse el error que se comete cuando se la utiliza para representar un conjunto de *n* pares de datos (y,t):

$$E = \sum_{1}^{n} (y_k - a_o - a_1.t_k - a_2.t_k^2 - a_3.t_k^3)^2 \qquad (65)$$

Los coeficientes pueden ser evaluados haciendo mínima la función que representa el cuadrado del error (McCraquen,1964 y Chapra,2004), con lo que se obtienen las ecuaciones:

$$\partial E/\partial a_i = 0 \;\; ; \;\; i = 0,1,2,3 \qquad (66)$$

que conducen a la expresión matricial:

$$\mathbf{B}.\mathbf{a} = \mathbf{c} \qquad (67)$$

donde los elementos de la matriz **B** y del vector **c** quedan definidas como:

$$B_{ij} = \sum_{1}^{n} t_k^{\,i+j-2} \qquad (68)$$

$$c_i = \sum_{1}^{n} y_k.t_k^{\,i-1} \qquad (69)$$

y en forma extendida el sistema queda expresado como:

$$\begin{pmatrix} n & \sum t_k & \sum t_k^2 & \sum t_k^3 \\ \sum t_k & \sum t_k^2 & \sum t_k^3 & \sum t_k^4 \\ \sum t_k^2 & \sum t_k^3 & \sum t_k^4 & \sum t_k^5 \\ \sum t_k^3 & \sum t_k^4 & \sum t_k^5 & \sum t_k^6 \end{pmatrix} \begin{Bmatrix} a_o \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} \sum y_k \\ \sum y_k.t_k \\ \sum y_k.t_k^2 \\ \sum y_k.t_k^3 \end{Bmatrix} \qquad (70)$$

de donde se obtienen los coeficientes de la parábola cúbica resolviendo el sistema de ecuaciones. Es decir:

$$\mathbf{a} = \mathbf{B}^{-1}.\mathbf{c} \qquad (71)$$

Una vez definidos estos coeficientes, la ecuación cúbica (64) permite extrapolar la función y obtener también la expresión de su derivada:

$$y' = a_1 + 2.a_2.t + 3.a_3.t^2 \qquad (72)$$

Al trabajarse con cuatro pares de puntos, como es éste el caso, pudo haberse determinado en forma directa una parábola cúbica que pase por todos ellos. Sin embargo, la función extrapolada no es parabólica y al forzarla con esta aproximación pueden generarse oscilaciones que ocasionarían un gran error al extrapolar y un error aún mayor en la derivada.

La regresión por mínimos cuadrados es apropiada para evitar este tipo de fenómenos y además ofrece mayor libertad, ya que permite que en cada caso se utilicen de las cantidades de pares de puntos que parezcan más convenientes.

3 Marco metodológico

Se retorna, en este apartado, al objeto del presente trabajo, que es la construcción de un modelo computacional destinado a simular la interacción de una unidad de control y un conjunto mecánico, y para ello se seguirán las pautas ya establecidas para el desarrollo de todo sistema de computación. Para comenzar, es conveniente presentar el contexto del problema y para ello se hace referencia al modelo en cascada del ciclo de vida del desarrollo de software, que es representado en la Figura 16. Este modelo prevé una fase inicial de ingeniería de requerimientos, que es seguida por las de diseño, programación y testeo. A esta fase inicial, que está destinada a la identificación, análisis y documentación de los requerimientos, se le asigna en la actualidad particular importancia, por ser la encargada de definir el primer modelo conceptual del sistema a ser desarrollado.

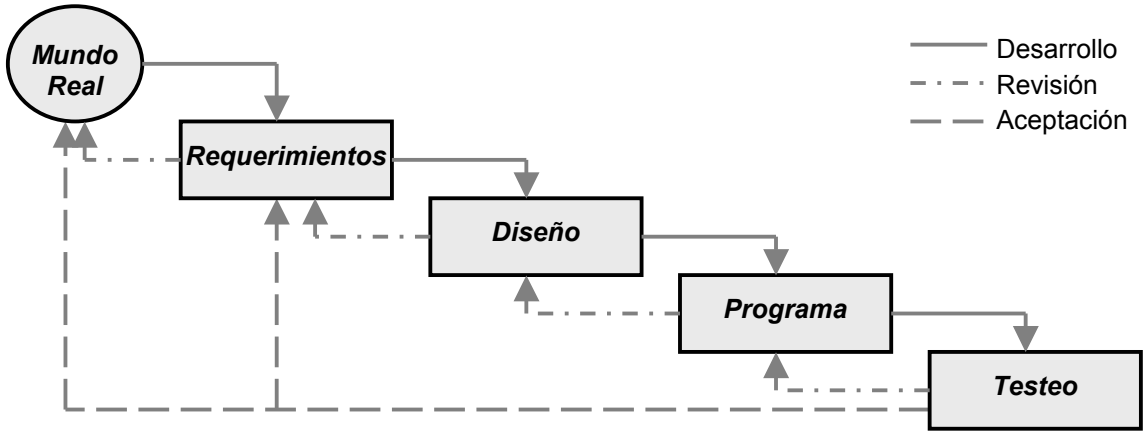


Figura 16 : Ciclo de vida del desarrollo de software

Si bien el *paradigma de objetos* ha cambiado la forma en que actualmente se desarrolla software, no son pocos los autores (Bustos Reinoso,1999) que consideran que es cuestionable su utilización en la fase inicial del modelado conceptual de un sistema; y para tratar este problema se han presentado numerosas y diversas propuestas.

Hacia fines de los años ’80, era común encontrar metodologías de análisis y diseño orientado a objetos (Constantine, 1989) que utilizaban técnicas ajenas a los objetos en el modelado conceptual. En estas propuestas, y con algunas dificultades, se identificaban a partir de los DFD y DER los objetos que formarían parte del modelo que luego sería utilizado en la posterior fase de diseño.

Poco tiempo después, al comenzar los ’90, varias de las nuevas metodologías sugerían que había que desechar todo modelado que no estuviese orientado a objetos y recomendaban completar íntegramente el ciclo de desarrollo bajo este paradigma. Como ejemplos de esta tendencia, pueden citarse los trabajos de Rebecca Wirfs-Brock (Wirfs-Brock,1990) y de Kenneth Rubin (Rubin,1992).

Sin embargo, estas metodologías entraron en conflicto con el principio de la ingeniería de software que establece que el análisis debe ser declarativo al definir “*qué”* debe hacerse, en contraposición con el diseño que, por ser una actividad creativa, debe definir “*cómo”* ha de hacerse. En la práctica, los modelos orientados a objetos surgidos de la fase de análisis sólo son menos detallados que los del diseño, con lo que la distinción conceptual de ambas fases aquí se desvanece. Más aún,

hasta podría decirse que los conceptos de *análisis* (descomposición y examen crítico) y de *orientación a objetos* (encapsulamiento de atributos y capacidades) en realidad se contraponen (Bustos Reinoso,1999).

Una forma de resolver este conflicto es postergar el encapsulamiento tanto como sea posible, lo que implica sacar el modelado conceptual fuera del paradigma de objetos, adoptando para ello métodos que dispongan de la necesaria flexibilidad y poder expresivo.

Bajo este razonamiento, se retornó a la utilización de modelos no orientados a objetos en la especificación de requerimientos, para luego derivar de éstos los modelos de objetos. Resulta indudable que para ello contribuyó enormemente la amplia aceptación que experimentó el modelo de *Casos de Uso* propuesto originalmente por Ivar Jacobson (Jacobson, 1992) y su posterior difusión acompañando a UML (Jacobson, Booch y Rumbaugh,1999).

No obstante, es necesario recordar que los *Casos de Uso* recibieron en sus comienzos algunos comentarios bastante desfavorables, que con el tiempo fueron quedando en el olvido. Si bien muchas de estas críticas fueron en su momento relativizadas por provenir de defensores del *Análisis Estructurado* o del modelado de "*puro objetos*", al releerlas se encuentran en ellas mensajes muy interesantes y algunas enseñanzas muy oportunas que mantienen hoy plena vigencia.

Una de estas opiniones desfavorables, que era compartida por numerosos autores, destacaba que los *Casos de Uso* inducían a perpetuar viejas soluciones al reproducir en detalle sistemas existentes, ya sean computarizados o no, en lugar de contribuir a encontrar la mejor propuesta para resolver un problema.

Recurriendo a críticas más específicas, puede citarse la opinión de Bertrand Meyer (Meyer, 1997:739), quién puntualizó que los *Casos de Uso* podían conducir con facilidad a una mala definición de las clases, por lo que solamente los recomendaría para el análisis orientado a objetos en caso de disponerse de un equipo de desarrollo muy experimentado.

Otra crítica se centraba en que los *Casos de Uso* quitaban flexibilidad a los modelos al enfatizar en los flujos de control una prematura secuencialidad, cuando hay todavía insuficiente conocimiento de un problema. Al respecto, Don Mills (Mills, 2002) destacó que las secuencias de operaciones que se obtenían normalmente representaban restricciones del modelo y no verdaderos requerimientos y que, por el contrario, las verdaderas secuencias deberían ser definidas con mayor nivel de abstracción a partir de precondiciones.

Por su parte, Larry Constantine enfatizó (Constantine, 1995) la necesidad de conocer con profundidad los problemas, dejando al margen elaboraciones innecesarias o restricciones artificiales. Para ello propuso sus "*Casos de Uso Esenciales*", incorporando a los *Casos de Uso* de Jacobson los conceptos del *Análisis Esencial* desarrollado por Steve McMenamin y John Palmer (McMenamin, 1984).

A la luz de todas estas críticas, parece oportuno destacar un enfoque que representó una muy interesante alternativa al UML. Éste fue presentado por Leandro Antonelli (Antonelli, 1999) y condujo a definir las mismas *Fichas CRC* (*Clases, Responsabilidades y Colaboraciones*) propuestas por Rebecca Wirfs-Brock (Wirfs-Brock,1990), solo que formando parte de una secuencia en la que se estableció

previamente el *Léxico Extendido del Lenguaje* (*LEL*) y la descripción del problema a través de *escenarios*. Para lo primero se utilizó una metodología propuesta por J.C. Leite (Leite,1995) a la que se le incorporaron los escenarios y las heurísticas necesarias para derivar estas *Fichas CRC*.

Todo lo expuesto permite comprobar que en las últimas dos décadas se ha venido desarrollando un importante cuerpo de teorías, métodos y técnicas que, con mayor o menor éxito, han procurado demostrar que son apropiadas para el análisis, diseño e implementación de sistemas de computación de variada naturaleza.

Sin embargo, y tal como ya fue anticipado en el prólogo de este trabajo, esta afirmación no es completamente válida para los sistemas de computación en general y lo es aún menos al estar referida a *sistemas de tiempo real*, en cuyo tratamiento todavía se aprecia un importante nivel de desconcierto y es frecuente el uso de metodologías ad-hoc carentes de toda generalidad (Huang, 2004).

Al respecto, ya fue señalado que los sistemas de tiempo real poseen requerimientos críticos que están referidos a su comportamiento temporal, precisión y seguridad, algunos de los cuales pueden ser identificados como requerimientos *no funcionales*. A esto es necesario agregar la también comentada creciente difusión que estos sistemas vienen teniendo, con su presencia en los más diversos e inclusive insospechados ámbitos del quehacer humano.

Una respuesta a este problema ha arribado por el lado de los métodos formales, que están siendo objeto de un renovado interés que se manifiesta con la presentación de una amplia variedad de propuestas. En efecto, su natural rigurosidad los convierte en una opción muy atractiva para estos casos en que la ambigüedad e inconsistencia son inaceptables.

Sin embargo, a pesar de que muchos de estos métodos demostraron ser apropiados para soportar complejos desarrollos de software, debe reconocerse que su difusión es normalmente muy limitada y aún más escasa su utilización en la industria del software. Es decir, estas propuestas difícilmente trascienden los ámbitos universitarios y los centros de investigación. Uno de los problemas es la disponibilidad de desarrolladores capacitados en estos métodos, que no crece con la rapidez con que lo hace la demanda de este tipo de software. Además, las necesidades de una formación más sólida y mayor tiempo de entrenamiento los hace demasiado costosos para el desarrollo “ordinario” de sistemas de tiempo real.

Por el contrario, el ya mencionado UML continúa consolidando su gran difusión en todo tipo de desarrollos e inclusive viene ganando popularidad en el campo de los sistemas de tiempo real. En efecto, UML para Tiempo Real (alias UML-RT) ya despierta mucho interés y sus especificaciones están incluidas en la definición de UML 2.0. A pesar de ello, la aplicación de UML-RT en el dominio de los sistemas de tiempo real manifiesta todavía algunos problemas importantes, tales como:

a) La carencia de una adecuada definición formal dificulta los procesos de verificación o simulación.

b) A pesar de tratarse de una notación efectiva para el diseño e implementación de sistemas, muestra su mayor debilidad en la definición y representación de sus especificaciones. En particular, no son soportadas algunas conductas tales como la no determinista y las que son consecuencia de la simultaneidad de eventos.

c) La representación del tiempo y de sus restricciones no son facultades nativas y las mejoras que son introducidas presentan todavía algunas restricciones.

Aquí cabe recordar que UML es un lenguaje de modelado cuya notación permite capturar y comunicar tanto la estructura de objetos como su comportamiento, pero que no fue originalmente concebido para modelar sistemas de tiempo real. Por ello, las limitaciones señaladas vienen motivado diversos líneas de investigación con la finalidad de superarlas. Entre otras muchas, pueden mencionarse las metodologías y herramientas para el desarrollo de sistemas de tiempo real orientados a objeto basadas en UML propuestas por Becker (Becker, 1999) y Drake (Drake, 2001).

Todo lo expuesto permitió comprobar que no existe un enfoque consolidado para el tratamiento de los sistemas de tiempo real, de donde surgió la idea de explorar un enfoque alternativo que combine diferentes metodologías y que es presentado a continuación

3.1 Ingeniería de requerimientos

La primera fase del ciclo de vida del desarrollo de software debe conducir a la elaboración de las *especificaciones de sus requerimientos* (*SRS*), como producto final de un proceso iterativo que incluye actividades de elicitación, análisis, documentación, verificación y validación. Las dos últimas actividades están destinadas a la revisión de los requerimientos y cabe aquí citar el difundido ejemplo de B. Boehm (Boehm, 1984) que contribuyó a establecer claramente la diferencia entre ambos conceptos, formulando las siguientes dos preguntas: a) *validación*: ¿estoy construyendo el producto correcto? y b) *verificación*: ¿estoy construyendo correctamente el producto?

Estas especificaciones (*SRS*) estarán destinadas a establecer los requerimientos del sistema de manera completa, correcta, precisa, no ambigua y consistente. Pare ello, la consideración de estos atributos es esencial realizarla en el marco de las cualidades exigidas para la solución al problema planteado, que por tratarse de un sistema de tiempo real incluye de manera destacada la confiabilidad y la eficiencia en su desempeño. Mas aún, es sabido que las especificaciones de los requerimientos de estos sistemas de tiempo real deben contemplar tanto un enfoque estructural como de comportamiento de sus componentes, y para esto último es esencial incluir un detallado estudio de sus interacciones con un riguroso tratamiento temporal.

Se comienza aquí por aceptar que para tratar correctamente un problema es fundamental un preciso conocimiento de la semántica de su vocabulario. Su correcta definición y la de su impacto en el problema contribuirán indudablemente a la obtención de especificaciones de requerimientos consistentes y carentes de ambigüedades. La veracidad de esta afirmación es aún más evidente en casos como el tratado, en el que se involucran áreas tan diversas como Mecánica, Electrónica, Teoría de Control, Metrología y Ciencia de la Computación.

3.1.1 Léxico extendido del Lenguaje

Por ello parecieron apropiadas la recomendaciones de JC Leite (Leite,1995) y Leonardi (Leonardi,1998) de construir un "*Léxico Extendido del Lenguaje*" (*LEL*)

como primer paso en la fase de elicitación de requerimientos, para lo cual se deja de lado, por el momento, el entendimiento del propio problema.

El *LEL* es un meta-modelo diseñado para ayudar a capturar el vocabulario de la aplicación y su entorno, utilizando el lenguaje natural para la representación de sus símbolos. Para ello se describen las nociones e impactos de cada palabra o frase, donde la noción describe su significado y el impacto determina los efectos del uso u ocurrencia del elemento en la aplicación. Es así que, según cada caso, la semántica de cada símbolo se representa con una o más nociones y uno o más impactos. Este conjunto de símbolos forman una red que permite representar al *LEL* en un hipertexto y navegar en él para conocer todo el vocabulario del dominio.

En la descripción de las nociones e impactos existen dos reglas básicas que se deben cumplir simultáneamente y son las siguientes:

a) *Principio de circularidad*: maximizar el uso de símbolos al definir el significado de otros símbolos.
b) *Principio del vocabulario mínimo*: minimizar el uso de símbolos externos al lenguaje de la aplicación.

Además, para poder lograr una descripción apropiada de los términos del *LEL* se debe establecer si éstos son utilizados como sujetos, verbos u objetos, o si describen situaciones en las frases que los actores utilizan para enunciarlos.

En el presente trabajo, se ha previsto la construcción de un *LEL* en forma manual a través de un editor de textos. Sin embargo, se advierte la conveniencia de disponer de una herramienta CASE que mediante la tecnología de hipertexto pueda facilitar la administración de la información y su validación. En las Figuras 18 y 19 se representan esquemas globales de la metodología propuesta y el lugar que allí ocupa el *LEL*.

3.1.2 Análisis esencial

Para continuar con la definición de los requerimientos, se descartaron los *Casos de Uso* propuestos originalmente por Ivar Jacobsen (Jacobsen,1992) y que se han convertido en una de las principales herramientas de prácticamente todas las metodologías *orientadas a objetos* de la actualidad. Estos *Casos de Uso* están destinados a describir cursos de acción entre un actor y el sistema, donde un curso de acción es llamado escenario y el actor cumple un rol acorde a la función seleccionada. Si bien el actor es un concepto abstracto que puede representar a un ser humano, una máquina o a un sistema, los *Casos de Uso* no parecen representar la mejor opción para especificar las interacciones entre los componentes de los sistemas de tiempo real. Su incapacidad para brindar información sobre las exigencias temporales del escenario representan su mayor limitación.

En su reemplazo, se decidió recurrir a la idea original de especificar un sistema a partir de su interacción con el entorno, bastante anterior al trabajo de Jacobsen, y como él mismo reconoció, fuente de inspiración de sus *Casos de Uso*. Se hace referencia aquí a la propuesta presentada por McMenamin y Palmer en su libro (McMenamin,1984) en el que los autores introducen el concepto de “evento” como delator de una circunstancia externa que afecta al sistema. Estos eventos

provocan los estímulos a los que el sistema debe responder y son los puntos de partida en el reconocimiento de sus características y su posterior modelado.

Esta metodología fue denominada "*Análisis Esencial de Sistemas*" y se centró en el desarrollo de un único modelo denominado "*Modelo Esencial*". Con este enfoque, el esfuerzo se concentra en identificar los requerimientos lógicos del nuevo sistema que es descompuesto a partir del reconocimiento de la ocurrencia de *"eventos*" en su entorno exterior. Así, los objetivos del sistema son presentados desde un punto de vista externo en lugar de interno, ya que es visualizado como una entidad dinámica que interactúa con su medio ambiente. Como consecuencia y confirmando las ventajas del procedimiento presentado, se obtienen modelos más rigurosos y en gran medida independientes de quienes lo construyen (Prieto, 2002).

De esta manera, un sistema de información es visto como un *Sistema de Respuesta Planificada*, en el que sus actividades son definidas considerando los eventos que deben ser atendidos, sin perjuicio de la forma en que éstas sean implementadas ni de la tecnología que pueda ser empleada. Aquí cabe aclarar que es necesario distinguir dos tipos de eventos, los *externos* y los *temporales*. Ya fue anticipado que todos los eventos se originan en el entorno externo al sistema y están fuera de su propio control, pero mientras los eventos *externos* se originan en cambios en las condiciones de su contexto, los eventos *temporales* tienen su origen en horarios, fechas (calendarización absoluta) o en el cumplimiento de intervalos de tiempo (calendarización relativa).

La ocurrencia de un evento hace normalmente reaccionar al sistema en forma predeterminada, cumplir cierta actividad y eventualmente entregar una respuesta al medio. Cabe aquí contemplar la posibilidad de que el sistema deba responder espontáneamente a un evento que no fue anticipado, denominado evento "ad hoc". En cualquier caso, el sistema cumple una "*actividad esencial*" que lo hará transitar desde su estado inicial a un cierto estado final en el que quedará nuevamente en reposo a la espera del próximo evento.

A partir de todo, esto se reconocen dos importantes características de las "*Actividades Esenciales*" (Satzinger, 1992):

a) Incluyen todas las acciones que se esperan de un sistema en respuesta a un único evento.

b) Son completas.

Además, las ventajas que este enfoque ofrece para facilitar la identificación de las exigencias temporales asociadas al comportamiento del sistema son evidentes.

Aquí corresponde reconocer que no es nueva la idea de recurrir al análisis esencial para el modelado conceptual y luego derivar desde allí el modelo de objetos, ya que ésta ha tenido numerosos antecedentes.

El primero de ellos se encuentra en el citado libro de McMenamin y Palmer (McMenamin, 1984, capítulo 7) que recomienda identificar los objetos a partir de un particionamiento de la memoria esencial, planteando claramente el concepto de clases sólo que sin asignarles esta denominación.

Otra propuesta en esta dirección fue la de Meilir Page-Jones (Page-Jones, 1992), que utilizaba los eventos de McMenamin y Palmer para definir sub modelos que conducían a los objetos, en un método que denominó "*Synthesis*".

En este contexto también Larry Constantine hizo su aporte con los ya mencionados *Casos de Uso Esenciales* (Constantine, 1995). Resulta quizás conveniente puntualizar que el *Análisis Esencial* y el de *Casos de Uso* pertenecen a diferentes niveles de abstracción. En efecto, mientras que el *Análisis Esencial* se apoya en las acciones y reacciones entre medio ambiente y sistema, el modelo de *Casos de Uso* lo hace con sus protagonistas (a*ctores*). Si en el *Análisis Esencial* se reemplazan las acciones por sus protagonistas ambos métodos quedan encolumnados con la sola diferencia de sus niveles de abstracción. Se convierte así el *Análisis Esencia*l en lo que Larry Constantine denomina "*Casos de Uso Esenciales*°. Según este enfoque, en el modelado de sistemas quedan establecidos tres niveles de abstracción: "*Casos de Uso Esenciales*", "*Casos de Uso*" y "*Escenarios*".

Asimismo, cabe reconocer que las ideas subyacentes en la propuesta de McMenamin y Palmer se mantienen presentes en la mayoría de los trabajos actuales referidos a sistemas de tiempo real, sólo que fragmentadas y por lo tanto carentes de identidad como metodología. Resulta curioso comprobar los esfuerzos de algunos autores actuales para adaptar los *Casos de Uso* al tratamiento de estos sistemas, cuando la respuesta está disponible desde hace tanto tiempo en el propio *Análisis Esencial*. Por ello, pareció conveniente recurrir a la propuesta original y a la excelente interpretación de John Satzinger (Satzinger,1992).

En resumen, y a partir de los antecedentes expuestos, se concluye que el *Análisis Esencial* conforma una perspectiva rigurosa que resulta muy apropiada para la identificación de los requerimientos de sistemas de Tiempo Real y, tal como se muestra en los esquemas de las Figuras 18 y 19, es adoptada en este trabajo.

3.1.3 Actividades esenciales, responsabilidades y fichas CRC

Una vez establecidos los términos del *LEL* y completado el *Análisis Esencial*, las actividades esenciales allí identificadas quedan directamente asociadas a las responsabilidades del sistema. Por su parte, las responsabilidades son sentencias que describen los servicios que deben brindar las clases del sistema, expresadas como acciones que se realizan o como conocimientos que se mantienen y proveen. Esto significa que el conjunto de las responsabilidades de todas las clases representa lo que un sistema global es capaz de realizar y la forma en que esta capacidad está distribuida.

Por lo tanto, la metodología aquí propuesta se apoya en las responsabilidades para vincular las actividades esenciales y los objetos, conformando un camino bidireccional entre ambos. Obsérvese que queda por lo tanto asegurada la trazabilidad entre las actividades esenciales y la estructura de clases.

Para ello, se contempla el ordenamiento de la información en un formato de ficha denominada *CRC* (*Clases, responsabilidades y colaboraciones*) que facilita un proceso de ajustes sucesivos que implica la asignación y delegación de responsabilidades, que finaliza con la obtención de una estructura que resulte satisfactoria.

Estas fichas, o más bien su formato, definen en realidad una herramienta informal que contribuye a la identificación de las clases y las relaciones entre ellas. Para ello, se debe establecer en cada caso una heurística apropiada, con la que se debe asociar a cada tarjeta una de las clases que van siendo identificadas, y en ellas se especifican sus responsabilidades y sus colaboraciones con otras clases.

Aquí, debe advertirse que un objeto puede cumplir con sus responsabilidades por sí mismo o solicitando a su vez la asistencia a otro objeto, es decir, solicitando la necesaria colaboración. Una colaboración es, por lo tanto, una solicitud realizada por un objeto a otro objeto, que asume la forma de un requerimiento de un cliente hacia un servidor. Para identificar las colaboraciones de cada clase es conveniente examinar sus responsabilidades, con el fin de determinar si necesitará de la interacción con otras clases para su cumplimiento. En caso afirmativo, estas últimas deben ser identificadas.

En resumen, las responsabilidades son asociadas con los objetos e identifican actividades que deben cumplirse. Estos objetos pueden, a su vez, atender sus responsabilidades acudiendo a otros objetos, designados como colaboradores. En este contexto, las responsabilidades guían la articulación del sistema, particionando clases y redistribuyendo, a su vez, estas responsabilidades.

Las tarjetas CRC fueron originalmente propuestas como una herramienta para la enseñanza del paradigma de objetos por Beck & Cunningham (Beck, 1989) y alcanzaron amplia difusión a partir del libro de Rebecca Wirfs-Brock (Wirfs-Brock, 1990). El formato que se adopta, en el marco del presente trabajo para implementar las fichas CRC es el habitual, conteniendo el nombre de la clase representada, el detalles de sus responsabilidades y el detalle de las clases que colaboran. Una vez que han sido identificadas, las clases son agrupadas en primarias y secundarias.

Obsérvese que el enfoque propuesto rescata la idea de Antonelli (Antonelli, 1999) de relacionar al *LEL* con las fichas *CRC*, sólo que en este caso se lo hace a través del *Análisis Esencial* y no a través de los *Escenarios*.

3.1.4 Metodología propuesta

Con esta metodología, en la que se incluye el *Léxico Extendido del Lenguaje*, el *Análisis Esencial* y las *Fichas de Clases, responsabilidades y colaboraciones*, se presenta una secuencia de actividades que resulta muy natural en el proceso de elicitar las características de un sistema y la identificación de los objetos que lo integran. La idea central es la de especificar el sistema a partir de su interacción con el entorno, de acuerdo a la propuesta presentada hace ya más de veinte años por McMenamin y Palmer, mientras que el *LEL* facilita esta tarea y con las fichas *CRC* se organizan sus resultados.

La interpretación de esta metodología requiere la consideración de las tres dimensiones o perspectivas relativamente ortogonales que describen un modelo de objetos:

- *Dimensión funcional*: propiedades relativas a las funciones de transformación que debe desempeñar el sistema.
- *Dimensión estructural*: propiedades estáticas de la estructura del modelo.
- *Dimensión de comportamiento*: propiedades dinámicas que describen el comportamiento individual y colectivo.

Las posibles estrategias generales para la elaboración de un modelo quedan conceptualmente establecidas según el orden con que se definen sus propiedades, y en la Figura 17 se representa la que aquí fue adoptada. Se comienza por la definición funcional del modelo (1), luego se avanza en su definición estática (2) para finalmente considerar sus propiedades dinámicas (3)

Éste aparece como un ordenamiento natural, donde la identificación de las funciones esperadas de un sistema se convierten en el paso previo a la definición de su arquitectura, y donde ambas son necesarias para establecer las cualidades y restricciones que debe mostrar su comportamiento.

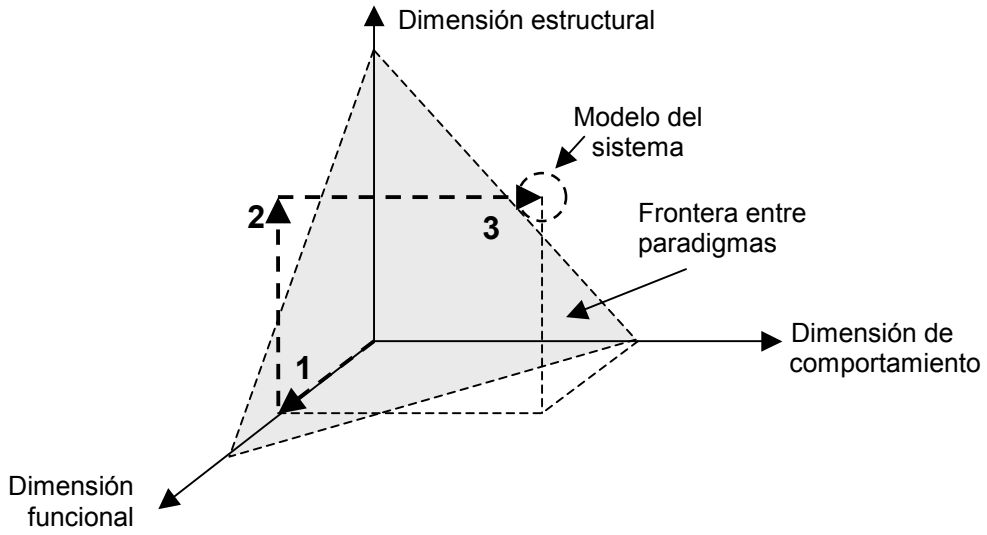


Figura 17: Estrategia general de la metodología propuesta

Una vez definida la estrategia general, deben establecerse los pasos necesarios para la efectiva aplicación de la metodología. Con este fin, se detallan todas las tareas previstas, su ordenamiento y la forma en que se articulan a través de sus resultados, de manera de facilitar el control de la integridad y consistencia del modelo. Esta secuencia de actividades se sintetiza en el esquema presentado en la Figura 18 y se detalla a continuación:

a) Se establecen claramente los límites del sistema.

b) Se identifica el *Universo de Discurso* (*UD*) de la aplicación y se especifican sus términos a través de la descripción de la noción e impacto de cada uno. Se integra así un *Léxico Extendido del Lenguaje* (*LEL*) del problema, que queda documentado como el *Vocabulario del LEL*.

c) Se seleccionan todas las interacciones esperadas del sistema con su ambiente exterior, para lo cual:

   c.1 Se reconocen los símbolos del *LEL* que se refieren a elementos externos al sistema y que impactan sobre el propio sistema.

   c.2 Se identifican los símbolos del *LEL* que pertenecen al sistema y cuyo comportamiento tiene alguna connotación temporal.

d) Se revisa la selección realizada sobre los símbolos del *LEL* para reconocer los *eventos* externos y temporales que estimulan el sistema, lo que lleva a producir

un "*Modelo de Contexto"*. Para ello, es necesario tener en cuenta que un evento debe reunir las siguientes condiciones (Page-Jones, 1992):

- Ocurrir en un cierto momento o instante de tiempo.
- Provenir del entorno del sistema.
- No estar bajo su control.
- Ser relevante.
- Ser reconocible.

Los productos que se obtienen en esta etapa son:

d.1 Un "*Diagrama de Contexto*": se denomina así a un *DFD* que muestra al sistema como una unidad y las líneas de flujo corresponden a los estímulos y las respuestas que lo vinculan con el medio exterior.

d.2 Una "*Tabla de Eventos*": es una tabla en la que se registran los eventos esperados, su tipo, el estímulo asociado, el origen o causante del evento, la *actividad esencial* prevista por el sistema, su respuesta y el destinatario de esa respuesta. La tabla de eventos sirve de base para particionar el sistema en módulos desacoplados que conducen al próximo paso.

e) Se construye un *modelo funcional* fragmentado, que representa internamente al sistema y es, en realidad, un refinamiento del "*Modelo de Contexto*" de la etapa anterior (Yourdon, 1993). Este modelo consiste en:

e.1 Un conjunto de *DFD*s (uno por cada actividad esencial).

e.2 Una descripción de la *Memoria Esencial* (*Modelo de Datos*), que es desarrollada a partir de reconocer la información que debe disponer el sistema para cumplir sus actividades esenciales.

e.3 El *Diccionario de Datos.*

e.4 La Descripción de Procedimientos.

La elaboración de este m*odelo funcional* fragmentado se realiza bajo un proceso de perfeccionamiento progresivo, en el que nuevos eventos y actividades pueden ser reconocidos a medida que se los completa (Ruble, 1997).

f) Se combinan los *DFD*s de cada actividad esencial en un único diagrama que representa al sistema en conjunto desde un punto de vista funcional. Los *DFD* fragmentarios son combinados siguiendo un procedimiento "*button up*" y se obtiene como resultado el llamado "*Diagrama Cero*", también denominado "*modelo particionado de eventos del sistema*", que es el producto final del *Análisis Esencial*.

g) Se realizan los primeros controles de consistencia y completitud del modelo, confrontando las *actividades esenciales* con el *Vocabulario del LEL,* dando lugar a un ciclo de ajuste y perfeccionamiento.

h) A partir del *modelo funcional,* se construye un primer diagrama de *Entidad-Relación*. Aquí debe notarse que los depósitos de datos del DFD, que son particiones de la memoria esencial, se corresponden biunívocamente con las entidades del *DER*, por lo que queda así establecida una relación entre el modelo funcional y un primer modelo estático del sistema.

i) Las actividades esenciales y memoria esencial, reunidos en el "*Diagrama Cero*" y el *DER* proveen el punto de partida para perfeccionar el modelo estático y avanzar hacia un primer análisis orientado a objetos (Wieringa, 1998), que se realiza sobre las fichas *CRC*.

Para ello, se reconoce que cada actividad esencial y sus particiones de la memoria esencial deben estar asociadas a responsabilidades del modelo. Además, estas responsabilidades deben estar a cargo de algún objeto (Rumbaugh et al, 1991) y para reconocer los objetos que tendrán a cargo estas responsabilidades se identifican:

i.1 Elementos del sistema impactados por los eventos.

i.2 Las componentes de los depósitos de datos.

i.3 Otras entradas al diccionario de datos.

i.4 Elementos externos que originan los eventos (objetos ajenos al modelo).

De esta manera, se obtiene la composición estructural del sistema, incluyendo los objetos, las clases a las que pertenecen y sus relaciones.

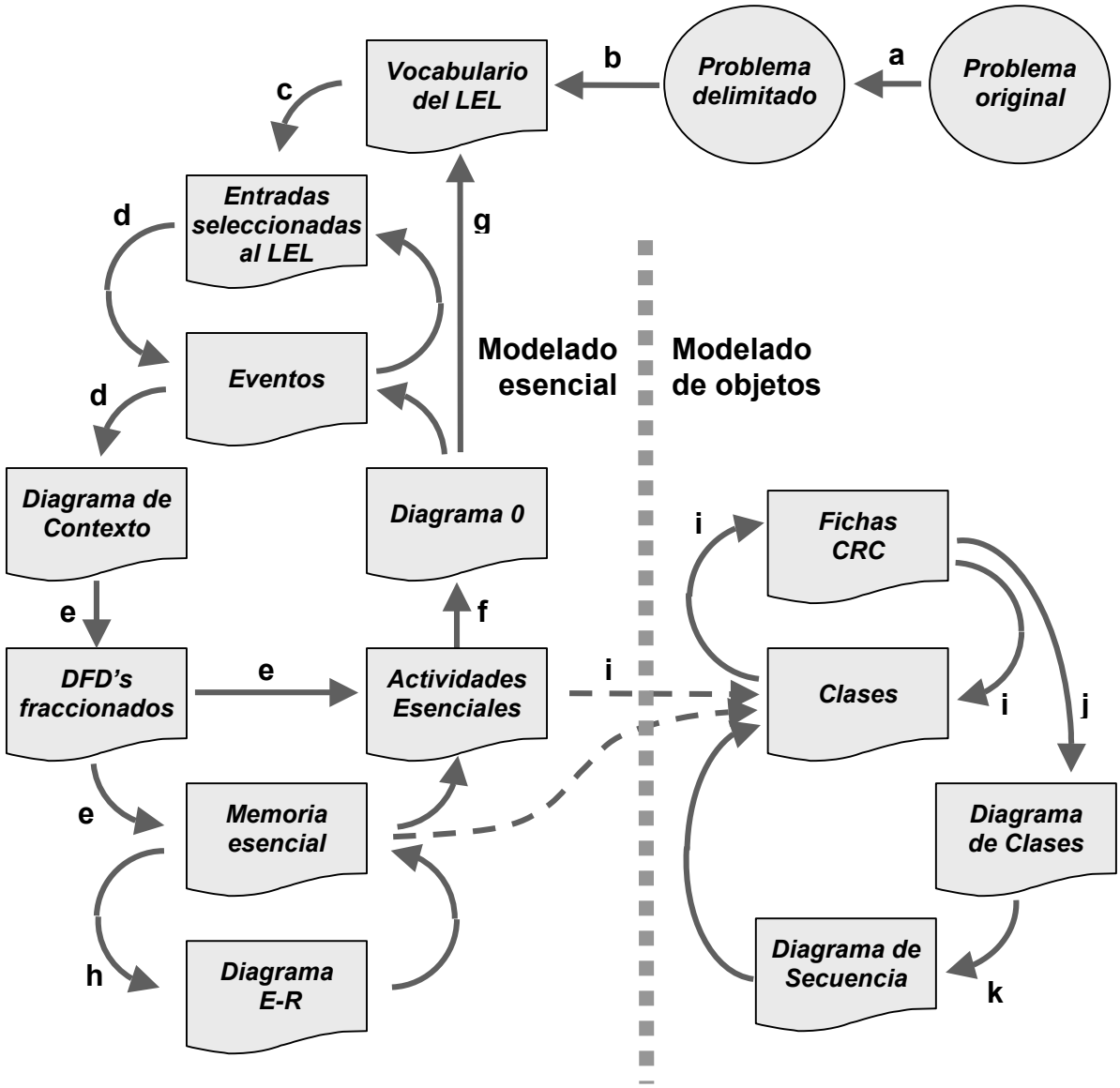


Figura 18: Ajuste progresivo de los Modelos Esenciales y de Objetos

j) Se utiliza la información de las fichas *CRC* para elaborar el *Diagrama de Clases* del sistema.

   Cabe destacar que este modelo no contendrá relaciones innecesarias entre sus objetos, ya que cada ficha CRC sólo referencia a otros objetos en la medida en que los necesita para propósitos específicos.

k) A partir del diagrama de clases y de las exigencias temporales del problema, se confeccionan los diagramas de secuencias. Estos diagramas documentan el comportamiento dinámico del sistema.

Tal como muestra la Figura 18, el procedimiento propuesto incluye, tanto para el modelo esencial como para el modelo de objetos, un trabajo de ciclos de ajuste que tiene por finalidad un perfeccionamiento sucesivo del modelo.

De esta manera, la metodología presentada, a la que se ha denominado *Modelado Esencial de Objetos*, permite completar el proceso de especificación de requerimientos de un sistema en forma natural, racional y rigurosa, lo que reduce los riesgos de omisiones, redundancias o inconsistencias. En la Figura 19 se presenta un esquema alternativo que resume las principales tareas y los productos que son obtenidos de cada una de ellas.

Sin embargo, cabe aquí reconocer que los reales beneficios que esta metodología puede ofrecer, así como sus debilidades, sólo podrán comprobarse con la práctica a través de su aplicación sistemática en problemas de diversa naturaleza, dimensión y complejidad. Por el momento, se procurará demostrar que es apropiada para resolver problemas similares al que aquí se ha planteado; una evaluación completa excede el alcance previsto para este trabajo.

Asimismo, es importante reiterar que, en su conjunto, la metodología propuesta resulta muy apropiada para asegurar la trazabilidad en el proceso que se inicia con el LEL y termina en el modelo de clases y objetos. Debe observarse que se otorga al concepto de trazabilidad el significado amplio al que Leandro Antonelli denomina “traceability” (Antonelli, 1999).

También es posible poner de manifiesto algunas de sus ventajas por comparación con las recomendaciones previstas en otras metodologías. Entre otras, se presenta la fragilidad de enfoque propuesto por Rebecca Wirfs-Brock, donde la identificación de los objetos se apoya en recomendaciones tan subjetivas como “... *ponga énfasis en lo importante y elimine lo irrelevante*” (Wirfs-Brock, 1990:38) y en subrayar los nombres en el texto de los requerimientos para hacer una lista preliminar de objetos.

Tampoco es demasiado convincente la propuesta de Bruce Powel Douglass, quien al no encontrar una forma clara de derivar los objetos desde los Casos de Uso termina recomendando un procedimiento que se apoya también en el subrayado de los nombres (Douglass, 2000:83).

Otras propuestas, como las que evolucionaron a partir del “Object Behavior Analysis” de Kenneth Rubin y Adele Goldberg (Rubin, 1992) apoyan sus fases de modelado en procedimientos mas rigurosos, aunque en estos casos cobra vigencia la ya citada opinión de Bertrand Meyer (Meyer, 1997:739) referida a la necesidad de asegurar su éxito con desarrolladores muy experimentados.

Metodología propuesta

Modelado Esencial de Objetos

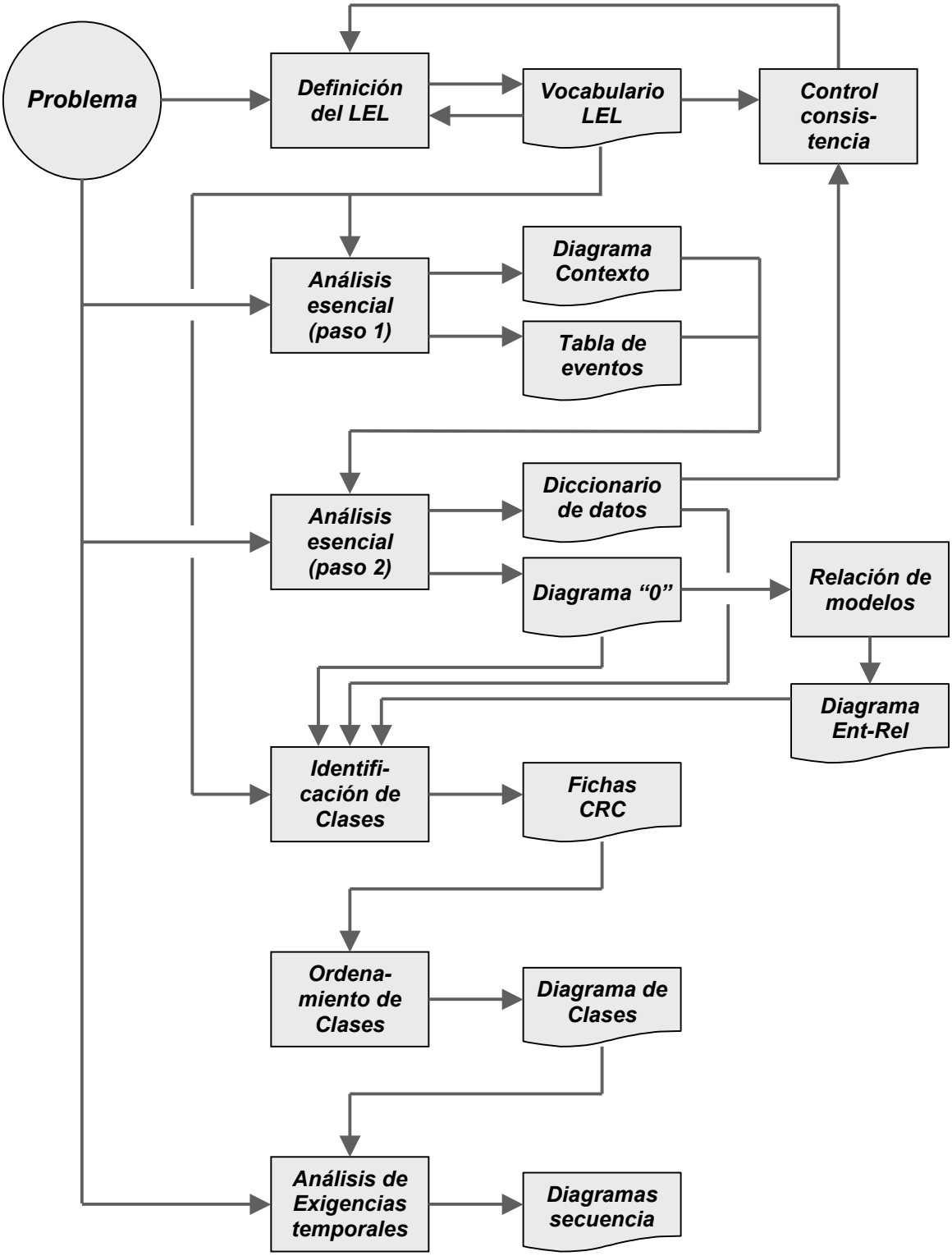


Figura 19 : Proceso de especificación de requerimientos del sistema (SRS) mediante LEL, Análisis Esencial y Fichas CRC

3.2 Diseño

El objetivo de la fase de *diseño* es establecer una solución particular para el problema que ya fue definido y modelado en la etapa anterior de *especificación de requerimientos*. Ésta es una actividad esencialmente creativa que reconoce tres niveles de detalle: 1) *diseño arquitectónico*, 2) *diseño intermedio* y 3) *diseño de detalle*. Todas las propuestas referidas al diseño de sistemas reconocen, con muy pocas diferencias substanciales, estos tres niveles y *UML* no es una excepción a esta regla (Douglass, 2000). Por esta razón y en lo sucesivo se concentrará la atención en los aspectos particulares que están referidos a los sistemas de tiempo real.

3.2.1 Diseño arquitectónico

El *diseño arquitectónico* se corresponde con una fase que tradicionalmente era denominada de "diseño conceptual". Su objetivo es mapear los requerimientos esenciales, que fueron identificados bajo el paradigma de "*tecnología perfecta*", con un modelo que se corresponda con una solución tecnológicamente posible. Se incorpora aquí una actividad creativa que procurará establecer la mejor solución posible al problema planteado. En esta etapa, se tomarán decisiones relativas a la forma en que el sistema será desdoblado en módulos, el alcance de cada uno y la colaboración prevista entre ellos. No se presentan aquí comentarios particulares referidos a los sistemas de tiempo real.

3.2.2 Diseño intermedio

Éste es un nivel intermedio que se ocupa de definir en detalle la colaboración que es necesaria o conveniente entre los objetos del sistema, reconociendo y adoptando patrones cada vez que sea posible.

En el caso de los sistemas de tiempo real, es necesario definir las exigencias temporales de esta colaboración entre objetos, no prestando por ahora atención a la forma en que las necesarias conductas y responsabilidades serán implementadas en las correspondientes clases. La definición de estas exigencias temporales del sistema requiere considerar los siguientes parámetros:

a. *Período de muestreo de señales*: define la frecuencia con que una señal es muestreada. En la elección de los periodos de muestreo debe considerarse la dinámica del proceso y la capacidad disponible en los componentes de hardware (computador, amplificadores, conversores de señales, etc.).
b. *Período de ejecución*: establece los intervalos de tiempo en que las diferentes tareas deben ser ejecutadas.
c. *Plazo de ejecución*: define la duración máxima neta de cada cómputo o tarea. Obviamente, el plazo de ejecución de una tarea debe ser siempre menor a su período.
d. *Atraso de ejecución*: establece la demora máxima admisible entre la orden de inicio de la ejecución de una tarea y el comienzo efectivo de su ejecución.
e. *Tiempo de finalización*: intervalo de tiempo total entre la orden de inicio de una tarea y la finalización de la ejecución.

f. *Niveles de Prioridad*: establece el orden en que son atendidas dos o más tareas que están a la espera de ser ejecutadas.

g. Criticidad de las tareas: debe definir las tareas que pueden ser interrumpidas o suspendidas en caso que se presente un conflicto en la demanda de atención del procesador u otros recursos.

La planificación de la asignación de recursos es sin lugar a dudas uno de los problemas principales en el diseño de sistemas de tiempo real. A los primitivos esquemas de planificación sincrónica, en los que se asignan los recursos en un orden cíclico prefijado, le han seguido esquemas más modernos y complejos de planificación basados en prioridades. Éstos se fundamentan en una asignación estática de prioridades a las distintas tareas y en un "scheduler" que se encarga de ejecutar la tarea de más alta prioridad en cada instante. Entre los diversos enfoques usados, puede mencionarse el *RMS* (*rate monotonic schedulling*) o el *DMS* (*deadline monotonic schedulling*). El primero de ellos plantea una asignación de prioridades en la que, cuanto más corto es su período de ejecución, más alta es la prioridad de la tarea. En el segundo caso, las prioridades se asignan según el tiempo de vencimiento de cada tarea. Estas estrategias RMS y DMS, al igual que otras similares, se basan en una serie de suposiciones relativas a la naturaleza de las tareas en conflicto y sus relaciones, las cuales pueden no verificarse estrictamente en la realidad.

Los diagramas de secuencias son el medio apropiado para documentar estas relaciones temporales, debiéndose incluir un detalle de las posibles contingencias, considerando en todos los casos las condiciones más desfavorables (Gao, 2003).

3.2.3 Diseño detallado

En este nivel, se definen detalles relativos a la definición de los objetos, incluyendo: 1) constantes de inicialización, 2) estructura de sus atributos, 3) protocolos de mensajes reconocidos, 4) métodos disponibles, 5) colaboraciones requeridas, 6) protocolos de respuestas previstas, 7) procedimientos o algoritmos previstos y 8) manejo de excepciones.

En el diseño de sistemas de tiempo real los diagramas de estados (máquinas secuenciales o autómatas finitos) constituyen una forma apropiada de representar el comportamiento interno de los objetos, y para ello *UML* ha previsto numerosas extensiones a los diagramas de estados convencionales (Del Bianco, 2003).

3.2.4 Conectividad externa

En la actualidad, un aspecto que no puede ser omitido en el diseño de un sistema es su conectividad y éste debe ser contemplado como parte del diseño arquitectónico e intermedio. Las aplicaciones industriales no son una excepción a esta regla tácita que establece que todos los sistemas han dejado de estar aislados para formar parte de redes, en lo que se conoce como la "*era de la comunicación*". Así, se verifica una fuerte tendencia a la interconexión de todos los equipos de las plantas industriales, conformando lo que se denomina un "*piso de planta*" (*Factory Floor*). Para ello, se implementan sistemas de supervisión y control que hacen posible el seguimiento de productos desde su montaje inicial hasta el despacho al cliente, interconectados con los sistemas de ingeniería, control de calidad, logística,

gestión administrativa y post venta, conformando lo que suele denominarse un "*Sistema Integrado de Manufactura"*.

Si el simulador que es objeto de este trabajo tuviese como destino el área de ingeniería de una industria, sería sin lugar a dudas necesaria su interconexión con los equipos reales que son representados, a efectos de disponer de datos de ensayos para perfeccionar y actualizar sus modelos. Por tal motivo, parece oportuno hacer referencia a las opciones de conectividad a pesar de que su implementación quede por el momento fuera del alcance de este trabajo.

En el campo de la conectividad de equipos industriales, pueden distinguirse dos épocas, que muestran las siguientes características:

a. *Interconexiones especiales*: se opera con conexiones en bus lineal sobre norma RS-485 o IEEE488 (GPIB) y protocolos acordes a las recomendaciones *OSI* (*Open Systems Interconectivity*) propios de cada fabricante de equipos. Si bien estas soluciones tienen un excelente desempeño en ambientes con ruidos de radiofrecuencia, su capacidad de transmisión de datos es relativamente baja y cada fabricante implementa su propio protocolo, dificultando la intercomunicación entre equipos de diferente procedencia. Puede citarse como ejemplo el protocolo UniTelway de Telemecanique.

b. *Interconexiones normalizadas*: para resolver este problema fue desarrollado un standard de conectividad industrial denominado *OPC* (*OLE for Process Control*) que es administrado por una fundación (*OPC Foundation*) y tiene el respaldo de más de 200 compañías del mercado, incluyendo todas las empresas líderes. Se trata de un protocolo destinado a comunicar numerosas fuentes de datos, incluyendo hardware de tipo industrial, bases de datos y sistemas de control de procesos. Se caracteriza por ofrecer una arquitectura Cliente/Servidor de comunicación abierta que soporta tanto interfases locales como remotas. Estos sistemas operan sobre vínculos TSP/IP.

En razón de lo expuesto, es obvio que la opción de dotar al simulador de la capacidad de comportarse como un *Cliente OPC* parece la más conveniente. Para ello, deben ser desarrolladas las correspondientes interfases, de conformidad con lo establecido por las normas específicas (*OPC Data Access Version 2.0*), y debe disponerse además de un servidor *OPC* (*KepServer OPC*). En su implementación deben preverse comandos de configuración, consulta y operación remota. Una alternativa que implicaría menor esfuerzo de desarrollo consistiría en una interfase *SQL* con alguna base de datos a través de *ODBC,* siendo quizás la mejor solución disponer de ambas posibilidades. Con esta combinación, se dispondría de acceso directo a bases de datos y la capacidad de *cliente OPC* para operar en red.

3.3 Programación

El producto final de todo proceso de desarrollo de software es un *programa* que debe ser escrito en cierto lenguaje y deberá operar sobre cierta plataforma o sistema operativo. La mejor especificación de requerimientos y el más oportuno diseño serán de poca utilidad si esta secuencia no culmina en un *programa* que sea capaz de cumplir correctamente con las funciones esperadas. Esta afirmación, en apariencia trivial e innecesaria, es olvidada con mucha más frecuencia de lo imaginable y de ello abundan testimonios en el mercado.

Si bien la selección de un lenguaje y de un sistema operativo es de por sí todo un tema, este trabajo alcanza connotaciones particulares cuando se trata de implementar sistemas de tiempo real. A continuación, se hará por lo tanto referencia a las exigencias particulares que impone este tipo de sistemas sobre los lenguajes y sistemas operativos.

3.3.1 Lenguajes de programación

La elección del lenguaje es un problema histórico que enfrentan los programadores desde que, al primer compilador Fortran de 1957, se le sumaron otros lenguajes como alternativas, tales como el Algol en 1958 o el Cobol en1960. Un viejo adagio, que no deja de expresar una verdad, procura dar respuesta a este interrogante postulando que el mejor lenguaje es aquél que sea el más conocido por el programador. Dejando de lado el mensaje literal, que parece excesivamente conservador, debe rescatarse la idea central de que un programador sólo cumplirá correctamente su tarea en la medida en que conozca profundamente el lenguaje que utiliza.

Esta incuestionable afirmación se contradice con la tendencia actual del mercado de los lenguajes, de ofrecer nuevas versiones con una frecuencia mayor que la que permitiría su efectivo aprendizaje. Asistimos así, como en casi todos los órdenes de la sociedad moderna, a la profusión de "*programadores light*", dispuestos a emplear la última versión de su lenguaje favorito, aunque incapaces de resolver racionalmente la menor dificultad que se les presente.

Afortunadamente, en el caso de los sistemas de tiempo real este problema de selección de lenguajes se simplifica, ya que los sistemas operativos apropiados para estas aplicaciones ofrecen por lo general pocas alternativas.

Más específicamente, puede señalarse que los lenguajes apropiados para implementar sistemas de tiempo real deben exhibir buenos desempeños con relación a los siguientes atributos:

a. Control de Tiempo
b. Concurrencia
c. Seguridad
d. Portabilidad
e. Eficiencia

Procurando ahora identificar los lenguajes más utilizados, deben distinguirse los siguientes dos casos:

a. *Sistemas pequeños*: su programación se realiza en lenguajes Assembler, C o PL/M y operan directamente sobre microcontroladores, en ausencia de sistema operativo.
b. *Sistemas mayores*: las aplicaciones se implementan en ANSI C, C++, Ada, Delphi o Java. El principal problema de C++ es que, pese a su gran difusión, su falta de normalización efectiva le provoca posteriores conflictos de integración. Por otra parte, en los últimos años se han definido especificaciones tales como *RTSJ* (*Real Time Specificaction for Java)* que procuran superar las limitaciones que ofrecía Java en este tipo de aplicaciones (Bollela, 2000).

3.3.2 Sistemas operativos

Para merecer la calificación de *RTOS* (*Real-Time Operating System*) un sistema operativo debe cumplir numerosas condiciones, destacándose las siguientes tres características que son consideradas básicas:

a) *Determinismo*: se dice que un sistema operativo es determinista cuando se puede calcular el máximo tiempo que puede demandar una invocación a cualquier recurso del sistema.

b) *Latencia de interrupciones*: se denomina "*latencia de interrupciones*" al máximo tiempo que transcurre desde que una señal de interrupción llega al procesador hasta que es ejecutada su *ISR* (*Interrupt Service Routine*) asociada. Las interrupciones son elementos clave de los *RTOS*. Cuando una interrupción tiene lugar, el procesador debe realizar varias tareas antes de ejecutar su *ISR* asociada, tales como: 1) identificar la interrupción, 2) comprobar su prioridad según el esquema de prioridades vigente y 3) preservar la actividad actual en caso que deba ser interrumpida.

c) *Cambio de contexto*: se refiere al intervalo de tiempo que es necesario para realizar un cambio de contexto (*context switch* o *thread switch)*, denominándose así al proceso de cambiar de una tarea a otra.

Las demás condiciones están referidas a su capacidad para acceder a los recursos del sistema, cantidad de memoria requerida, sistema de protección de memoria, arquitectura micro kernel y entorno de desarrollo apropiado. Además, los sistemas operativos de tiempo real modernos deben cumplir las recomendaciones *POSIX* (*Portable Operating System Interface X*) del *IEEE* que procura ofrecer un entorno normalizado de recursos básicos a los programadores.

A partir de las características enunciadas, se deduce que se trata de sistemas operativos especiales, altamente especializados, de escasa difusión y habitualmente de elevado costo. En este último caso, se trata de productos comerciales, aunque debe destacarse que hay también numerosos sistemas operativos de tiempo real que son desarrollados por universidades y centros de investigación. Uno de los clásicos que corresponde al primer grupo es el *QNX-Neutrino*, que además de cumplir con las normas *POSIX,* exhibe características de multitarea, multiusuario y multiplataforma.

Las circunstancias señaladas y la creciente demanda de este tipo de sistemas han contribuido a que numerosas aplicaciones de tiempo real sean implementadas sobre sistemas operativos convencionales, especialmente *Windows*. Esta tendencia se ve favorecida por la gran difusión y aceptación que tienen las diversas versiones de *Windows* en los ámbitos industriales. Estos sistemas pueden ser denominados de "*cuasi Tiempo Real*" y son empleados en aplicaciones tolerantes a cierta flexibilidad en el manejo de los tiempos que ya fueron calificadas como básicos. Aun aceptando que no se trata aquí de verdaderos sistemas de tiempo real, el masivo incremento de estas aplicaciones sobre entornos *Windows* es preocupante, tanto por la falta de fiabilidad y robustez que son características de estos sistemas operativos, como también por los desmedidos recursos de hardware que demandan.

Para la implemetación de estos sistemas de "*cuasi Tiempo Real*" el clásico *MS-DOS* era una excelente opción, que lamentablemente ha perdido vigencia por su falta de compatibilidad con los recursos disponibles en la actualidad. Por otro lado, es posible reconocer a *Windows* como una opción aceptable para llevar adelante

procesos de simulación y en razón de su amplia disponibilidad es adoptado para este trabajo.

3.4 Verificación y validación

La fase final de desarrollo de un proyecto exige someter la primera versión del producto a una serie planificada de pruebas con el objeto de comprobar su correcto funcionamiento bajo todas las condiciones posibles de operación.

Sin embargo, las pruebas de verificación y validación deben comenzar antes, con el fin de comprobar que los métodos de cada objeto cumplan individualmente con sus responsabilidades. Se trata de un control botton-up que debe comenzar con los métodos más elementales y ascender en la estructura del sistema hasta llegar a la evaluación de todo el sistema en conjunto, tal como es representado en el esquema de la Figura 20. Este control progresivo, que es recomendable en todo sistema, es esencial en sistemas técnicos donde sus métodos son implementados a través de complejos algoritmos.

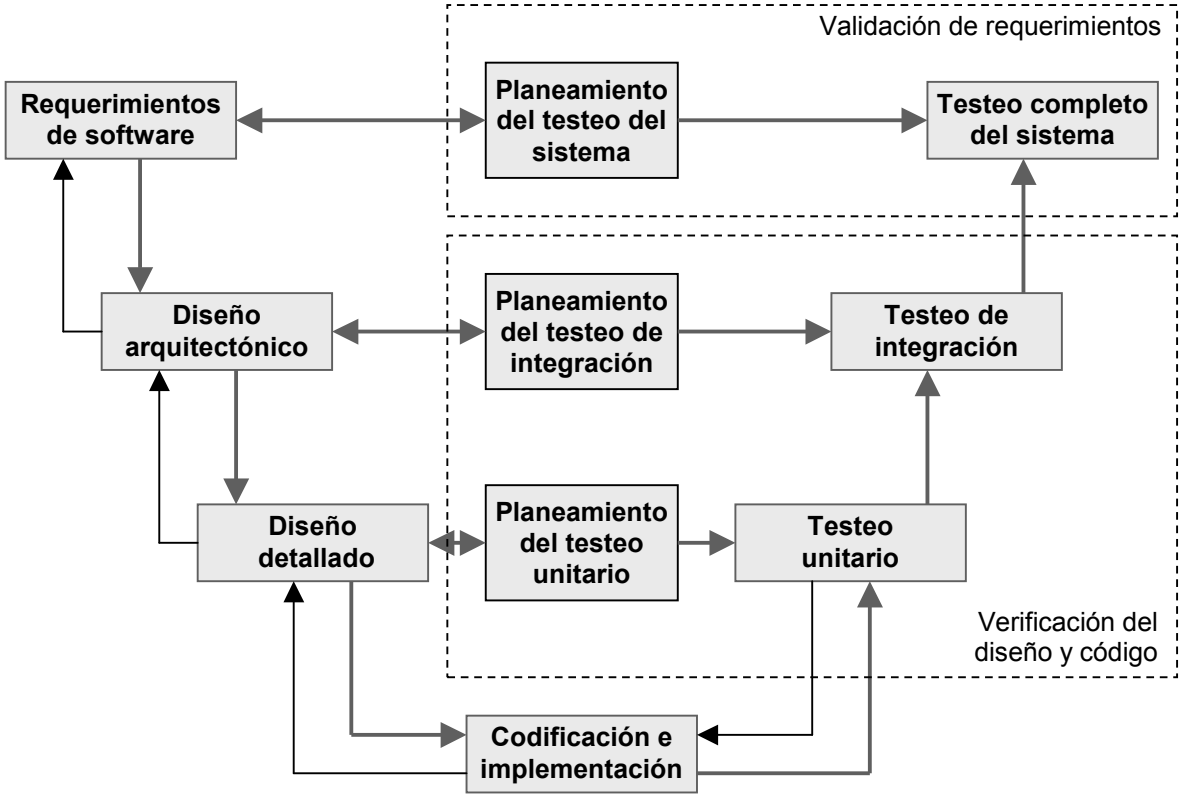


Figura 20: El modelo en “V” de desarrollo de software

En este esquema, que representa el modelo en “V” de desarrollo de software (Oliveros, 2001), se muestran con claridad los lugares que ocupan los procesos de verificación y validación de un sistema.

A propósito, parece aquí conveniente volver a las definiciones de los conceptos de verificación y validación, recurriendo para ello nuevamente a los clásicos interrogantes propuestos por Boehm (Boehm, 1984):

Verificación: ¿ Estoy construyendo correctamente el producto?

Validación : ¿ Estoy construyendo el producto correcto?

De manera natural, se asocia la validación y verificación de un programa con un proceso, adecuadamente planificado, que está destinado a comprobar su correcto funcionamiento. Sin embargo, no puede el testeo dinámico ser usado como único elemento para certificar la corrección de un programa, ya que éste sólo podrá mostrar la presencia de errores pero no garantizar la ausencia de ellos. En efecto, en la mayoría de los casos es imposible asegurar que han sido verificadas todas las posibles combinaciones de condiciones de operación, por lo que el testeo dinámico es una condición necesaria pero no suficiente de corrección.

Para superar esta dificultad, las propuestas se vienen orientando a obtener suficiente conocimiento sobre el comportamiento de los programas a partir del examen anticipado de sus diseños y códigos, en lugar de hacerlo a partir del resultado de sus ejecuciones. Se recurre a la *inspección y verificación estática* para las comprobaciones de corrección.

Se llega así al enfoque moderno de la revisión de un programa*,* que se refiere al proceso de demostrar anticipadamente que los resultados que proporcionará serán los deseados, o bien detectar la presencia de eventuales errores e identificarlos completamente. Este proceso está orientado a comprobar esta propiedad esencial de todo programa que es su *corrección* y que puede definirse como "la capacidad de los productos de software de ejecutar sus tareas en concordancia con las definiciones de sus requerimientos y especificaciones".

En este contexto, la *verificación* formal de programas está destinada a demostrar, usando argumentos matemáticos, que un programa satisface su especificación. Una especificación (E1, E2) de un programa P, establece que si P comienza a funcionar en un estado que satisface la precondición E1, entonces termina, en un tiempo finito, en un estado que satisface la condición final E2. La *corrección* se establece a través de la correspondencia entre el comportamiento deseado del programa y su comportamiento real, donde el comportamiento deseado es definido mediante especificaciones y el comportamiento real debe deducirse mediante el análisis del texto. Este análisis se hace en términos de las componentes elementales que conforman el programa. En resumen, todo proceso de verificación formal debe estar dirigido a comprobar algunos aspectos esenciales tales como que la ejecución de un programa siempre termina y que su código es consistente con su especificación.

Es aquí importante establecer dos condiciones que son necesarias para poder realizar una verificación formal basada en un enfoque matemático: a) todas las características del lenguaje de programación, sintácticas y semánticas, tiene que estar definidas formalmente, y b) el programa debe especificarse con una notación que sea consistente con la técnica de verificación matemática usada.

Dadas las exigencias de robustez, la verificación y validación de los sistemas de tiempo real deben ser cuidadosamente planificadas y minuciosamente realizadas. No tienen, sin embargo, especificaciones particulares y su proceso de testeo debe ser conducido en concordancia con las normas establecidas, entre las que se destacan:

- IEEE 829 : Standard para la documentación de los test de software
- IEEE 1008 : Standard para el test de unidades
- IEEE 1028 : Standard para las revisiones de software
- IEEE 1044 : Standard para la clasificación de anomalías de software
- BSS 7925-2 : Standard para el test de componentes de software

4 Definición de requerimientos

A continuación, se comienza por describir las connotaciones propias del objeto de estudio, con el fin de presentar y conocer el problema desde sus diferentes puntos de vista, para finalmente especificar sus requerimientos. Al hacerlo, se evitan los tecnicismos propios de la mecánica que sean innecesarios para el objetivo de este trabajo.

4.1 Ensayos de motores

Una particularidad de los motores de combustión interna es su permanente vigencia. Su historia de más de cien años muestra una progresiva e ininterrumpida mejora en sus prestaciones y rendimientos, a pesar de mantenerse inalterada la concepción mecánica básica que fue anticipada por Beau de Rochas en 1862 y materializada por August Otto en su exitoso primer motor de cuatro tiempos en 1876. Esto no fue obviamente fruto de la casualidad. Mas bien fue el resultado de un persistente y esmerado trabajo de ingeniería, que con un claro enfoque multidisciplinario, perfeccionó todos los detalles del motor desde los puntos de vista de la cinemática, dinámica, transferencia de calor, termodinámica, metalurgia, resistencia de materiales y fluidodinámica, entre otros. Y acompañando esta evolución fue necesario desarrollar y perfeccionar equipos y técnicas de ensayo, ya que no había otra forma de comprobar los resultados de las sucesivas mejoras que no fuese a través de los ensayos experimentales.

En efecto, si bien la secuencia de operaciones de un motor en funcionamiento está perfectamente definida y tiene un sólido respaldo conceptual, los fenómenos físicos que se presentan son lo suficientemente complejos como para que el desarrollo de los motores haya debido estar siempre apoyado en los test de prototipos.

Es así que los motores de combustión interna fueron y son ensayados experimentalmente. A las mediciones tradicionales de torque, potencia y consumo, entre otras, se suma el análisis de la composición de los gases de escape, que por sus características contaminantes viene adquiriendo creciente importancia. En la actualidad, estas pruebas se realizan en forma automática y para ello son requisitos esenciales tanto la operación del motor en condiciones preestablecidas de velocidad y carga, con un margen de error muy estrecho, como la posibilidad de pasar con gran agilidad de una condición de operación a otra.

Las instalaciones para cumplir con tales exigencias, denominadas “bancos de ensayos”, están dotadas de una bancada para fijar el motor a ser ensayado, un freno dinamométrico, una transmisión cardánica para vincular el motor y el freno, numerosos sensores destinados a medir parámetros de variada naturaleza, un actuador electromecánico para operar el acelerador y un sistema computarizado de supervisión y control. A estos elementos, deben agregarse los circuitos de combustible, refrigeración, ventilación y sistemas auxiliares.

Este conjunto de un motor acoplado a un freno encontrará su equilibrio a diferentes velocidades cuando en un régimen estacionario el torque entregado por el motor iguale la carga absorbida por el freno. Puede establecerse para ello que el motor desarrollará su torque según una familia de curvas, cada una de las cuales corresponde a una condición de alimentación dada por la posición de la bomba inyectora. A su vez, la capacidad de frenado del dinamómetro responderá a otra

familia de curvas, donde cada una está asociada a la tensión aplicada en su bobinado en el caso del freno eléctrico o del caudal de circulación de agua en el caso del freno hidráulico.

Para alcanzar rápidamente, en todas las condiciones de marcha, la requerida estabilidad del conjunto motor-freno es entonces necesario un apropiado sistema de control y un cuidadoso ajuste de sus parámetros. Este sistema operará para ello sobre las dos variables básicas de control, que son la alimentación de combustible al motor y la capacidad de frenado del dinamómetro. Aquí, debe acotarse que en un mismo banco se ensayan habitualmente motores de diversa potencia, que un mismo motor cambia su respuesta según la disponibilidad de algunos accesorios, tales como el turbocompresor o el Intercooler, y que aún con los mismos accesorios, debido a su regulación electrónica, un motor suele mostrar comportamientos muy diferentes a distintos regímenes de velocidad. Por todo ello, el requerido ajuste del sistema de control suele en algunos casos ser una tarea muy laboriosa.

No son, sin embargo, los ensayos la solución a todos los problemas. Muy por el contrario, en muchos casos debe reconocerse que son más bien una nueva e inevitable fuente de dificultades. En una breve enumeración, pueden mencionarse las dificultades inherentes a la medición de parámetros, el tiempo demandado por la construcción de los prototipos e instalaciones de ensayo, el tiempo requerido por los propios ensayos, la necesaria y cuidadosa interpretación de los resultados y los elevados costos involucrados. También es necesario advertir que en estos ensayos se conjugan algunos factores claramente contradictorios. En efecto, se pretende utilizar equipos electrónicos sofisticados para medir señales físicas con elevado nivel de precisión y alcanzar condiciones de marcha estacionaria con márgenes de error muy estrechos, todo ello en un ambiente con muy altas temperaturas, presencia de gases de combustión y elevados niveles de contaminación sonora y vibratoria.

Por otra parte, la complejidad alcanzada en algunos casos por los motores y sus sistemas de control es tan grande que las metodologías tradicionales de diseño basadas en uso intensivo de ensayos han quedado prácticamente obsoletas. Así, al considerarse el alto valor práctico que indudablemente tiene aquí la validación dinámica, es decir las pruebas de buen funcionamiento, debe tenerse presente que sólo es posible asegurar la corrección de un diseño a partir de haber evaluado experimentalmente todas las condiciones de operación posibles. Esto lleva a reiterar que la validación dinámica es una excelente herramienta para comprobar la presencia de errores, pero en la mayoría de los casos no es suficiente para asegurar la ausencia de ellos.

4.2 Simulación de ensayos

Por todo lo expuesto, no debe sorprender que siempre haya habido un marcado interés por desarrollar modelos matemáticos capaces de representar los fenómenos físicos involucrados en el funcionamiento de motores. Tampoco debe sorprender que la aparición y perfeccionamiento de las computadoras hayan servido de estimulo a la mejora de tales modelos y a integrar estos modelos en procesos de simulación. En efecto, se comprendió desde un principio el incalculable valor que tendría la posibilidad de evaluar el funcionamiento de motores a través de modelos numéricos correctos y confiables.

Es así que la posibilidad de conducir estos procesos de simulación llevó a replantear el objeto de los ensayos, ya que en muchos casos su finalidad pasó a ser la de suministrar datos para ajustar y validar el desempeño de los modelos, convirtiéndose estos últimos en las herramientas centrales del proceso de diseño.

Además, el desarrollo de modelos y los procesos de simulación trajeron aparejados algunos beneficios colaterales muy convenientes (Boehm, 1973), tales como:

a. La necesidad de definirlos llevó a una mejor comprensión de los fenómenos físicos involucrados y la interacción entre ellos.
b. Se sistematizó el conocimiento sobre el objeto estudiado y se lo almacenó en forma accesible.
c. Se identificaron áreas sobre las que había insuficiente conocimiento y se estimularon nuevas líneas de investigación.
d. Se aprovecharon mejor los ensayos, reduciéndose la cantidad requerida y favoreciéndose la optimización de los diseños.

La necesidad de asegurar las condiciones para que un modelo y sus resultados reproduzcan acabadamente la esencia y conductas del objeto real, que es estudiado, se explica si se entiende que la simulación de un sistema es la manipulación experimental por computadora de un modelo lógico-matemático. Más aún, en caso de tratarse de sistemas de tiempo real estas conductas deben ser reproducidas estrictamente, sin distorsiones en el tiempo, por lo que esta última exigencia se suma a las demás condiciones de diseño, propias del paradigma de objetos.

Sin embargo, también es necesario destacar que el mejor modelo no es el más preciso ni el más completo, sino más bien el que represente las variables esenciales del sistema con la precisión suficiente. Es así que la diversidad de modelos empleados para simular motores pueden ser, en primera instancia, agrupados como "*cajas blancas", "grises" y "negras":*

Los modelos de "*cajas blancas"* son usados cuando se conoce profundamente el fenómeno físico, se dispone de herramientas de análisis numérico basadas en el método de los elementos finitos –*FEM*- y se cuenta con un gran poder computacional. Normalmente, estos modelos tridimensionales presentan gran complejidad geométrica y están destinados a reproducir fenómenos de interacción entre medios continuos, tales como fluido-dinámicos, termodinámicos y de sólidos elásticos. Su objetivo final es predecir la performance del motor a partir de su geometría y condiciones de funcionamiento.

Hay casos en que estos modelos se convierten progresivamente en "*cajas grises"* o *"negras",* a medida que ciertos datos o la falta de conocimiento suficiente sobre los fenómenos físicos involucrados deben ser reemplazados por información experimental. En la selección del tipo de representación más apropiada intervienen también, como factores condicionantes, las exigencias funcionales del sistema representado, como son el tiempo de respuesta o la predictibilidad de sus resultados.

Una de las líneas de investigación más importantes en el campo de la simulación de motores se viene desarrollando en el Departamento de Ingeniería Mecánica de la Universidad de Salerno, Italia. Los modelos allí empleados se

apoyan en redes neuronales multicapa de perceptrones (Arsie, 2001), que a partir de algunos parámetros de entrada permiten predecir el desempeño del motor. Se trata de modelos de "caja negra", que son entrenados mediante el procedimiento de *backpropagation* (Arsie, 1998) con conjuntos de datos provenientes de los resultados de ensayos. Estos modelos han demostrado sus ventajas en la representación de fenómenos no-lineales y en aquellos casos en que los datos disponibles presentan un nivel de incerteza importante. Éste y otros modelos similares permiten hacer un máximo aprovechamiento de la información obtenida de los ensayos, brindando resultados relativos a la potencia, composición de las emisiones del motor, consumo u otros parámetros y permitiendo simular estrategias de control.

En otros casos, se representa el funcionamiento de estos conjuntos a partir de la combinación de modelos más detallados de los diferentes componentes del motor. Por ejemplo, se evalúa el torque generado por cada pistón en cada momento a partir de: 1) la fase en el ciclo del motor (admisión, compresión, explosión y escape), 2) la posición del pistón, 3) la masa de aire cargada en el cilindro, 4) la cantidad de combustible inyectado y 5) el tiempo de encendido. La secuencia de condiciones en cada cilindro es determinada por un autómata finito y su contribución es incorporada al conjunto de acuerdo a su orden de encendido y la velocidad del motor (Balluchi, 2000). En estos modelos de "cajas grises", los valores representativos del comportamiento del motor en el tiempo son obtenidos a partir de la combinación de las contribuciones de sus distintos componentes.

4.3 Dominios involucrados

El conjunto controlado y el sistema de control pertenecen a dos dominios diferentes, que son el mecánico y el informático. Tal como se muestra en la figura 21, estos dominios se valen a su vez de un tercer dominio para comunicarse entre sí: el eléctrico / electrónico.

Figura 21: Dominios del problema estudiado

En el dominio de la mecánica, las señales primitivas corresponden a desplazamientos lineales, desplazamientos angulares, temperaturas y fuerzas; mientras que en el dominio eléctrico corresponden a corrientes y tensiones. Finalmente, en el dominio informático se representan y operan todas las señales anteriores a través de números binarios y el álgebra discreta. Por otra parte, debido a la naturaleza dinámica de los fenómenos estudiados se deberá incorporar el tiempo como una variable común a todos estos dominios.

Aquí cabe recordar que, desde la remota antigüedad y hasta no hace mucho tiempo, toda la actividad de medición y control se desarrollaba íntegramente en el dominio de la mecánica. Testimonios de lo primero son los manómetros de columnas de agua, las balanzas de resortes y los termómetros de mercurio. Por su parte, para implementar acciones de control con recursos exclusivamente mecánicos se

diseñaron dispositivos muy diversos e ingeniosos, como es el caso del regulador de velocidad de Watt de las máquinas de vapor. Estos dispositivos apoyaban su funcionamiento en las leyes fundamentales de la física y hay testimonios de que algunos de ellos ya fueron usados por los griegos varios cientos de años antes de Cristo. Mucho más recientemente, la revolución industrial continuó usando y perfeccionó estos dispositivos mecánicos, los que permanecieron en vigencia hasta bien entrado el siglo XX.

Mientras tanto, desde finales del siglo XIX se fue progresivamente consolidando el dominio de la electricidad y luego de la electrónica como auxiliares esenciales para las mediciones y el control. Esto trajo naturalmente aparejado, por primera vez, la necesidad de convertir señales de distinta naturaleza, es decir de uno a otro dominio. Para convertir señales mecánicas en eléctricas se desarrollaron sensores, como es el caso de las termocuplas, termorresistencias, strain gages, sensores piezoeléctricos y fotocélulas, entre otros. En sentido contrario, es decir para convertir señales eléctricas en mecánicas, pueden como ejemplo mencionarse las resistencias eléctricas y una amplia diversidad de dispositivos electromagnéticos entre los que se destacan actuadores y motores de distinto tipo. Las acciones de control se generaban en forma de señales eléctricas a partir de temporizadores, contactores y potenciómetros y eran convertidas en calor o movimiento a través de esos dispositivos.

La incorporación del dominio informático en la actividad de control tiene numerosos precursores. Entre ellos, no pueden dejar de mencionarse las publicaciones prácticamente simultáneas de Alan Turing y Emil Prost de 1936. Estos investigadores presentaron, en forma independiente, formulaciones equivalentes de máquinas abstractas capaces de cumplir procedimientos efectivos a condición de ser adecuadamente programadas (Arbib, 1987). Entre las otras muchas contribuciones que siguieron, merece en particular citarse el esfuerzo de Norbert Wiener por reunir el control realimentado, los sistemas y las comunicaciones en una nueva ciencia denominada *cibernética* (Wiener, 1948). A partir de ellos, se asistió a una rápida evolución de la ciencia de la computación, la teoría de control y la tecnología que en algo más de cincuenta años ha llevado a alcanzar la situación actual, en la que parece impensable un dispositivo de medición y control en el cual la informática esté ausente. Con esta evolución, volvió a presentarse la necesidad de convertir eficientemente señales entre diferentes dominios: en este caso, entre el eléctrico y el informático. Para cumplir esta función, en uno y otro sentido, hicieron su aparición los conversores Análogo-Digital (A/D) y Digital-Analógico (D/A).

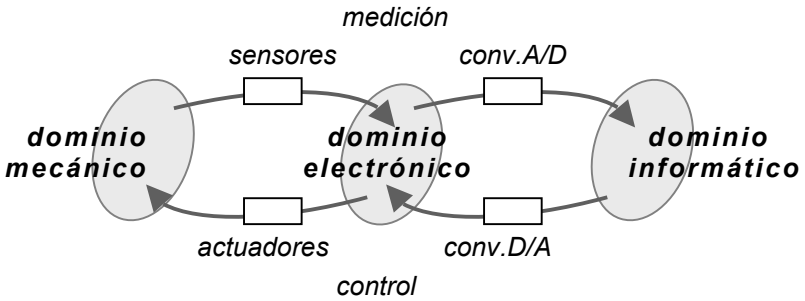


Figura 22: Interfases entre los dominios

Tal como se presenta en la figura 22, se completa así un ciclo cerrado que involucra los tres dominios, en el que a través de acciones informáticas se monitorea, evalúa y controla el funcionamiento de un cierto sistema mecánico.

4.4 Identificación de los componentes principales

Los principales elementos involucrados y los vínculos entre ellos están presentados en el esquema general de la Figura 23. Tal como se verá más adelante, en ausencia de acciones exteriores y una cierta velocidad el conjunto motor-freno tendrá una operación estable.

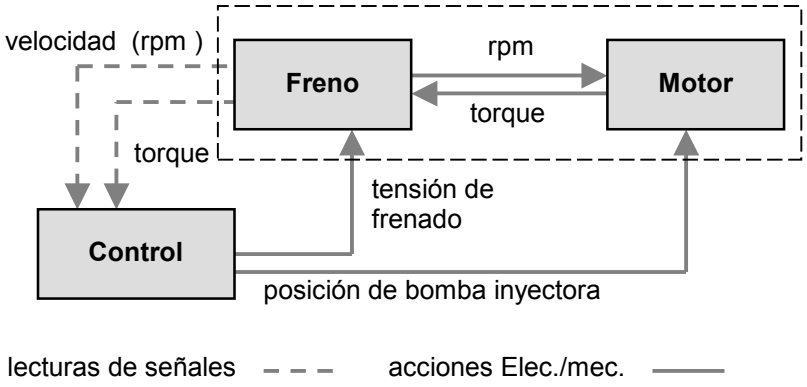


Figura 23 – Esquema general de los componentes del sistema

El objeto del sistema de control es definir las condiciones para que esta operación estable tenga lugar a una velocidad predeterminada, o a una sucesión de ellas, pudiendo así puntualizarse los dos requerimientos que se imponen sobre esta unidad de control:

- Operar el motor en régimen estacionario a una cierta velocidad.
- Posibilitar un rápido cambio de un régimen de operación a otro.

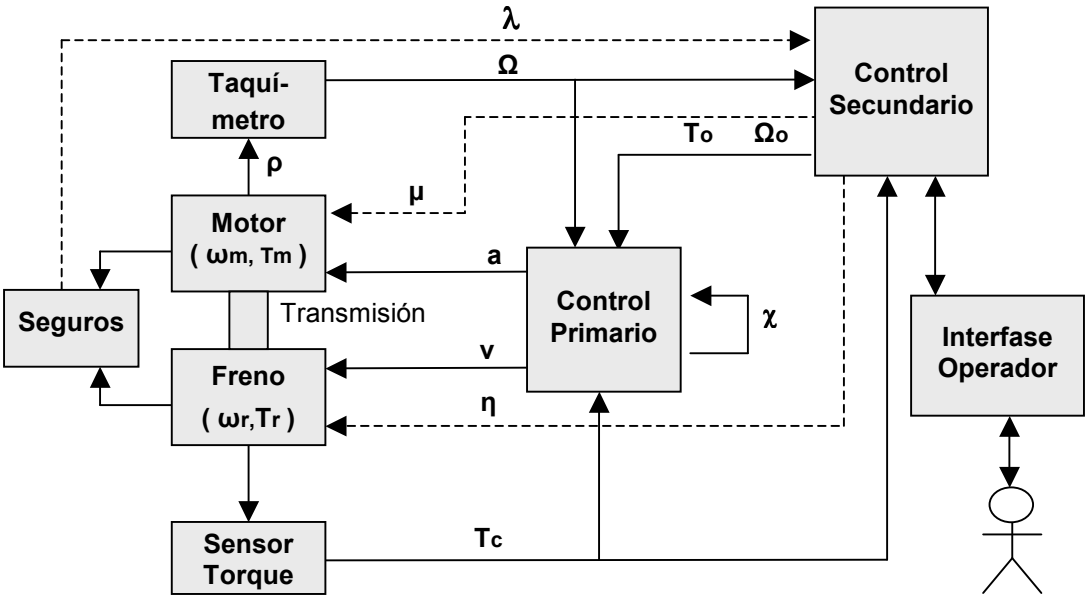


Figura 24 – Esquema detallado del Sistema

Al entrarse en mayor nivel de detalle deben distinguirse los siguientes elementos representados en la Figura 24: motor, freno, acoplamiento entre motor y freno, seguros, sensor de velocidad, sensor de torque, unidades de control primario, unidad de control secundario e interfase con el operador. En el mismo esquema se

identifican las señales que vinculan los diferentes elementos, con una simbología que se mantendrá en el resto de este trabajo y se define en el anexo A.

Las principales características de todos estos componentes son descriptas en forma individual en los puntos siguientes. Entre éstos, se destacan las unidades de control primario y de control secundario, que conforman lo que se denomina *sistema digital de control* y cuyo estudio es el objeto de este trabajo.

4.4.1 Motor

La finalidad de un motor es entregar un torque a un eje. En el caso de un motor de combustión interna el valor de este torque dependerá: 1) de las características intrínsecas del propio motor, 2) de su regulación y condiciones internas de operación, tales como su temperatura de aceite y las temperaturas de sus componentes, 3) de las condiciones externas de operación, definidas principalmente por la temperatura de aire de admisión, la presión atmosférica y la temperatura del combustible, 4) de su velocidad (ω_m) de rotación y 5) de la posición del acelerador o Bomba Inyectora (a). Si se asume que, durante un ensayo, los primeros tres factores se mantienen constantes, se puede representar al torque entregado por el motor como una función de los dos últimos, es decir, de su velocidad y posición del acelerador. Se tiene así una expresión que, por ser periódica, puede ser desdoblada en otras dos separando la dependencia del tiempo:

$$T_m(\omega_m, a, t) = T_m'(\omega_m, a).f_m(t) \qquad (73)$$

donde f_m es la función periódica. Para el caso de un motor de cuatro tiempos con "N_c" cilindros, su frecuencia es :

$$\omega_t = \omega_m . N_c / 2 \qquad (74)$$

Para representar el torque entregado por el motor, en algunos casos, es suficiente su valor medio $\mathbf{T}_m'$, mientras que en otros es necesario conocer la evolución de $\mathbf{T}_m$ en el tiempo. .En la Figura 25, se representa el torque medio de un motor para diferentes posiciones del acelerador.

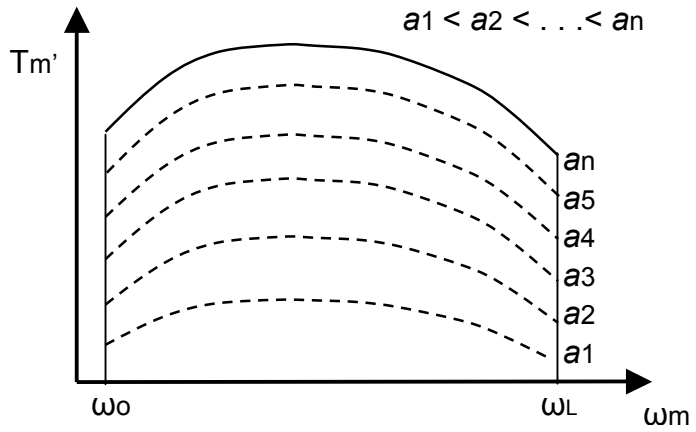


Figura 25: Curvas del torque medio de un motor

En el espacio de estados, la región de operación del motor está limitada por su velocidad mínima, la curva de torque máximo y su velocidad límite. Tal como se verá más adelante, es esencial el conocimiento de las regiones de operación de los componentes del sistema para la definición de las estrategias de control.

Además de la respuesta estacionaria del motor ante una cierta posición del acelerador, es necesario conocer la rapidez de respuesta ante un cambio en este valor que es definido como "a". En la Figura 26, se representan los parámetros que definen esta condición transitoria:

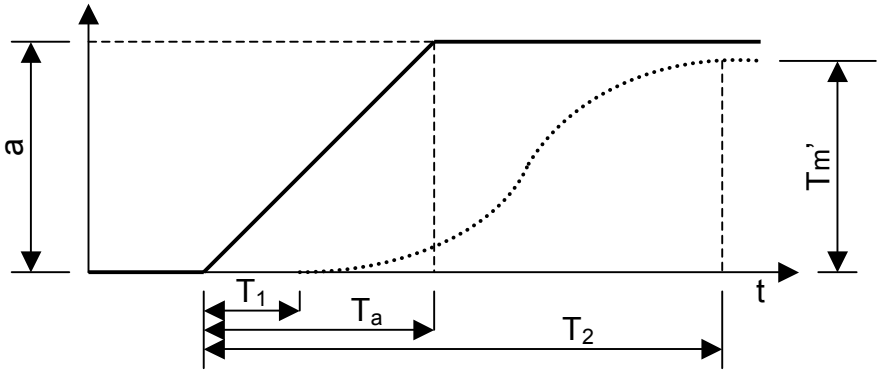


Figura 26: Tiempos en la respuesta de un motor

donde:

T_a : Tiempo del acelerador para completar un desplazamiento "a"
T_1 : Demora en iniciarse el cambio de la condición del motor
T_2 : Tiempo en alcanzar el motor una nueva condición estacionaria T_m'

4.4.2 Freno dinamométrico

Ya fue definido al freno dinamométrico como una máquina rotativa destinada a absorber el torque entregado por un motor. Entre sus diferentes variantes, los eléctricos y los hidráulicos son los más utilizados.

Los frenos eléctricos son también llamados de "corriente de Fucol" o de "corrientes parásitas". En estos equipos, a partir de la circulación de corriente por el bobinado de un estator se desarrolla un campo magnético. Este a su vez genera corrientes parásitas que convierten en calor (efecto Joule) la energía suministrada por el motor. La regulación del torque de frenado se obtiene variando la intensidad de la corriente continua aplicada sobre el bobinado.

Los frenos hidráulicos, por su parte, disponen de un estator y un rotor sumergidos en agua, cuyo movimiento relativo convierte en calor la energía que debe ser absorbida. Por su principio de funcionamiento, estos frenos requieren que la presión de agua de alimentación se mantenga constante y que el caudal sea suficiente para evacuar el calor. Aquí, la regulación del torque de frenado se obtiene variando el caudal de agua a la salida del freno.

En estas máquinas, la capacidad de absorber torque depende de su velocidad de rotación y de la regulación de su capacidad de frenado. La expresión toma la forma:

$$\mathbf{T}r\ (\omega r,\ v,\ \theta e) = \mathbf{T}r'\ (\omega r,\ v\ ) \,.\, g\ (\theta e) \qquad (75)$$

Aquí, θe representa un movimiento relativo que puede presentarse entre el estator del freno y su montaje. En la mayoría de los casos, este movimiento no existe o es despreciable, por lo que sólo es considerado en estudios de fenómenos vibratorios muy especiales. En la mayoría de los casos, se considera que:

$$\mathbf{T}r\ (\omega r,\ v\ ) = \mathbf{T}r'\ (\omega r,\ v\ ) \qquad (76)$$

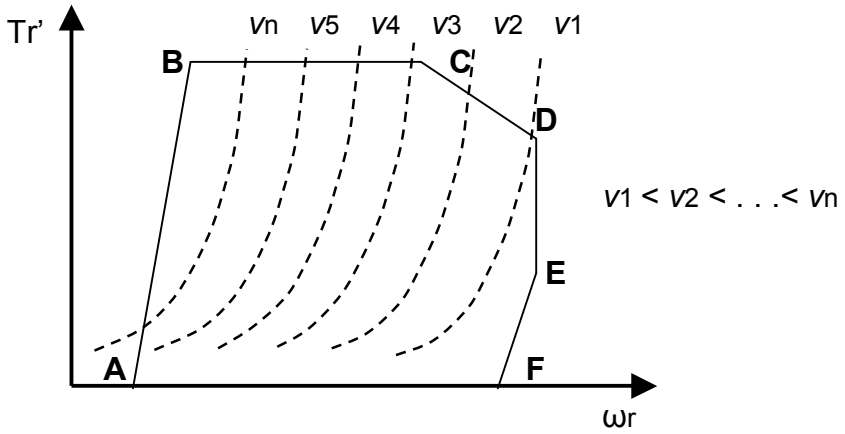


Figura 27: Curvas de frenado del dinamómetro

La región de operación del freno está también limitada en el espacio de estados. En este caso, la región de operación queda definida por una poligonal que es representada en la Figura 27, donde también se muestran las curvas de frenado que corresponden a las diferentes regulaciones.

Los segmentos de esta poligonal tienen la siguiente interpretación:

AB – Máxima Potencia a baja velocidad
BC – Torque Máximo admitido
CD – Potencia Máxima admitida
DE – Máxima Velocidad de Operación
EF – Límite inferior de potencia medible

Al igual que con el motor, es necesario conocer el retardo con que reacciona un freno ante un cambio en su excitación, y éste queda definido por los parámetros que se muestran en la Figura 28:

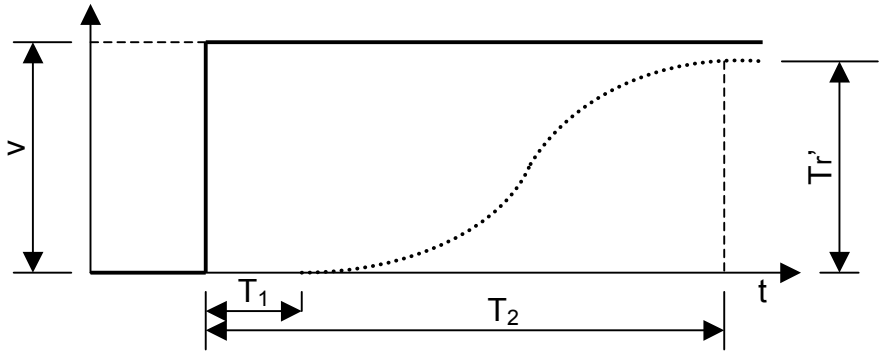


Figura 28: Tiempos en la respuesta del dinamómetro

donde:
T_1 : Demora en iniciarse el cambio de la condición del freno
T_2 : Tiempo en alcanzar el frenado una nueva condición estacionaria T_r'

4.4.3 *Acoplamiento Motor-freno*

Al ser acoplados entre sí, con un eje que toma la forma de una transmisión cardánica, el motor y el freno se convierten en un conjunto que tendrá características particulares.

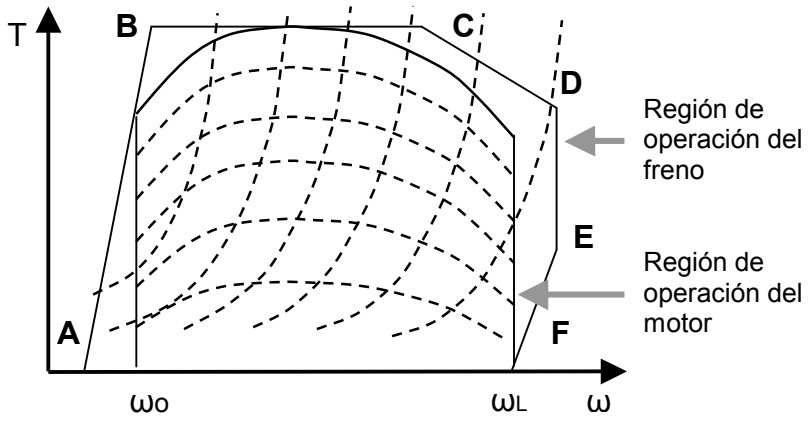


Figura 29: Región de operación del conjunto motor-freno

Confirmando lo ya anticipado, tal como se presenta en la Figura 29, la región de operación de este conjunto en el espacio de estados resultará de la intersección de las regiones en las que pueden operar los tres componentes: motor, freno y transmisión cardánica.

Por lo tanto, resulta obvio que para no penalizar su funcionamiento la región de operación del motor debe estar incluida en las otras dos, es decir que:

$$\mathbf{U}_m \subset (\mathbf{U}_f \cap \mathbf{U}_t) \qquad (77)$$

Para desempeñar correctamente su función, este eje debe cumplir con numerosas condiciones, que son propias del diseño mecánico, y cuyo detalle no es relevante para los fines de este análisis. Sin embargo, es necesario mencionar algunas de estas condiciones que están vinculadas a la operación del conjunto. En efecto, tal como expresa la ecuación 77, un diseño inadecuado del eje restringirá las regiones de operación que son propias del motor y freno por separado. Estas condiciones son:

- Capacidad de transmitir el torque máximo capaz de ser entregado por el motor.
- Capacidad de girar a la máxima velocidad esperada.
- Diseño apropiado para absorber la eventual falta de alineación entre el motor y freno. De lo contrario, quedaría también restringida la velocidad máxima de operación del conjunto.

El comportamiento de este conjunto puede ser representado en la mecánica newtoniana por ecuaciones diferenciales que incluyen las cargas del motor y freno, la inercia de cada elemento, el amortiguamiento y eventualmente la rigidez de la transmisión cardánica entre motor y freno. Se trata, por lo tanto, de un fenómeno continuo que es resuelto a partir de la integración de una familia de ecuaciones diferenciales de movimiento (rotación) que están asociadas a las distintas condiciones del motor y del freno dinamométrico.

En el croquis de la Figura 30, se representa la inercia del motor I_m y la del freno dinamométrico, distinguiéndose en este último la inercia del rotor I_r y del estator I_e. También se representa la elasticidad de la transmisión cardánica k, la elasticidad de la celda de carga $\boldsymbol{k}_c$ y la distancia de la celda de carga al eje de rotación L.

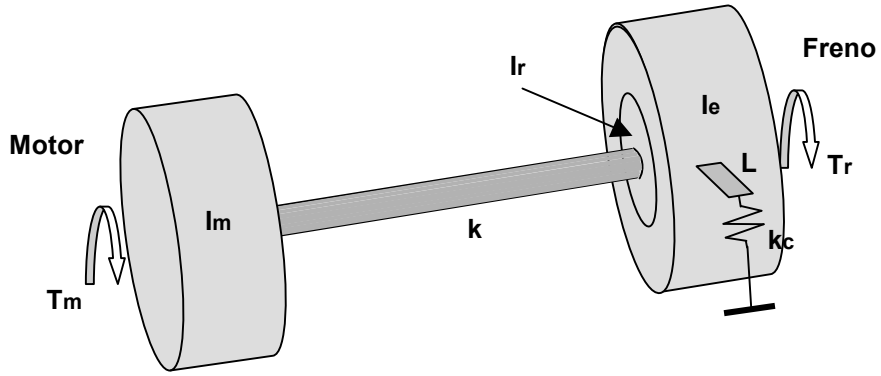


Figura 30: Diagrama de cuerpo libre del conjunto motor-freno

Como ya fue indicado, el par motor T_m es función de la velocidad del motor, la posición de la bomba inyectora "*a*" y el tiempo "*t*". Por su parte, el par de frenado *Tr* es función de la velocidad del rotor, posición del estator y tensión aplicada sobre su bobinado "*v*". La rotación del motor, rotación del rotor y oscilación angular del estator son representadas por θ_m, θ_r y θ_e respectivamente y con F_{s1} y F_{s2} se representan a las fuerzas de fricción que son independientes del movimiento.

El comportamiento del sistema queda expresado por el siguiente sistema de ecuaciones diferenciales:

$$\mathbf{I}_m \ddot{\theta}_m + \mathbf{C}_m \dot{\theta}_m + \mathbf{k} (\theta_m - \theta_r) = \mathbf{T}_m (\dot{\theta}_m , \mathbf{a}, t) + F_{s1} \quad (78)$$

$$\mathbf{I}_r \ddot{\theta}_r + \mathbf{C}_r \dot{\theta}_r + \mathbf{k} (\theta_r - \theta_m) = \mathbf{T}_r (\dot{\theta}_r , \mathbf{v}, \theta_e) + F_{s2} \quad (79)$$

$$\mathbf{I}_e \ddot{\theta}_e + \mathbf{C}_e \dot{\theta}_e + \mathbf{k}_c L \theta_e = \mathbf{T}_r (\dot{\theta}_r , \mathbf{v}, \theta_e) \quad (80)$$

a. Modelo elástico

La integración de las ecuaciones 78 a 80 permitirá conocer la respuesta del sistema y explicar diversos fenómenos vibratorios que pueden poner en riesgo la integridad del conjunto motor-freno o afectar la operación del sistema. Entre estos últimos, puede citase el caso en que una elevada amplitud de oscilación en el estator del freno θ_e perturbe significativamente la lectura de carga e impida la operación normal del sistema de control.

A través de consideraciones energéticas, puede conocerse el momento de inercia del motor $\mathbf{I}_m$. Para ello, se determina un momento de inercia rotatorio que sea equivalente al del las masas rotativas y alternativas presentes. Debe, además, reconocerse que se trata de un momento de inercia que presenta una variación periódica en el tiempo, cuya frecuencia es $\omega_t = \omega_m . N_c$, donde N_c representa la cantidad de cilindros del motor.

Los tratados clásicos de mecánica de vibraciones de Den Hartog (Den Hartog, 1964) y Timoshenko (Timoshenko y Young, 1956) tratan este problema con suficiente detalle y profundidad.

b. Modelo rígido

Si se asume que tanto la transmisión cardánica como la celda de carga tienen ambas rigideces muy elevadas (infinitas), resulta $\theta_r = \theta_m = \theta$ y $\theta_e = 0$. Esta última

condición elimina la ecuación 80. Sumando miembro a miembro las expresiones restantes se tiene:

$$(\mathbf{I}_m+\mathbf{I}_r)\ \ddot{\theta} + (\mathbf{C}_m+\mathbf{C}_r)\ \dot{\theta} = \mathbf{T}_m(\dot{\theta}, a, t) + \mathbf{T}_r(\dot{\theta}, v) + F_{s1} + F_{s2} \qquad (81)$$

que puede ser reescrita:

$$\mathbf{I}\ \ddot{\theta} + \mathbf{C}\ \dot{\theta} = \mathbf{T}_m(\dot{\theta}, a, t) + \mathbf{T}_r(\dot{\theta}, v) + F_s \qquad (82)$$

Reconociendo que $\dot{\theta}$ representa la velocidad de rotación del motor ω_m , la ecuación 82 puede reescribirse como:

$$\mathbf{I}\ \dot{\omega} + \mathbf{C}\ \omega = \mathbf{T}_m(\omega, a, t) + \mathbf{T}_r(\omega, v) + F_s \qquad (83)$$

Es decir que el modelo se reduce a un único grado de libertad al eliminarse la flexibilidad de la transmisión *k* y la de la celda de carga *kc ,* y queda apropiadamente representado por una única ecuación diferencial 83. Aunque esto impide la representación de fenómenos vibratorios, este modelo será muy útil para reproducir el comportamiento del conjunto motor-freno en su interacción con el sistema digital de control. Como ya se dijo, este modelo facilitará la optimización de los parámetros que determinan el comportamiento de este último.

En este caso, es suficiente asignar a la inercia del motor un valor medio que permanezca constante en el tiempo, equivalente al momento de inercia variable y obtenido a partir de consideraciones energéticas.

Por su parte, el amortiguamiento puede ser determinado experimentalmente midiendo el ángulo θ girado libremente por el conjunto motor-freno a partir de una velocidad inicial ω_o hasta su detención. Reconociendo que un coeficiente de amortiguamiento *h* puede reescribirse en función de *C* e ***I*** se tiene:

$$h = C / (2\,\boldsymbol{I}) \qquad (84)$$

y considerando la solución de la ecuación diferencial homogénea puede concluirse que (Bykhovsky, 1972):

$$h = \omega_o\, I\ (2\theta) \qquad (85)$$

4.4.4 *Taquímetro*

La medición de la velocidad del motor involucra dos etapas. En la primera, la velocidad de rotación ω es convertida en un tren de pulsos ρ. Para ello, se utiliza un sensor magnético montado próximo a una rueda dentada que gira solidaria al eje del motor o dinamómetro. En la segunda etapa, interviene el propio taquímetro, cuya finalidad es convertir este tren de pulsos en el valor de velocidad Ω.

El período del tren de pulsos es una función de la velocidad del eje y de la cantidad de dientes N_r de la rueda:

$$\tau\rho = 2.\pi\ /\ (\omega\ .\ N_r) \qquad (86)$$

y la velocidad final es

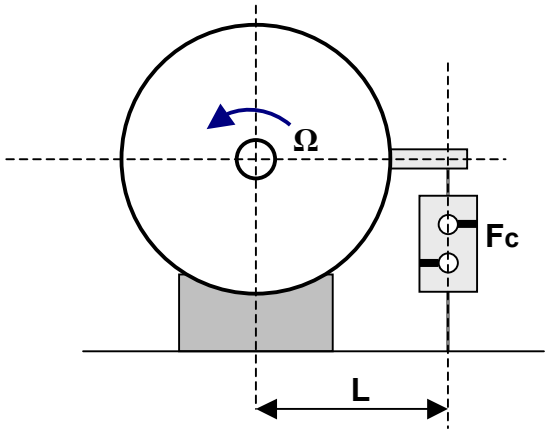
$$\Omega = 60\ /\ (\tau\rho\ .\ N_r) \qquad (87)$$

La necesidad de determinar la velocidad Ω con una frecuencia no menor a 10 Hz y con una resolución del orden de 1 rpm lleva a que los algoritmos utilizados sean más complejos de lo que podría pensarse. En efecto, al medir velocidades bajas conviene hacerlo determinando el intervalo de tiempo entre las señales generadas

por dos dientes sucesivos, mientras que cuando se trabaja en el otro extremo del intervalo el cálculo de apoya en la cantidad de pulsos recibidos en cierto intervalo de tiempo. Un análisis más detallado permite establecer cuando conviene usar uno u otro procedimiento a efectos de obtener la mayor precisión en la medición.

4.4.5 *Sensor de torque*

Tal como presenta la Figura 31, el torque de frenado T_r se transmite íntegramente a la celda de carga, ya que el freno está suspendido en un montaje basculante y giratorio sobre su propio eje por medio de rodamientos. El torque T_c es medido a partir de la determinación de la carga reactiva F_c que tiene su línea de acción a una distancia L del eje de rotación del conjunto motor-freno. Para ello, se emplea una celda de carga axial de tracción.



$$T_c\ [Kg.m] = L \cdot F_c \qquad (88)$$

$$P\ [CV] = 2\,\pi\,L\,F_c\,\Omega / 4500 = L\,F_c\,\Omega / 716$$

donde :

Ω [rpm] : Velocidad de rotación
L [m] : Distancia al eje de rotación
F_c [Kg] : Fuerza en celda de carga

Figura 31 : Medición de torque y potencia en un dinamómetro

Sin embargo, obsérvese que se está haciendo referencia al torque absorbido por el freno, cuando el valor que más interesa es el del torque entregado por el motor, que no es posible de medir directamente. Si bien ambos valores se equilibran cuando el conjunto motor-freno se encuentra en funcionamiento estacionario, son diferentes en régimen transitorio, que es precisamente cuando debe intervenir el sistema de control. Así, el torque absorbido por el freno puede ser medido a través de una celda de carga y el torque entregado por el motor debe ser determinado en forma indirecta.

Considerando aquí el modelo rígido de un único grado de libertad, la ecuación 83 puede ser reescrita y se tiene:

$$T_m = I\,d\Omega/dt + C\,\Omega - T_c - F_s \qquad (89)$$

lo que indica que el torque entregado por el motor es determinado a partir de las lecturas de los valores del torque en la celda de carga T_c y la velocidad medida por el taquímetro Ω. La aceleración $d\Omega/dt$ debe ser obtenida derivando numéricamente la curva de velocidad, para lo cual se utilizará el procedimiento definido en el punto 2.4. Por último, los parámetros I , C y F_s son característicos del sistema dinámico y deben ser evaluados experimentalmente.

La necesidad e importancia de distinguir entre los torques del motor y del freno fue puntualizada por Hori (Hori, 1995), pero pasó completamente inadvertida para otros muchos investigadores. Evaluar la incidencia que realmente tiene el uso de uno u otro valor de torque en la unidad de control es uno de los objetivos de este simulador.

Además, como en el caso de la lectura de velocidad, es esencial conocer la frecuencia de su muestreo, por la incidencia que tiene en la calidad de la respuesta del sistema de control.

4.4.6 Seguros

Se trata de una unidad que, implementada de diferentes maneras según cada caso, debe delatar condiciones de operación del Motor o Freno que puedan poner en peligro la integridad de sus componentes. Estas condiciones pueden referirse a valores alcanzados por variables del vector de estado u otras variables tales como:

- Motor: presión de aceite, temperatura de agua de salida, salto térmico en el agua entre entrada y salida del motor, etc.
- Dinamómetro: presión, temperatura y caudal de agua de refrigeración, temperaturas de cojinetes, etc.
- Conjunto motor-freno: velocidad superior a la admisible (sobrevelocidad).

La detección de alguna de estas anomalías debe generar un mensaje a la unidad de control con el fin de provocar la detención inmediata del motor. Este mensaje será reconocido por la unidad de control como un evento que merece la más alta prioridad.

4.4.7 Control primario

La unidad de control primario es el elemento central en el ensayo del motor, ya que ésta debe permitir operarlo con la mayor flexibilidad en toda su región del espacio de estados. Esta unidad incluye las denominadas “lógicas de control” y el selector que tiene la función de elegir la lógica más apropiada (Figura 4).

Todos los puntos pertenecientes a la región del espacio de estados en que el sistema es operable corresponden a condiciones de equilibrio del motor y freno. Esto se aprecia con claridad superponiendo las curvas del motor y del freno, tal como se presenta en la Figura 32.

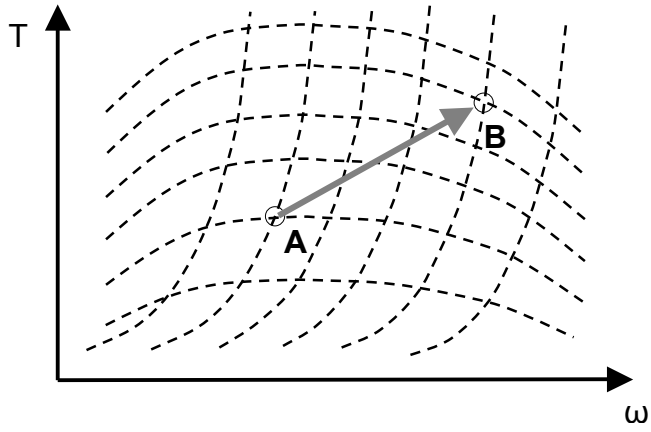


Figura 32: Condiciones de equilibrio del conjunto motor-freno

Luego, la misión fundamental de esta unidad es llevar las variables de proceso a alcanzar rápidamente los valores requeridos (Set Points) evitando oscilaciones u otras condiciones de marcha no deseadas. Esto significa conducir rápidamente el motor-freno de una condición A (Ω_A, T_A) a otra condición B (Ω_B, T_B) y mantener el sistema en régimen estacionario en éste último estado.

El conjunto motor-freno debe ser reconocido como un sistema *MIMO* (Muti Input – Multi Output) ya que, como fue anticipado, sus condiciones de operación quedan definidas por dos variables de proceso: velocidad y torque. Sin embargo, estos sistemas han sido tradicionalmente diseñados como *SISO* (Single Input – Single Output), donde ambas variables son controladas en forma independiente y simultánea. En algunos casos es necesario una coordinación externa y para ello se suelen prever procesos de sincronización o secuenciado de las acciones sobre el freno y acelerador, que es incorporado una vez que se han ajustado los parámetros de los ciclos básicos por separado. Tal como presenta las Figuras 23 y 32, ambos sistemas están normalmente acoplados a través del eje de transmisión cardánica, que es el que vincula a los dos elementos controlados. En particular, el esquema de control representado en la Figura 33 será en lo sucesivo identificado como "*Lógica 1*" de control.

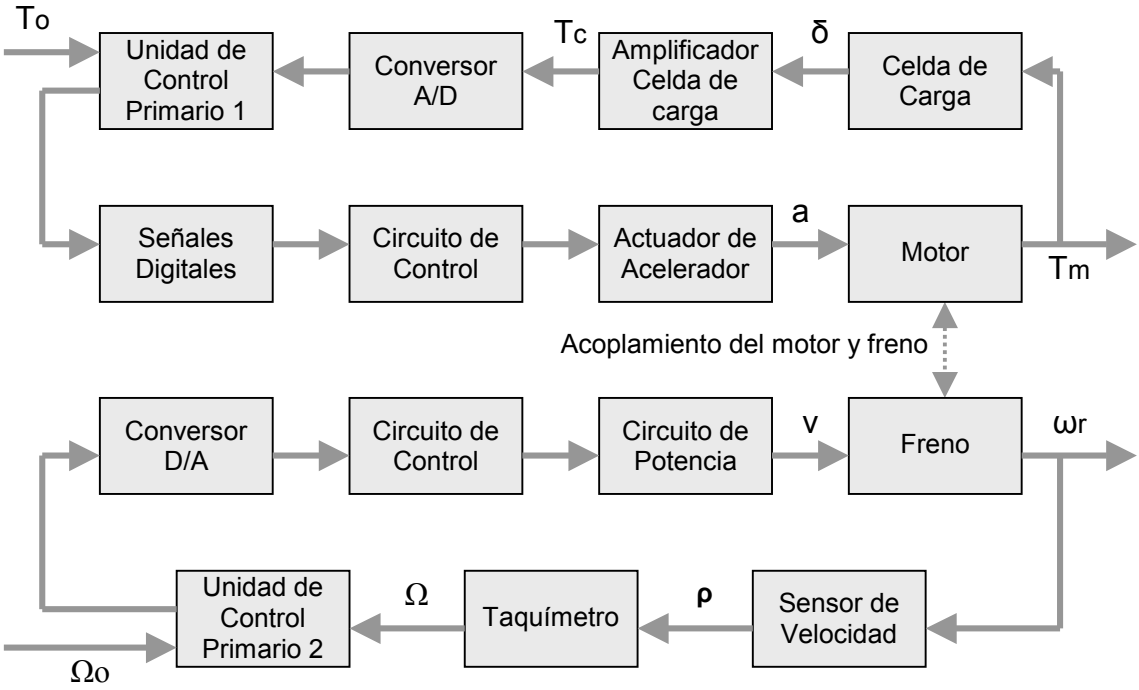


Figura 33: Control independiente de las variables de proceso

En general, la experiencia ha demostrado que ésta es una solución apropiada para el control de motores. A pesar de ello, la literatura advierte que el control independiente de variables de proceso en sistemas *MIMO* sólo es aplicable a casos especiales y carece de generalidad, ya que aun en estos casos, pueden presentarse serias limitaciones en sus posibilidades (Ogata, 2003).

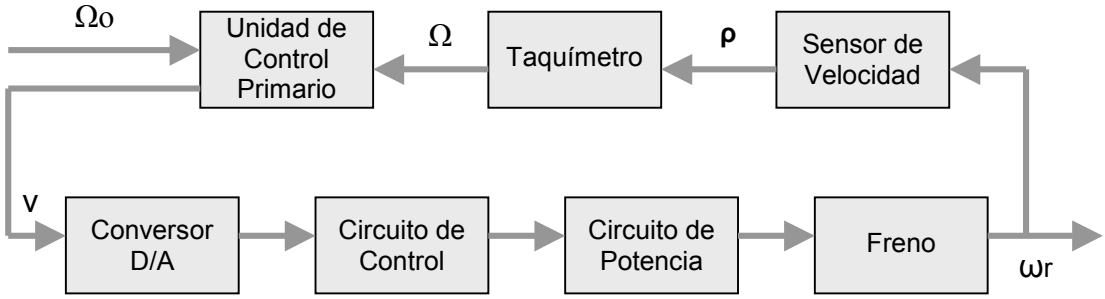


Figura 34 : Control de la velocidad mediante el freno

Los ensayos prevén algunas condiciones especiales en que se impone un valor prefijado a una de las variables de proceso y sólo se controla la restante, con lo que el sistema se reduce a la condición de *SISO*.

Un caso de este tipo se obtiene cuando el motor es operado con el acelerador en una posición prefijada y se utiliza el freno para regular su velocidad, tal como se presenta en la Figura 34. Obsérvese que éste es un sistema *SISO* que representa un caso particular de la *"Lógica 1"* (MIMO) antes definida.

En esta condición, el motor recorre una de las curvas que corresponde a una posición del acelerador constante, tal como presenta la Figura 35. Aquí, la curva AB es la de máximo torque ($a = a_{max}$) y la curva CD corresponde a una posición de acelerador intermedia ($0 < a < a_{max}$).

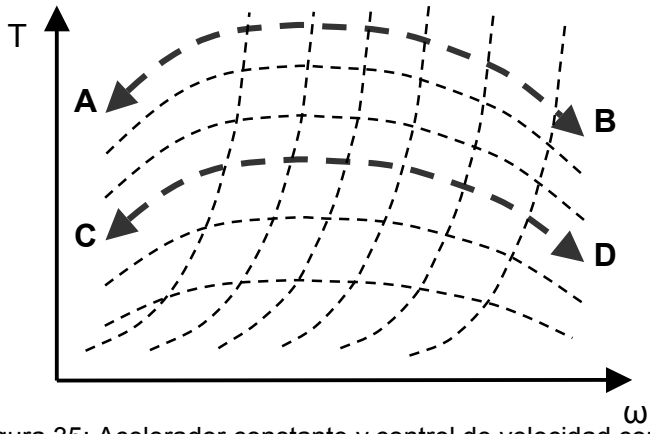


Figura 35: Acelerador constante y control de velocidad con el freno

En algunos otros casos, se utiliza el recurso de intercambiar las realimentaciones de ambas variables de proceso con el fin de obtener transiciones suaves y rápidas en condiciones particulares de operación, tal como se presenta en la Figura 36.

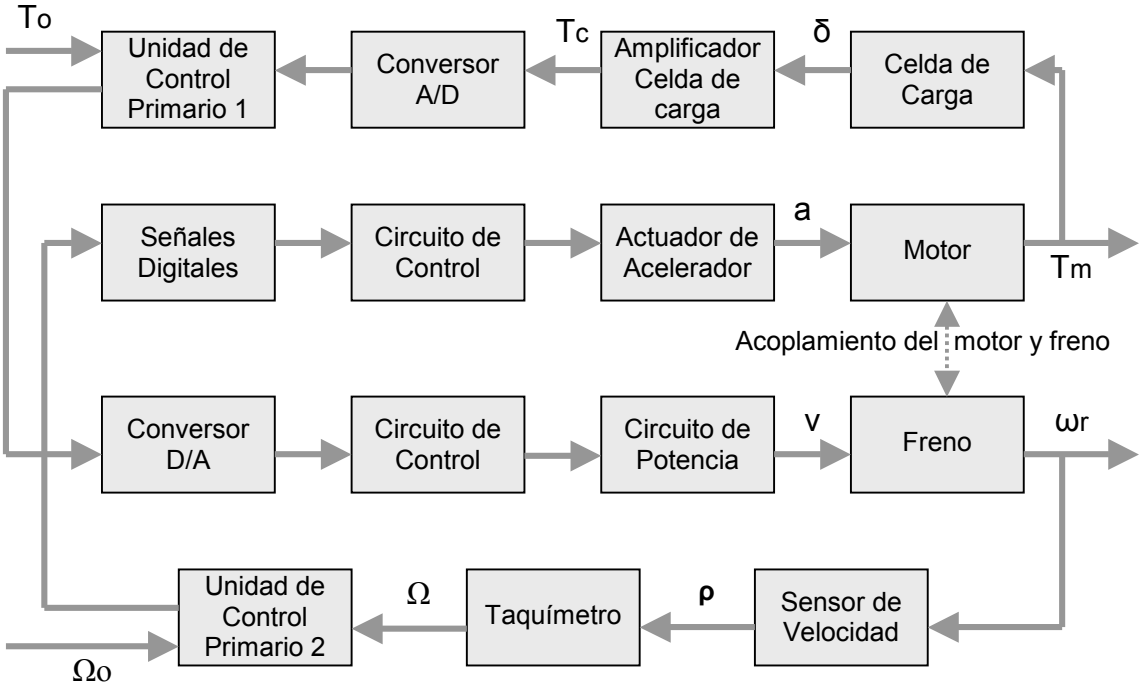


Figura 36: Control independiente con realimentaciones cruzadas

En este caso, la acción sobre el motor es comandada según el error cometido en el valor de la velocidad y el frenado es regulado a partir del error en el torque. Esta condición de control es tipificado como "*Lógica 2"*.

A su vez, operando el motor sin carga exterior, donde el torque generado es sólo el necesario para equilibrar las fuerzas de rozamiento que corresponden a cierta velocidad, se obtiene otra condición SISO que es un caso particular de la anterior. En este caso, presentado en la Figura 37, el motor descargado es operado por el acelerador a velocidades predeterminadas.

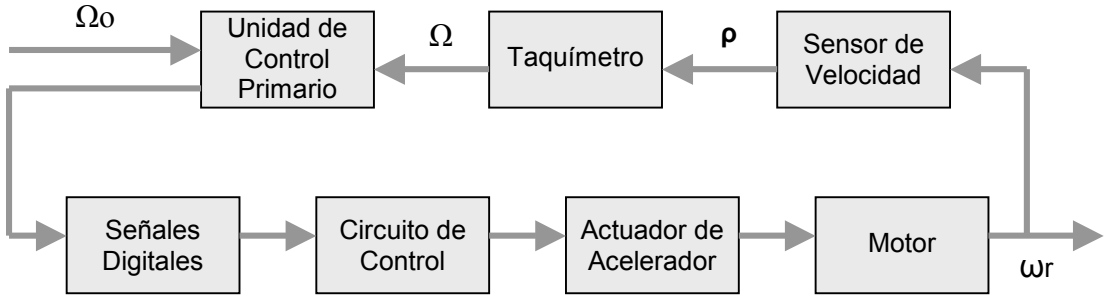


Figura 37: Control de la velocidad sin carga en el freno

En estos casos en que se involucran dos lazos de realimentación independientes y simultáneos, es necesaria la selección tanto de la lógica como de la opción de realimentación (PID, PI-D o PID-D) más conveniente. Tal como ya fue anticipado, el selector de lógicas de control (Figura 4) es el encargado de conmutar entre las diferentes opciones disponibles y esto es realizado a partir de un complejo proceso que considera el vector de estado, las posibles superficies de deslizamiento y un conjunto de reglas que definen los criterios de control disponibles.

Esto puede ser ejemplificado considerando el caso en que debe transitarse de un punto A a un punto B sobre el espacio de estados. Para ello y dependiendo de la forma en que se sincronicen las acciones básicas de control, se presentan tres alternativas fundamentales (Figura 38):

a) Operar sólo el freno (AC) y luego sólo el acelerador (CB).
b) Operar sólo el acelerador (AD) y luego sólo el freno (DB).
c) Operar coordinadamente sobre freno y acelerador (AB), lo que en realidad implica un número infinito de trayectorias posibles.

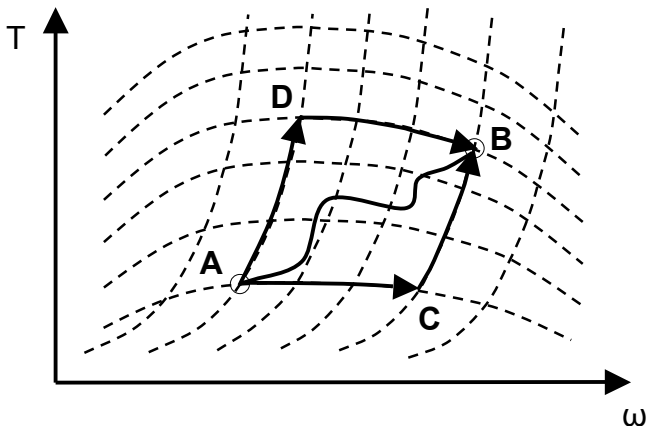


Figura 38: Efecto de diferentes secuencias de conmutación

4.4.8 Control secundario

Una de las misiones de la unidad secundaria de control es conducir el cumplimiento de un ensayo en forma completamente automática y para ello esta unidad es implementada como un intérprete de *programas*. Cabe aquí definir como "*programa de ensayo*" a una secuencia ordenada de condiciones de marcha del motor, que es necesario cumplir para completar un ensayo. A cada una de estas condiciones de operación del motor se las denomina "*paso*" del programa, quedando definido, como mínimo, por los siguientes parámetros:

- Identificador o número de paso
- Tiempo de permanencia en esta condición
- Velocidad del motor
- Torque esperado
- Próximo paso a ser ejecutado

A estos parámetros básicos, se agregan otros que están destinados a garantizar la seguridad de la prueba, a permitir el cumplimiento de ciclos repetidos, los comandos de arranque y parada del motor y los comandos que accionan equipos auxiliares.

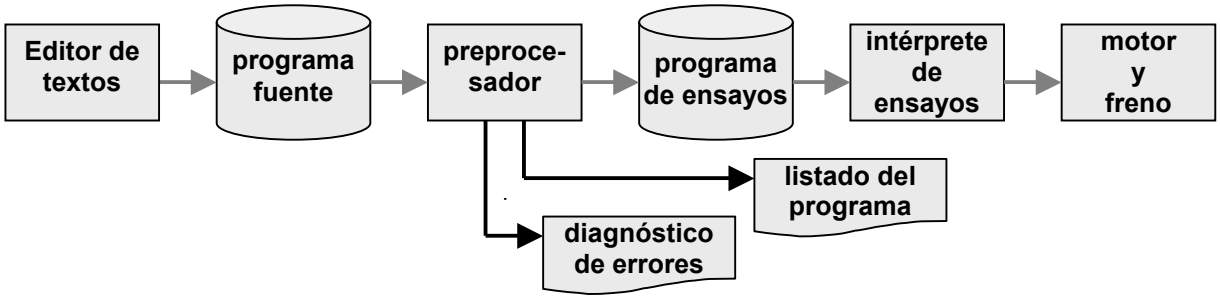


Figura 39: Definición de programa de ensayos

Para la definición de los programas de ensayos, se establece un lenguaje que responde a una gramática regular (Tipo 3 de Chomsky), y se desarrolla un preprocesador que cumple la función de un compilador. Su misión es verificar la corrección sintáctica y semántica del programa de ensayos y prepararlo para su ejecución, como es representado en la Figura 39.

Para facilitar la interpretación de la interacción entre las unidades de control primaria y secundaria, resulta conveniente un esquema, tal como el presentado en la Figura 40:

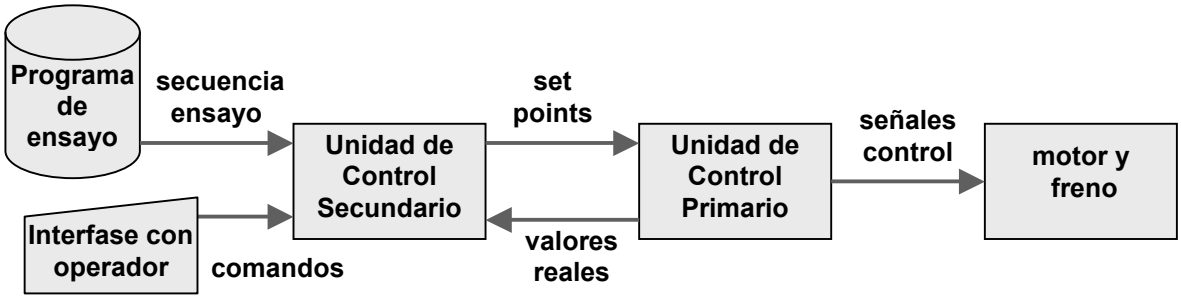


Figura 40: Interacción entre las unidades de control

La unidad secundaria de control debe, además, reconocer los comandos destinados a la operación manual del motor, que permiten su puesta en marcha, detención y operación remota del acelerador y freno. Las primeras órdenes son seleccionadas con pulsadores y las dos últimas con potenciómetros de precisión.

4.4.9 Interfase con el operador

Es de destacar que esta interfase reviste gran importancia en la operación de todo sistema real, pero no reviste mayor interés para el proceso de simulación que es objeto de este trabajo. Tampoco lo es desde el punto de vista de su respuesta en tiempo real y, por lo tanto, las características de esta interfase responderán más bien a los objetivos particulares perseguidos con las simulaciones.

Las dos funciones básicas de toda interfase con el usuario, en la actualidad tipificadas como *GUI* (*Graphic User Interface*) son las siguientes:

a. *Representación en pantalla de la condición de operación del sistema*: se muestran los valores de las variables, gráficos históricos y gráficos en tiempo real.

b. *Aceptación de comandos del operador*: operación manual del motor, alteración del orden normal de la secuencia de ensayos, cambios en las condiciones de seguridad y selección de las opciones de visualización.

Aquí, es oportuno señalar que las órdenes del operador son ingresadas por teclado, teclados especiales de funciones, mouse y potenciómetros. Estos últimos para la definición de set-points que son ingresados como señales analógicas.

4.5 Conversión de señales

Como ya fue anticipado, la necesidad de comunicación entre dominios de diferente naturaleza hace indispensable la conversión de las señales. La implementación de un modelo numérico para representar el fenómeno continuo de operación del conjunto motor-freno no hace más que incorporar nuevas exigencias y particularidades a estas comunicaciones entre los diferentes dominios.

En este punto es conveniente destacar que, en los sistemas reales, la conversión de señales es una fuente de dificultades y en la mayoría de los casos determina tanto su calidad como el alcance de sus prestaciones. Es por ello que en los modelos estas conversiones deben ser reproducidas con el mayor detalle y fidelidad posible a efectos de que éstos sean capaces de reproducir los verdaderos comportamientos de los sistemas que representan.

Para comenzar, debe reconocerse que el conjunto de las señales que representan fenómenos dinámicos pueden ser agrupadas según los valores que adoptan y la forma en que estos valores están definidas en el tiempo (Mascarós, 2002). Se distinguen así los siguientes cuatro grupos: 1) señales continuas en tiempo continuo (s_{cc}), 2) señales continuas en tiempo discreto (s_{cd}), 3) señales discretas en tiempo continuo (s_{dc}) y 4) señales discretas en tiempo discreto (s_{dd}). En los subconjuntos en tiempo discreto, deben a su vez distinguirse las señales sincrónicas y las asincrónicas. En las señales sincrónicas y en las de tiempo continuo, es también importante reconocer las periódicas y las no periódicas.

Otro concepto que ya fue definido es el de *evento*, reservándose en algunos casos esta denominación a ciertas condiciones particulares que se presentan en el sistema continuo en un cierto instante y que están destinadas a llamar la atención del sistema de control. En efecto, los eventos deben ser oportunamente reconocidos para posibilitar la activación de las acciones correctivas de control requeridas en cada caso. En algunos de estos casos, los eventos están asociados al comportamiento de señales individuales, tanto discretas como continuas, y en otros son consecuencia de cierta relación entre dos o más señales. También, se denominan eventos a decisiones externas al sistema, que a través de la unidad de control condicionan su operación. Cualquiera sea el caso, el espectro temporal de un evento es nulo ya que no tiene duración.

Las variables $\boldsymbol{y}$ que representan el fenómeno físico aquí estudiado corresponden a señales continuas definidas en tiempo continuo ($\boldsymbol{s_{cc}}$), ya que los valores que adoptan pertenecen al campo de los números reales y están definidas en todos los instantes de tiempo. Sin embargo, como ya fue anticipado, el fenómeno físico es representado por ecuaciones diferenciales que serán resueltas por integración numérica con cierto intervalo de tiempo Δt, lo que las convierte en señales continuas $\boldsymbol{u}$ en tiempo discreto y sincrónicas. El proceso de integración numérica $\boldsymbol{I}_n$ establece así una relación entre ambos conjuntos y en el modelo pasan a quedar representadas por estas últimas:

$$\boldsymbol{I}_n : \boldsymbol{s_{cc}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad \boldsymbol{u} = \boldsymbol{I}_n (\boldsymbol{y}, \Delta t) \qquad (90)$$

En lo sucesivo, al hacerse referencia a las mediciones sobre el sistema real debe reflejarse la siguiente dualidad: la representación de las señales de los sensores debe respetar las características propias de cada uno, sólo que en lugar de operar sobre señales del conjunto $\boldsymbol{s_{cc}}$ lo debe hacer sobre las del conjunto $\boldsymbol{s_{cd}}$.

4.5.1 Señales de Torque

Ya fue anticipado que el único torque que puede ser leído es el T_c , que es evaluado a través de las deformaciones δ de una celda de carga. En el sistema real, esta señal del conjunto $\boldsymbol{s_{cc}}$ es muestreada regularmente con un período que es impuesto par la rapidez del conversor análogo / digital disponible τ_a y es convertida en una señal sincrónica del conjunto $\boldsymbol{s_{cd}}$, luego:

$$\mathbf{f}_c' : \boldsymbol{s_{cc}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad T_c = \mathbf{f}_c' (\delta , \tau_a) \qquad (91)$$

Sin embargo, en el modelo sólo se cambia la representación en el tiempo de la señal dentro del mismo conjunto $\boldsymbol{s_{cd}}$ ya que:

$$\boldsymbol{I}_n : \boldsymbol{s_{cc}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad \delta = \boldsymbol{I}_n (\mathbf{y}, \Delta t) \qquad (92)$$

$$\mathbf{f}_c : \boldsymbol{s_{cd}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad T_c = \mathbf{f}_c (\delta , \tau_a) \qquad (93)$$

4.5.2 Señales de Velocidad

La medición de la velocidad del motor Ω merece un comentario especial. En efecto, se trata de la variable ω_r que ya fue convertida al conjunto $\boldsymbol{s_{cd}}$ por la integración numérica. Luego, su medición se realiza a través de un dispositivo apropiado que la convierte en un tren periódico de pulsos ρ ($\boldsymbol{s_{dd}}$) y para su

interpretación física es vuelta a convertir en una señal perteneciente al conjunto s_{cd} identificada como Ω:

$$\boldsymbol{I}_n : \boldsymbol{s_{cc}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad \omega r = \boldsymbol{I}_n (\mathbf{y}, \Delta t) \qquad (94)$$

$$\mathbf{f_\rho} : \boldsymbol{s_{cd}} \rightarrow \boldsymbol{s_{dd}} \quad ; \quad \rho = \mathbf{f_\rho} (\omega r) \qquad (95)$$

$$\mathbf{f_\Omega} : \boldsymbol{s_{dd}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad \Omega = \mathbf{f_\Omega} (\rho) \qquad (96)$$

donde $\mathbf{f_\rho}$ representa al dispositivo de medición y $\mathbf{f_\Omega}$ representa al taquímetro del sistema, los que serán tratados en detalle por separado.

4.5.3 Eventos

La señal de seguros λ , la señal de habilitación del freno η y la de encendido del motor μ, pertenecen al conjunto $\boldsymbol{s_{dd}}$ de señales asincrónicas y están relacionadas a eventos que producirán cambios en las condiciones de operación del sistema. Se originan y son aplicadas sin sufrir transformaciones en su representación.

Otros eventos que son reconocidos por la unidad de control primario están relacionadas con ciertas condiciones de las variables **y** en el caso del sistema real o de las variables **u** del modelo y actúan sobre la misma unidad de control. Su objeto es provocar las transiciones en el autómata finito a efectos de hacer la regulación apropiada de las acciones de control.

$$\mathbf{f_\chi} : \boldsymbol{s_{cd}} \rightarrow \boldsymbol{s_{dd}} \quad ; \quad \chi = \mathbf{f_\chi} (\Omega, T_c) \qquad (97)$$

4.5.4 Acciones de Control

Las acciones de control son realizadas a través de dos señales que tienen por destino al freno dinamométrico y el acelerador del motor. El freno se regula a través de la intensidad del campo magnético en su estator, que será proporcional a la amplitud de la señal eléctrica continua (v) suministrada a su circuito de comando. Por su parte, para controlar el motor se genera un tren de pulsos que es convertido en una posición del acelerador (a). Estas señales del conjunto $\boldsymbol{s_{cd}}$ se originan en la unidad de control, y en el sistema real, son convertidas al conjunto $\boldsymbol{s_{cc}}$ a través de un conversor digital / analógico y al conjunto $\boldsymbol{s_{cd}}$ mediante un actuador electromecánico. Se tiene entonces:

$$\mathbf{f_f'} : \boldsymbol{s_{cd}} \rightarrow \boldsymbol{s_{cc}} \quad ; \quad v = \mathbf{f_f'} (\Omega, T_c) \qquad (98)$$

$$\mathbf{f_m'} : \boldsymbol{s_{cd}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad a = \mathbf{f_m'} (\Omega, T_c) \qquad (99)$$

y las acciones de control en el modelo se convierten según se presenta:

$$\mathbf{f_f} : \boldsymbol{s_{cd}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad v = \mathbf{f_f} (\Omega, T_c) \qquad (100)$$

$$\mathbf{f_m} : \boldsymbol{s_{cd}} \rightarrow \boldsymbol{s_{cd}} \quad ; \quad a = \mathbf{f_m} (\Omega, T_c) \qquad (101)$$

4.6 Especificación de requerimientos

El contenido de los puntos anteriores tuvo la finalidad de presentar y analizar las características y condiciones de los ensayos de motores. Este proceso de elicitación de las exigencias del sistema culmina con la preparación de los distintos documentos que forman parte de la especificación de requerimientos de sistemas (SRS), según la metodología adoptada y resumida en el esquema de la Figuras 18 y 19.

4.6.1 Límites del sistema

El sistema se circunscribe a las unidades de control primaria y secundaria, que tienen a su cargo la operación del conjunto motor-freno, según las pautas que están establecidas en el programa de ensayos y las acciones del operador. El contexto del sistema está por lo tanto conformado por todos los demás elementos ya descriptos en el punto 4.4 y representados en el esquema de la Figura 24.

4.6.2 Vocabulario del Léxico Extendido del Lenguaje

Una vez presentado el problema en forma general, debe continuarse con el reconocimiento y definición precisa de su terminología. Se recurre para ello al *LEL* (*Léxico Extendido del Lenguaje*) según las previsiones de la metodología adoptada y para implementar su vocabulario se emplea una tabla en la que se describen las nociones e impactos de cada palabra o frase. Aquí, debe recordarse que la noción representa el significado de cada término y el impacto determina los efectos de su uso u su ocurrencia en la aplicación**.**

Según las recomendaciones de Leandro Antonelli (Antonelli,1999 y 2001), este vocabulario es progresivamente ajustado y perfeccionado a medida que se progresa en el conocimiento del problema real y sus resultados son presentados en la Tabla 2.

N°	Símbolo	Noción	Impacto
1	Motor	Máquina rotativa-alternativa de combustión interna.	Conjunto Motor-Freno Variable de Proceso
2	Freno dinamométrico	Máquina rotativa destinada a absorber la potencia entregada por un motor. Es una evolución del “freno de Prony”.	Conjunto Motor-Freno Variable de Proceso
3	Acoplamiento o Conjunto Motor-Freno	Conjunto formado por un Motor y un Freno vinculados entre sí por una transmisión cardánica.	Variables de proceso
4	Ensayo	Evaluación experimental de un equipo en general y de un motor en particular.	Conjunto motor-freno
5	Taquímetro	Instrumento destinado a determinar la velocidad de un motor.	Vector de estado
6	Sensor de Torque	Instrumento destinado a determinar el torque de un motor o freno.	Vector de estado

Tabla 2: Vocabulario del LEL

Nº	Símbolo	Noción	Impacto
7	Seguros	Unidad destinada a delatar si las condiciones de operación del Motor o Freno pueden poner en peligro su integridad.	Control secundario
8	Medición	Operación experimental destinada a asignar a cierta magnitud de un dominio físico una cantidad de unidades de una escala apropiada (Maiztegui, 2000).	Determinación
9	Determinación	Asignación de un valor a una variable física, ya sea directamente a través de una medición o en forma indirecta a partir de otras mediciones.	Taquímetro Freno dinamométrico
10	Evento	Identificador atemporal de un cambio en la condición externa de un sistema..	Unidad de control
11	Estado	Condición distintiva en que se encuentra un sistema.	Variable de estado
12	Variable de estado	El menor conjunto de variables que definen completamente el estado de un sistema.	Vector de estado
13	Vector de estado	Arreglo unidimensional que agrupa a todas las variables de estado. Determina unívocamente el estado de un sistema.	Espacio de estados Interfase con operador
14	Espacio de estados	Espacio n-dimensional cuyos ejes de coordenadas están formados por las variables de estado. Todo estado de un sistema queda representado como un punto en su espacio de estados	Control primario
15	Variable de proceso (VP)	Una de las variables de estado que se desea controlar, como es el caso de la velocidad del motor o su torque.	Taquímetro Freno dinamométrico
16	Set point (SP)	Es el valor deseado de la variable de proceso, es decir, el valor que debe ser alcanzado y mantenido estable por la unidad de control.	Control primario
17	Error (e)	Diferencia entre el set point (SP) y la variable de proceso (PV).	Control primario
18	Control	Acción destinada a procurar que en todo momento las variables de proceso alcancen los valores prefijados por los set points, es decir, que el error sea nulo.	Unidad de control
19	Unidad de Control	Dispositivo destinado a cumplir acciones de control.	Control primario Control secundario
20	Control primario	Función de la Unidad de Control, que tiene la finalidad de operar directamente sobre el objeto controlado.	Variable de control Señales de Control Lógica activa

Tabla 2: Vocabulario del LEL

N°	Símbolo	Noción	Impacto
21	Control secundario	Función de la Unidad de Control, que tiene la finalidad de interpretar y ejecutar los Programas de Ensayo.	Control primario Interfase con operador Paso de Ensayo Activo
22	Variable de control (VC)	Variable que está asociada a la acción utilizada para controlar un sistema. En el caso de un motor se trata de la posición del acelerador.	Vector de control
23	Vector de control	Vector en el que se reúnen las variables de control (VC).	Motor Freno dinamométrico
24	Señal de control	Señales digitales de encendido del motor y habilitación del freno.	Motor Freno dinamométrico
25	Modo automático	Modo de operación de un motor en el que se da cumplimiento a un Programa de Ensayo.	Control secundario
26	Controlabilidad	Posibilidad de cambiar el estado de un sistema entre dos puntos cualesquiera del espacio de estados en un intervalo de tiempo finito. Obsérvese que la condición de controlabilidad determina la existencia de una solución en el problema del diseño de un sistema de control.	Control primario
27	Obsevabilidad	Posibilidad de acceder a todas las variables de estado a través de mediciones. Si un sistema no es completamente observable, algunas de sus variables deben ser determinadas en forma indirecta.	Variables de estado
28	Realimentación	Aplicación de una cierta lógica activa. Este proceso suele denominarse de lazo cerrado.	Control primario
29	Lógica activa	Proceso a través del cual se calculan valores convenientes para la variable de control (VC) a partir de los valores alcanzados por la variable de proceso (VP) y su error.	Realimentación
30	Paso de Ensayo	Condiciones deseadas para la operación de un motor durante un cierto intervalo de tiempo.	Programa de ensayo
31	Paso de Ensayo Activo	Paso de Ensayo empleado en cierto instante para el control del motor.	Control primario
32	Programa de Ensayo	Secuencia teórica de Pasos de Ensayo acordes a los objetivos de un ensayo.	Control secundario
33	Secuencia de ensayo	Secuencia real de Pasos de Ensayo a ser cumplidos.	Interfase con operador
34	Parada de emergencia	Orden de detención inmediata del motor.	Interfase con operador

Tabla 2: Vocabulario del LEL

Nº	Símbolo	Noción	Impacto
35	Final del ensayo	Orden de finalizar la secuencia de ensayo abandonando el modo automático y detener el motor	Interfase con operador
36	Función de transferencia (G)	Cociente entre las expresiones matemáticas de las variables de proceso (salida) y las variables de control del sistema (entrada) en función del tiempo. Suele también denominarse transmitancia.	Conjunto motor-freno
37	Interfase con operador	Pantalla de representación, teclado y otros medios de ingreso de datos.	Control secundario

Tabla 2: Vocabulario del LEL

4.6.3 Identificación de eventos

Del vocabulario del *LEL* se seleccionan los símbolos que se refieren a elementos externos al sistema y que impactan sobre el propio sistema o que pertenecen al sistema y su comportamiento tiene alguna connotación temporal. Luego, se trabaja sobre esta selección hasta reconocer los *eventos* externos y temporales que estimulan el sistema, teniendo aquí en cuenta las ya mencionadas condiciones que debe cumplir un evento y que fueron enumeradas por Page-Jones (Page-Jones,1992).

Entradas al LEL	Evento	Contexto / Sistema	Sistema	Contexto
		Origen / intermediarios del evento	Receptor de mensaje y elementos intermedios	Destinos de respuesta
5 / 13	1	Taquímetro	Vector de estado	Interfase con operador
6 / 13	2	Sensor de Torque	Vector de estado	Interfase con operador
34/37/21/ 20/24	3	Parada de emergencia Interfase con operador	Control secundario Control primario Señales de control	Motor Freno dinamométrico Interfase con operador
35/37/21/ 20/24	4	Final del ensayo Interfase con operador	Control secundario Control primario Señales de control	Motor Freno dinamométrico Interfase con operador
33/37/21/ 31	5	Alterar secuencia de ensayo Interfase con operador	Control secundario Paso de ensayo activo	Interfase con operador
7/21/20/ 24	6	Seguros	Control secundario Control primario Señales de control	Motor Freno dinamométrico Interfase con operador
32/21	7	Programa de Ensayo	Control secundario Paso de ensayo activo	Interfase con operador
28/20/22/ 23	8	Realimentación	Control primario Variable de control Vector de control	Motor Freno dinamométrico
13/14/20/ 29	9	Vector de estado	Espacio de estados Control primario Lógica activa Realimentación	--------

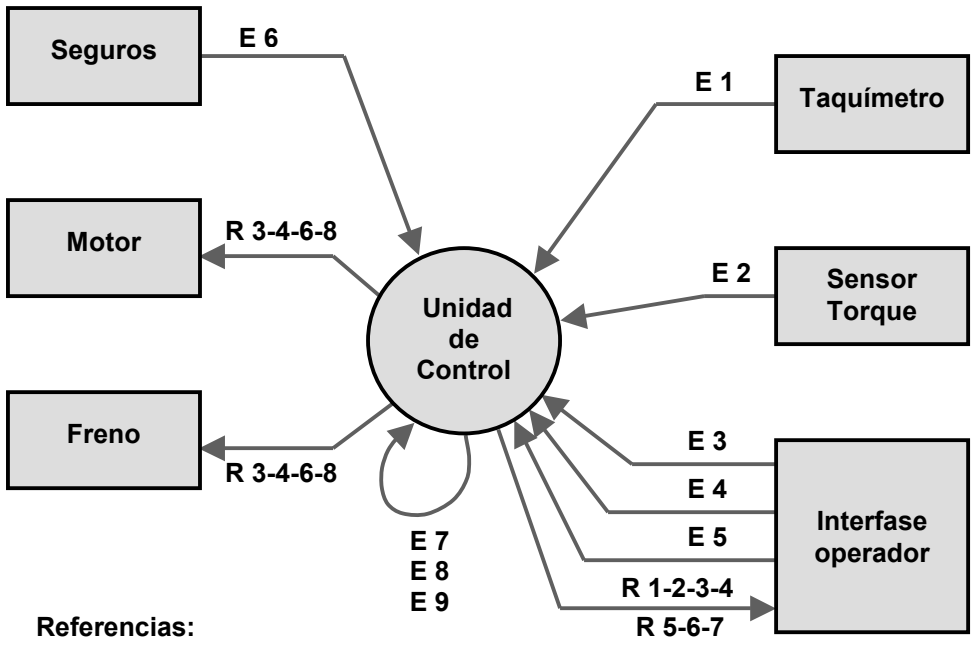
Tabla 3: Entradas al LEL y eventos

El resultado de este proceso de selección progresiva de eventos fue resumido en la tabla 3. Obsérvese que se ha tenido en cuenta el encadenamiento de elementos que surgen naturalmente a partir de considerar los impactos de las entradas al *LEL*.

En el caso de tratarse de elementos internos al sistema, estos encadenamientos o secuencias que vinculan las entradas al *LEL* contribuyen a reconocer los flujos de control, luego van a facilitar la definición de las actividades esenciales y serán muy importantes para permitir la trazabilidad en el modelo.

4.6.4 Diagrama de contexto

Considerando exclusivamente la interacción mutua entre los elementos externos y el propio sistema, se representa el diagrama de contexto, que es presentado en la Figura 41:



Referencias:

1. Taquímetro informa nueva velocidad
2. Sensor de torque informa nuevo torque
3. Operador solicita parada de emergencia
4. Operador solicita final del ensayo
5. Operador altera secuencia de ensayo
6. Seguros ordenan detención del motor
7. Transición a nuevo paso de ensayo (temporal)
8. Ajuste regular de variables de control (temporal)
9. Conmutación de lógica de control (ad-hoc)

Figura 41: Diagrama de contexto del sistema

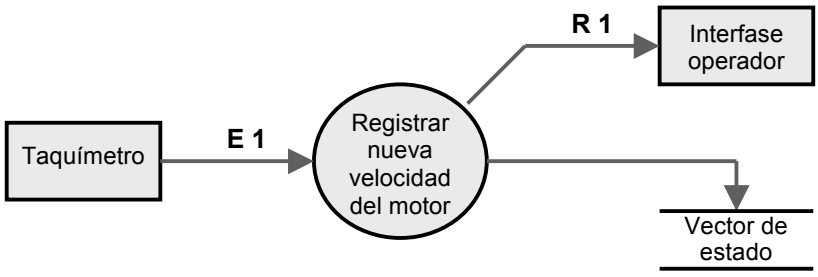
4.6.5 Tabla de eventos

Los eventos identificados en primera instancia en la Tabla 3 y representados en el diagrama de contexto, son presentados en la nueva Tabla 4 en la que se identifican sus actividades esenciales.

EVENTO					ACTIVIDAD ESENCIAL	RESPUESTA	DESTINO
Nº	**Denominación**	**Tipo**	**Fuente**	**Estímulo**			
1	Taquímetro informa nueva velocidad	ext.	Taquí-metro	Mensaje	Recibir y registrar nueva velocidad del motor	Mensaje con valor de velocidad	Interfase operador
2	Sensor de torque informa nuevo torque	ext.	Sensor de Torque	Mensaje	Recibir y registrar nuevo torque en el freno	Mensaje con valor de Torque	Interfase operador
3	Operador solicita parada de emergencia	ext.	Interfase operador	Mensaje	Detener el motor en forma inmediata	Mensaje de detención del motor	Motor y Freno Interfase operador
4	Operador solicita final del ensayo	ext.	Interfase operador	Mensaje	Terminar secuencia de ensayo y detener el motor	Mensajes de fin de ensayo y detención del motor	Motor y Freno Interfase operador
5	Operador altera secuencia de ensayo	ext.	Interfase operador	Mensaje	Alterar la secuencia de ensayo	Mensaje con nuevo paso de ensayo activo	Interfase operador
6	Seguros ordenan detención del motor	ext.	Seguros	Mensaje	Detener el motor en forma inmediata	Mensaje de detención del motor	Motor y Freno Interfase operador
7	Transición a nuevo paso de ensayo	Tpo	Unidad de Control	Intervalo de Tiempo	Cumplir las previsiones del programa de ensayo	Mensaje con nuevo paso de ensayo activo	Interfase operador
8	Ajuste regular de variables de control	Tpo	Unidad de Control	Intervalo de Tiempo	Asegurar una condición estable de operación	Señales de control de motor y freno	Motor y Freno
9	Conmutación de lógica activa de control	cond. esp.	Unidad de Control	Condición de vector de estado	Definir la lógica activa de control más apropiada	----	----

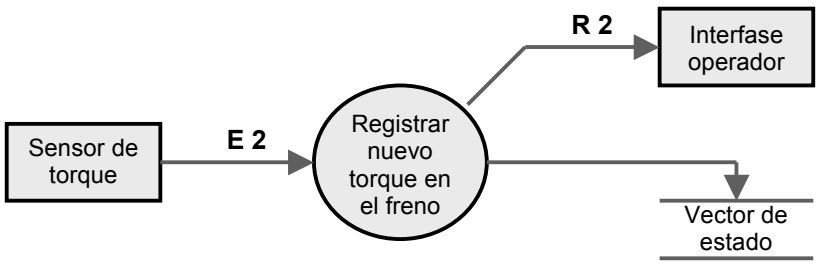
Tabla 4: Nómina de eventos

4.6.6 Fichas de eventos

N° Evento	**1**
Denominación	Taquímetro informa nueva velocidad
Tipo	Externo
Descripción	El taquímetro ha completado la determinación del último valor de velocidad del motor y la transmite a la unidad de control, que debe actualizar su vector de estado.
Fuente	Taquímetro
Estímulo	Mensaje conteniendo el valor de la velocidad del motor expresada en rpm.
Actividad esencial	Recibir y registrar nueva velocidad del motor: • Completa recepción de mensaje • Controla consistencia del mensaje recibido • Decodifica valor de velocidad del motor • Actualiza vector de estado del sistema • Transmite valor de velocidad del motor a la interfase con operador
Respuesta	Mensaje con valor de la velocidad del motor
Destino	Interfase con operador
Efecto	Actualiza condición de operación del motor
DFD	

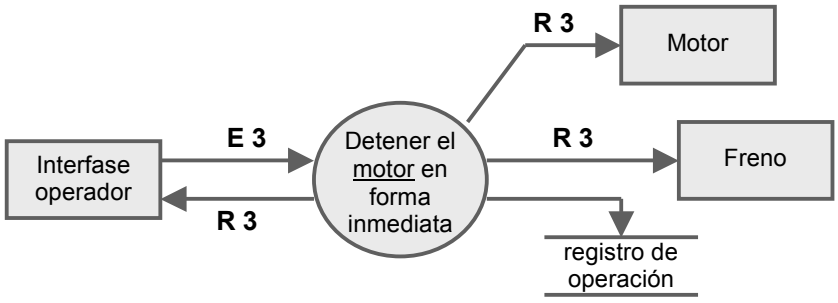
Cuadro 1: Ficha de evento N°.1

Fichas de eventos (cont.)

N° Evento	**2**
Denominación	Sensor de torque informa nuevo torque
Tipo	Externo
Descripción	El sensor de torque ha completado la determinación del último valor de torque del freno y lo transmite a la unidad de control, que debe actualizar su vector de estado.
Fuente	Sensor de torque
Estímulo	Mensaje conteniendo el valor del torque del freno expresada en Kg.m.
Actividad esencial	Recibir y registrar nuevo torque en el freno : • Completa recepción de mensaje • Controla consistencia del mensaje recibido • Decodifica valor de torque del freno • Actualiza vector de estado del sistema • Transmite valor de torque del freno a interfase con operador
Respuesta	Mensaje con valor del torque del freno
Destino	Interfase con operador
Efecto	Actualiza condición de operación del freno
DFD	

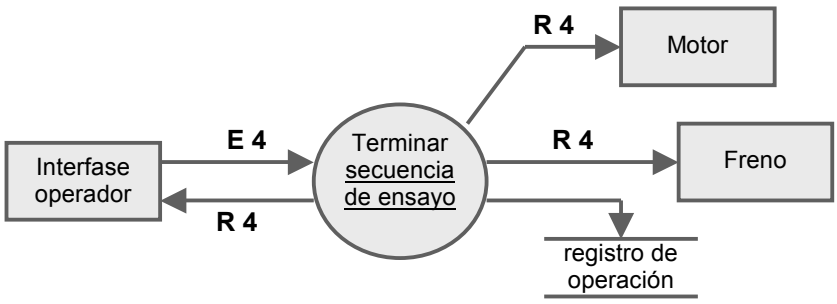
Cuadro 2: Ficha de evento N°.2

Fichas de eventos (cont.)

N° Evento	**3**
Denominación	Operador solicita parada de emergencia
Tipo	Externo
Descripción	El pulsador de Parada de Emergencia es presionado y debe detenerse el motor en forma inmediata
Fuente	Interfase con operador
Estímulo	Mensaje de Parada de Emergencia
Actividad esencial	Detener el motor en forma inmediata: • Completa recepción de mensaje • Controla consistencia del mensaje recibido • Interpreta mensaje • Ordena detención del motor (señal de control) • Deshabilita freno dinamométrico (señal de control) • Actualiza registro de operación • Actualiza Interfase con operador
Respuesta	Mensajes de parada del motor, deshabilita freno y condición de operación
Destino	Motor, Freno Dinamométrico e Interfase con operador
Efecto	Detención inmediata del motor
DFD	

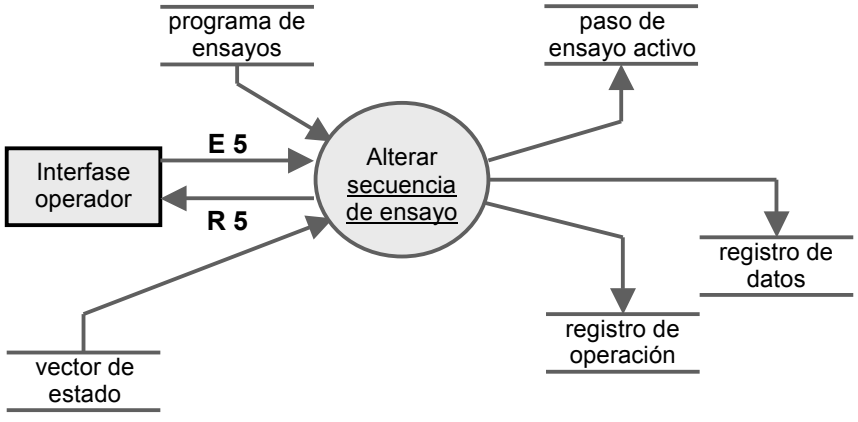
Cuadro 3: Ficha de evento N°.3

Fichas de eventos (cont.)

N° Evento	**4**
Denominación	Operador solicita final del ensayo
Tipo	Externo
Descripción	Se solicita el final del ensayo (fin de operación en modo automático) y la detención del motor
Fuente	Interfase con operador
Estímulo	Mensaje de Fin de Ensayo
Actividad esencial	Terminar secuencia de ensayo y detener el motor: • Completa recepción del mensaje • Controla consistencia del mensaje recibido • Interpreta mensaje • Finaliza operación en Modo Automáticol • Ordena detención del motor (señal de control) • Deshabilita freno dinamométrico (señal de control) • Actualiza registro de operación • Actualiza interfase con operador
Respuesta	Mensajes de parada del motor, deshabilita freno y condición de operación
Destino	Motor, Freno Dinamométrico e Interfase con operador
Efecto	Fin del Ciclo de ensayo y detención del motor
DFD	

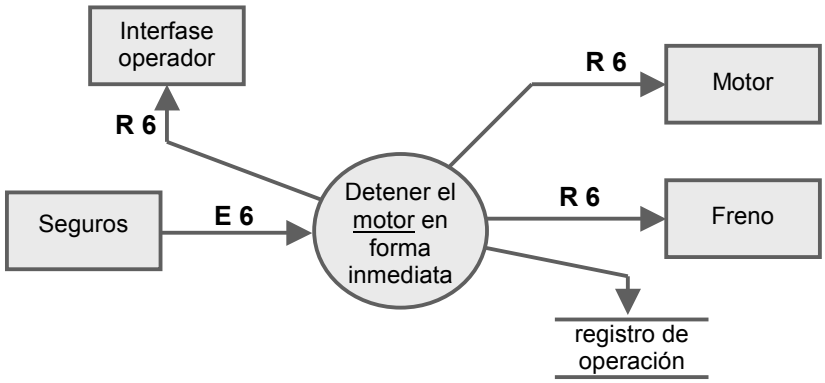
Cuadro 4: Ficha de evento N°.4

Fichas de eventos (cont.)

N° Evento	**5**
Denominación	Operador altera secuencia de ensayo
Tipo	Externo
Descripción	El operador ordena alterar la secuencia prevista en el programa de ensayo, con avance a próximo paso o retroceso a paso anterior.
Fuente	Interfase con operador
Estímulo	Mensaje con el comando que corresponda
Actividad esencial	Alterar la secuencia de ensayo prevista en el programa: • Completa recepción de mensaje • Controla consistencia del mensaje recibido • Interpreta mensaje • Almacena registro de datos en paso actual • Actualiza registro de paso de ensayo activo • Actualiza registro de operación • Actualiza interfase con operador
Respuesta	Mensaje informativo
Destino	Interfase con operador
Efecto	Cambiar la condición de operación del conjunto motor-freno
DFD	

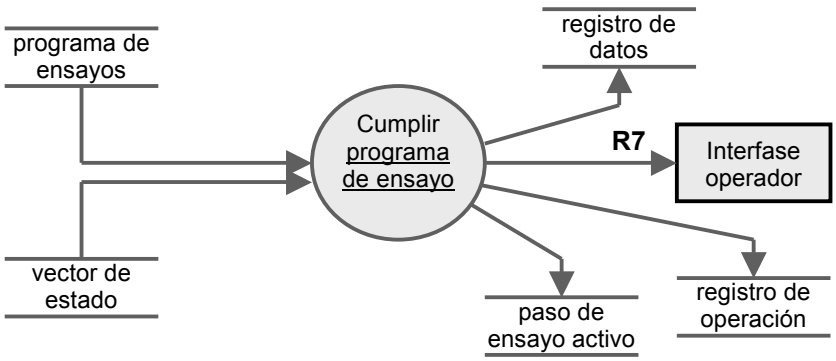
Cuadro 5: Ficha de evento N°.5

Fichas de eventos (cont.)

N° Evento	**6**
Denominación	Seguros ordenan detención del motor
Tipo	Externo
Descripción	Debe detenerse el motor en forma inmediata. Se presentó una condición de anomalía en el motor, freno dinamométrico o instalaciones auxiliares
Fuente	Seguros
Estímulo	Mensaje indicando anomalía en seguros.
Actividad esencial	Detener el motor en forma inmediata: • Completa recepción de mensaje • Controla consistencia del mensaje recibido • Decodifica e identifica seguro • Ordena detención del motor (señal de control) • Deshabilita freno dinamométrico (señal de control) • Actualiza registro de operación • Actualiza interfase con operador
Respuesta	Mensajes de parada del motor, deshabilita freno dinamométrico y condición de operación
Destinos	Motor , freno dinamométrico e Interfase con el operador
Efecto	Detener el motor e identificar la causa de la falla
DFD	

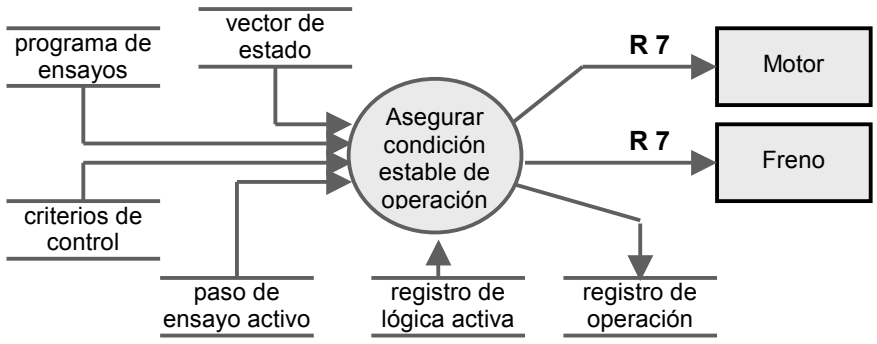
Cuadro 6: Ficha de evento N°.6

Fichas de eventos (cont.)

N° Evento	**7**
Denominación	Transición a nuevo paso de ensayo
Tipo	Temporal
Descripción	Activa próximo paso de ensayo según previsión de programa de ensayo
Fuente	Sistema de Control
Estímulo	
Actividad esencial	Cumplir las previsiones del programa de ensayo: • Lee programa de ensayo • Lee vector de estado • Almacena registro de datos en paso actual • Determina próximo paso • Actualiza registro de paso de ensayo activo • Actualiza registro de operación • Actualiza interfase con operador
Respuesta	Mensaje de nueva condición de operación
Destino	Interfase con operador
Efecto	Cambiar la condición de operación del conjunto motor-freno
DFD	

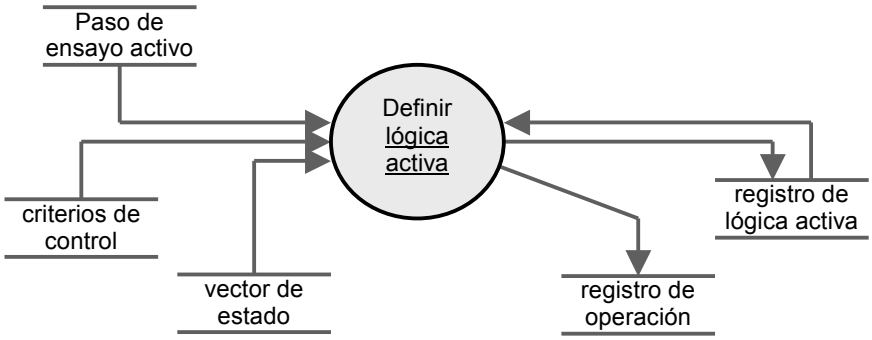
Cuadro 7: Ficha de evento N°.7

Fichas de eventos (cont.)

N° Evento	**8**
Denominación	Ajuste regular de variables de control
Tipo	Temporal
Descripción	Ajuste regular de las variables de control para asegurar una operación estable acorde a las exigencias del programa de ensayo
Fuente	Sistema de Control
Estímulo	
Actividad esencial	Asegurar una condición estable de operación : • Lee velocidad y torque de vector de estado • Lee programa de ensayo • Lee paso de ensayo activo • Calcula error de variables controladas • Lee criterios de control • Lee lógica activa • Calcula ajuste necesario en variables de control • Determina nuevos valores de las variables de control del Motor y freno dinamométrico • Actualiza registro de operación
Respuesta	Mensajes de control
Destino	Motor y freno dinamométrico
Efecto	Hacer mínima las desviaciones respecto de los set-points
DFD	

Cuadro 8: Ficha de evento N°.8

Fichas de eventos (cont.)

N° Evento	**9**
Denominación	Conmutación de lógica activa de control
Tipo	Condición especial
Descripción	Conmutación de la lógica de control para asegurar la operación dentro de los límites establecidos
Fuente	Sistema de Control
Estímulo	
Actividad	Definir la lógica activa de control mas apropiada: • Lee vector de estado • Lee paso de ensayo activo • Lee el registro de lógica activa • Lee criterios de control • Identifica condiciones de conmutación • Selecciona la lógica activa más apropiada • Actualiza registro de la lógica activa a ser usada • Actualiza registro de operación
Respuesta	
Destino	
Efecto	Seleccionar la lógica de control más apropiada para operar el conjunto motor-freno dentro de los límites establecidos
DFD	

Cuadro 9: Ficha de evento N°.9

4.6.7 Modelo de eventos del sistema (diagrama 0)

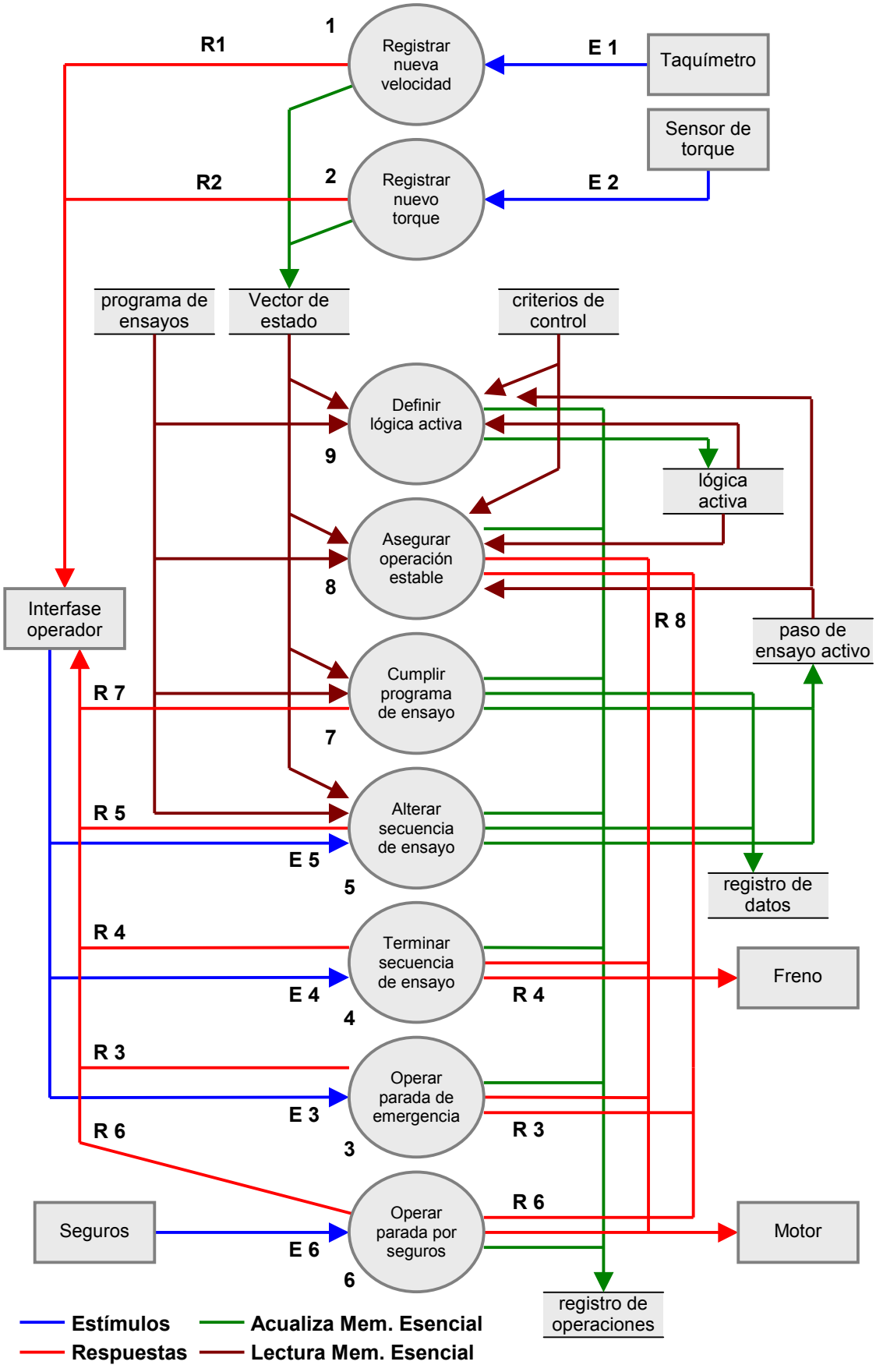


Figura 42: Modelo de eventos (diagrama “0”)

4.6.8 Matriz de eventos - datos

A partir de los eventos del sistema, puede armarse una matriz que resume la forma en que cada uno opera sobre los datos, representada en la Tabla 5 tal como lo sugiere David Ruble (Ruble, 1997):

Evento	Programa de ensayo	Vector de Estado	Criterios de Control	Lógica activa	Paso de ensayo activo	Registro de datos	Registro de operación
1		U					
2		U					
3							C
4	R						C
5	R	R			U	C	C
6							C
7	R	R			U	C	C
8	R	R	R	R	R		C
9	R	R	R	R,U			C

Tabla 5: Matriz de eventos

Referencias:
- C : Carga de nuevo registro (Create)
- R : Lectura de registro (Read)
- U : Actualización (Update)
- D : Borrado (Delete)

4.6.9 Diagrama de Entidad-Relación (DER)

Una primera versión de un diagrama de Entidad-Relación es obtenida a partir del *Diagrama 0* y de las descripciones contenidas en las *Fichas de Eventos*. La consistencia de este diagrama puede confirmarse en la *Matriz de Eventos-Datos*, donde cada relación debe estar respaldada por algún evento que contenga ambas entidades.

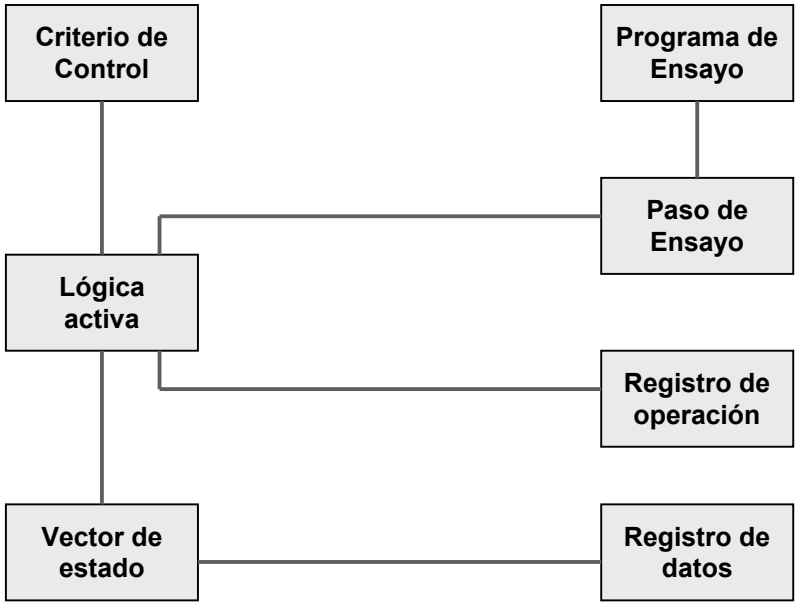


Figura 43: Diagrama de Entidad-Relación

4.6.10 Estructuras de almacenes de datos

Nombre almacén	Estructura del registro
Paso de ensayo	Tiempo de permanencia
	Velocidad
	Torque
	Identificador de próximo paso
Vector de estado	Velocidad
	Torque en el freno
	Velocidad extrapolada a tiempo de referencia
	Torque de freno extrapolado a tiempo de referencia
	Torque de motor calculado en tiempo de referencia
	Derivada de velocidad
	Derivada de torque del freno
	Derivada de torque del motor
Criterio de control	Antecedente (regla de producción)
	Consecuente (regla de producción)
Lógica activa	Identificador de opción de control
	Identificador de realimentación PID
Registro de datos	Tiempo de ensayo
	Identificador de paso de ensayo activo
	Identificador de lógica activa
	Velocidad
	Torque en freno
	Torque en motor
Registro de operación	Tiempo de ensayo
	Código de operación
	Argumento de operación

Tabla 6: Estructuras de almacenes de datos

4.6.11 Control de completitud

Para implementar un control de completitud, se ha recurrido a una representación como la presentada en la Tabla 7, donde cada fila corresponde a una entrada al vocabulario *LEL*, el primer grupo de columnas corresponde a los eventos y el segundo grupo de columnas representa las actividades esenciales y sus procedimientos. En esta tabla, una cruz significa que:

- La entrada al LEL está presente en la descripción del evento.
- La referencia a la entrada al LEL está en la descripción de las funciones de la actividad esencial.
- La entrada fue utilizada al definir la noción o impacto de una entrada que cumplió alguna de las condiciones anteriores. En este caso, se utiliza la última columna.

Una condición necesaria pero no suficiente de completitud del modelo establece que: toda entrada al *LEL* esté presente en algún evento, actividad esencial o fue necesaria para definir las anteriores. Se comprueba así que:

a) Cada evento esté asociado a algún término del LEL.

b) Cada actividad esencial involucre uno o más término del LEL.

Utilizando hipervínculos podría desarrollarse una herramienta que brinde un diagnóstico automático sobre la completitud del modelo, identificando las entradas al *LEL* no referenciadas o los eventos y actividades esenciales que no están

respaldadas. En este caso, el control de completitud fue implementado manualmente.

Entradas LEL	Eventos									Actividades esenciales									Otros
	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6	7	8	9	
	Taquímetro informa nueva velocidad	Sensor de torque informa nuevo torque	Operador solicita parada de emergencia	Operador solicita final del ensayo	Operador altera secuencia de ensayo	Seguros ordenan detención del motor	Transición a nuevo paso de ensayo	Ajuste regular de variables de control	Conmutación de lógica activa de control	Registra nuevo valor de velocidad	Registra nuevo valor de torque	Opera parada de emergencia	Opera final del ensayo	Actualiza el registro de paso de ensayo activo	Operar parada por seguros	Interpreta programa de ensayo	Ajusta variables de control	Conmutar lógica de control	
1										X		X	X		X		X		
2											X	X	X		X		X		
3																			X
4																			X
5	X																		
6		X																	
7						X													
8																			X
9																			X
10																			X
11																			X
12																			X
13										X	X					X	X	X	
14																			X
15																	X		
16																			X
17																	X		
18																			X
19																	X		
20																			X
21																			X
22								X									X		
23																			X
24																			X
25													X						
26																			X
27																			X
28																			X
29									X								X		
30							X							X		X			X
31														X		X	X		
32														X			X		
33					X								X			X			
34			X																
35				X															
36																			X
37										X	X	X	X	X	X	X			

Tabla 7: Control de completitud del modelo

4.6.12 Actividades esenciales y objetos

La tarea de establecer vínculos entre las actividades esenciales y los objetos puede facilitarse enormemente con un ordenamiento como el de la Tabla 8, donde se enumeran las actividades esenciales, se identifican las responsabilidades asociadas y a cada una de ellas se le asigna el objeto que lo contendrá. Este proceso se apoya en los siguientes dos postulados básicos:

a) La funcionalidad global de un sistema queda completamente definida por su Modelo Esencial.
b) El conjunto de las responsabilidades de todas las clases representa lo que el sistema en su conjunto es capaz de realizar.

Al plantearse esta relación, se están equilibrando los modelos Esenciales y de Objetos, ya que cada una de las ***N*** capacidades del sistema debe estar presente en ambos modelos, es decir que:

$$\boldsymbol{f}_k(E) = \boldsymbol{f}'_k(O) \quad ; \quad k = 1, ..., \boldsymbol{N} \qquad (102)$$

Actividades esenciales		Responsabilidades	Objetos	
1	Recibir y registrar nueva velocidad del motor	Recibir un mensaje del taquímetro	OperadorDeMensajes	**2**
		Controlar consistencia y decodificarlo	DecodificadorDeMensajes	**3**
		Operar mensajes del taquímetro	InterfaseTaquimetro	**1**
		Actualizar vector de estado	VectorDeEstado	**4**
		Informar nueva velocidad a la Interfase con el operador	InterfaseConOperador	**5**
2	Recibir y registrar nuevo torque en el freno	Recibir un mensaje del Sensor de Torque	OperadorDeMensajes	**2**
		Controlar consistencia y decodificarlo	DecodificadorDeMensajes	**3**
		Operar mensajes del sensor de torque	InterfaseSensorTorque	**6**
		Actualizar vector de estado	VectorDeEstado	**4**
		Informar nuevo torque a la Interfase con el operador	InterfaseConOperador	**5**
3	Detener el motor en forma inmediata por parada de emergencia	Recibir un mensaje de la Interfase con el Operador	OperadorDeMensajes	**2**
		Operar orden de detención por parada de emergencia	InterfaseConOperador	**5**
		Controlar consistencia y decodificarlo	DecodificadorDeMensajes	**3**
		Ordenar detención del motor	ControlaMotorFreno	**7**
		Informar parada de emergencia a la Interfase con el operador	InterfaseConOperador	**5**
		Actualiza registro de operaciones	RegistroDeOperaciones	**8**

Tabla 8: Relación de actividades y objetos

Actividades esenciales y objetos (Cont.)

Actividades esenciales		Responsabilidades	Objetos	
4	Terminar secuencia de ensayo y detener el motor	Recibir un mensaje de la Interfase con el Operador	OperadorDeMensajes	**2**
		Operar orden de finalizar el ensayo y detener el motor	InterfaseConOperador	**5**
		Controlar consistencia y decodificarlo	DecodificadorDeMensajes	**3**
		Finalizar la operación en Modo Automático	IntérpreteDeEnsayos	**9**
		Ordenar detención del motor	ControlaMotorFreno	**7**
		Informar fin de ensayo a la Interfase con el operador	InterfaseConOperador	**5**
		Actualiza registro de operaciones	RegistroDeOperaciones	**8**
5	Alterar secuencia de ensayo según orden del operador	Recibir un mensaje de la Interfase con el Operador	OperadorDeMensajes	**2**
		Operar orden de cambiar la secuencia de ensayo	InterfaseConOperador	**5**
		Controlar consistencia y decodificar el mensaje	DecodificadorDeMensajes	**3**
		Almacenar condición del ensayo	RegistroDeDatos	**10**
		Leer el Programa de Ensayos	ProgramaDeEnsayos	**12**
		Leer el Vector de Estado	VectorDeEstado	**4**
		Determinar próximo paso de ensayo	IntérpreteDeEnsayos	**9**
		Actualizar registro de Paso de Ensayo Activo	RegistroDePasoActivo	**11**
		Actualiza registro de operaciones	RegistroDeOperaciones	**8**
		Informar de nuevo paso activo a interfase con operador	InterfaseConOperador	**5**
6	Detener el motor en forma inmediata por condición de los seguros	Operar orden de seguros de detener el motor	OperadorDeMensajes	**2**
		Recibir un mensaje de los seguros	InterfaseSeguros	**13**
		Controlar consistencia y decodificar el mensaje	DecodificadorDeMensajes	**3**
		Ordenar detención del motor	ControlaMotorFreno	**7**
		Informar detención del motor a la Interfase con el operador	InterfaseConOperador	**5**
		Actualiza registro de operaciones	RegistroDeOperaciones	**8**

Tabla 8: Relación de actividades y objetos

Actividades esenciales y objetos (Cont.)

Actividades esenciales		Responsabilidades	Objetos	
7	Transitar a un nuevo paso de ensayo según previsión del programa	Controlar el tiempo de ensayo	ControlSecundario	**14**
		Leer el Programa de Ensayos	ProgramaDeEnsayos	**12**
		Leer el Vector de Estado	VectorDeEstado	**4**
		Almacenar condición del ensayo	RegistroDeDatos	**10**
		Determinar próximo paso de ensayo	InterpreteDeEnsayos	**9**
		Actualizar registro de Paso de Ensayo Activo	RegistroDePasoActivo	**11**
		Actualiza registro de operaciones	RegistroDeOperaciones	**8**
		Informar nuevo paso activo a la Interfase con el operador	InterfaseConOperador	**5**
8	Asegurar una condición estable de operación en el conjunto motor-freno	Determinar los ajustes necesarios en las variables de control	ControlPrimario	**15**
		Lee Vector de Estado	VectorDeEstado	**4**
		Leer el Programa de Ensayos	ProgramaDeEnsayos	**12**
		Lee Registro de Paso de Ensayo Activo	RegistroDePasoActivo	**11**
		Lee Criterios de Control	TablaCriteriosDeControl	**16**
		Lee registro de Lógica Activa	Registro de Lógica Activa	**17**
		Ajusta variables de control	AjustaVariablesControlPID	**18**
		Almacenar condición del ensayo	RegistroDeDatos	**10**
		Cambiar condiciones del Motor y Freno	OperaMotorFreno	**7**
		Actualiza registro de operaciones	RegistroDeOperaciones	**8**
9	Seleccionar la Lógica de Control más conveniente	Seleccionar la Lógica de Control	ControlPrimario	**15**
		Lee Registro de Paso de Ensayo Activo	RegistroDePasoActivo	**11**
		Lee Vector de Estado	VectorDeEstado	**4**
		Lee Criterios de Control	TablaCriteriosDeControl	**16**
		Determina Lógica Activa	SeleccionaLógicaActiva	**19**
		Almacena nueva lógica de control	RegistroDeLogicaActiva	**17**
		Actualiza registro de operaciones	RegistroDeOperaciones	**8**

Tabla 8: Relación de actividades y objetos

4.6.13 Clases, responsabilidades y colaboraciones

El ordenamiento anterior ha permitido identificar los objetos directamente relacionados con las actividades esenciales. Luego, para resumir las responsabilidades de cada uno y para establecer las relaciones entre ellos se recurre a las fichas *CRC*.

Este proceso que ha permitido la identificación de las principales clases será así completado siguiendo muchas de las recomendaciones habituales para el manejo de las fichas *CRC*, de manera de facilitar la asignación de responsabilidades, definir la jerarquía entre clases e identificar clases auxiliares o de servicios. Para ello se ha tomado como referencia el trabajo de David Bellin y Susan Suchman Simone (Bellin, 1997).

Para la identificación de las clases, Bellin y Suchman sugieren en su trabajo dos técnicas grupales: “brainstrorming” y “role playing”. Si bien la eficacia de estas técnicas puede ser discutible para los casos en que se debe partir desde cero, es indudable que representan una alternativa válida para perfeccionar modelos que están muy avanzado y donde deben ajustarse sus detalles e interrelaciones.

Se presentan a continuación las fichas que definen las clases, sus responsabilidades y colaboradores (*CRC*) que resultan de este proceso:

Clase: 1 InterfaseTaquímetro	
Superclases:	
Subclases:	
Descripción: Al recibirse un mensaje del taquímetro, es necesario leerlo, comprobar su consistencia, decodificarlo, actualizar el vector de estado e informar a la interfase con el operador sobre la nueva velocidad.	
Responsabilidades: - Operar Mensajes de taquímetros	**Colaboradores:** - VectorDeEstado (4) - InterfaseConOperador (5)

Clase: 2 OperadorDeMensajes	
Superclases:	
Subclases:	
Descripción: El sistema recibe mensajes que contienen información proveniente del taquímetro, sensor de torque, seguro e interfase con el operador. Estos mensajes deben ser leídos para cada caso.	
Responsabilidades: - Leer mensaje de taquímetro - Leer mensaje de sensor de torque - Leer mensaje de seguros - Leer mensaje de interfase c/operador - Transmitir mensajes a interfase c/op. - Transmitir mensajes a motor/freno	**Colaboradores:** - AdministradorPuertaDeEntrada (20) - InterfaseTaquimetro (1) - InterfaseConOperador (5) - InterfasSensorTorque (6) - InterfaseSeguros (13) - DecodificadordeMensajes (3)

Clase: 3 DecodificadorDeMensajes	
Superclases:	
Subclases:	
Descripción: Según su origen, los mensajes contienen números enteros, reales, comandos o textos. Estos contenidos deben ser reconocidos y validados.	
Responsabilidades: - Decodificar mensaje de taquímetro - Decodificar mensaje de torque - Decodificar mensaje de seguros - Decodificar mensaje de interfase con operador	**Colaboradores:**

Clase: 4 VectorDeEstado	
Superclases:	
Subclases:	
Descripción: Actualizar y lee datos de variables del vector de estado. Debe también extrapolarar nuevos valores y derivarlos.	
Responsabilidades: - Actualiza velocidad en vector de estado - Actualiza torque en vector de estado - Lee velocidad en vector de estado - Lee torque en vector de estado - Extrapola velocidad - Extrapola torque - Deriva velocidad - Deriva torque	**Colaboradores:** - ExtrapoladorDerivador (21)

Clase: 5 InterfaseConOperador	
Superclases:	
Subclases:	
Descripción: Se reciben y transmiten mensajes a la Interfase con el Operador. Estos mensajes deben ser preparados o interpretados según su origen y destino.	
Responsabilidades: - Informar nueva velocidad - Informar nuevo torque - Informar parada de emergencia - Informar fin de ensayo - Informar cambio en secuencia ensayo - Informar detención del motor - Informar nuevo paso activo - Recibir mensajes del operador - Operar parada de emergencia - Operar fin del ensayo - Operar cambio en secuencia ensayo	**Colaboradores:** - ControlaMotorFreno (7) - InterpreteDeEnsayo (9)

Clase: 6 InterfaseSensorTorque	
Superclases:	
Subclases:	
Descripción: Al recibirse un mensaje del sensor de torque, es necesario leerlo, comprobar su consistencia, decodificarlo, actualizar el vector de estado e informar a la interfase con el operador sobre el nuevo torque.	
Responsabilidades: - Operar Mensajes de sensor de torque	**Colaboradores:** - VectorDeEstado (4) - InterfaseConOperador (5)

Clase: 7 ControlaMotorFreno	
Superclases:	
Subclases:	
Descripción: Operar sobre el motor y freno dinamométrico a través de señales digitales de habilitación o de las variables de control.	
Responsabilidades: - Habilitar el motor - Habilitar el freno dinamométrico - Deshabilitar el motor - Deshabilitar el freno dinamométrico - Definir posición del acelerador - Definir capacidad de frenado	**Colaboradores:** - OperadorDeMensajes(2) - RegistroDeOperaciones (8)

Clase: 8 RegistroDeOperaciones	
Superclases:	
Subclases:	
Descripción: Registrar la actividad cumplida por el sistema en un medio de almacenamiento permanente.	
Responsabilidades: - Registrar paradas de emergencia - Registrar fin de ensayos - Registrar cambios en secuencias de ensayo - Registrar paradas por seguros - Registrar transiciones a nuevo paso de ensayo - Regsitrar regulación de las variables de control - Registrar conmutación de lógica de control	**Colaboradores:**

Clase: 9 **IntérpreteDeEnsayos**	
Superclases:	
Subclases:	
Descripción: Reconocer los comandos del programa de ensayo.	
Responsabilidades: - Habilitar el modo automático de ensayo - Volver a modo manual de ensayo - Determinar próximo paso de ensayo - Determinar paso anterior	**Colaboradores:** - VectorDeEstado(4) - InterfaseConOperador (9) - RegistroDeDatos (10) - RegistroDePasoActivo(11) - ProgramaDeEnsayos (12)

Clase: 10 **RegistroDeDatos**	
Superclases:	
Subclases:	
Descripción: Almacenar registros hsitóricos del vector de estado en un medio permanente.	
Responsabilidades: - Almacenar registros de datos de ensayo	**Colaboradores:**

Clase: 11 **RegistroDePasoActivo**	
Superclases:	
Subclases:	
Descripción: Registrar y leer el paso de ensayo que está activo	
Responsabilidades: - Registrar paso de ensayo activo - Leer paso de ensayo activo	**Colaboradores:**

Clase: 12 **ProgramaDeEnsayos**	
Superclases:	
Subclases:	
Descripción: Leer registros de pasos de ensayo	
Responsabilidades: - Leer paso de ensayos	**Colaboradores:**

Clase: 13 **InterfaseSeguros**	
Superclases:	
Subclases:	
Descripción: Al recibirse un mensaje de la interfase de seguros, es necesario leerlo, comprobar su consistencia, decodificarlo y actuar en consecuencia.	
Responsabilidades: - Operar Mensajes de seguros	**Colaboradores:** - ControlaMotorFreno (7) - InterfaseConOperador (5)

Clase: 14 **ControlSecundario**	
Superclases:	
Subclases:	
Descripción: Verificar regularmente la recepción de mensajes y el cumplimiento del programa de ensayos.	
Responsabilidades: - Interpretar el programa de ensayos - Verificar la presencia de mensajes	**Colaboradores:** - IntérpreteDeEnsayos (9) - OperadorDeMensajes (2)

Clase: 15 ControlPrimario	
Superclases:	
Subclases:	
Descripción: Seleccionar la lógica de control más apropiada y determinar los ajustes necesarios en las variables de control.	
Responsabilidades: - Seleccionar la lógica de control - Ajustar las variables de control	**Colaboradores:** - AjustaVariablesControlPID (18) - SeleccionaLogicaActiva (19)

Clase: 16 TablaCriteriosDeControl	
Superclases:	
Subclases:	
Descripción: Acceder a la tabla de criterios de control, leerlos y decodificarlos. Los criterios de control están expresados como reglas de producción.	
Responsabilidades: **-** Leer criterios de control	**Colaboradores:**

Clase: 17 RegistroDeLogicaActiva	
Superclases:	
Subclases:	
Descripción: Registrar la lógica se se considera más conveniente en cada caso.	
Responsabilidades: - Leer el registro de lógica activa - Actualizar el registro de lógica activa	**Colaboradores:**

Clase: 18 **AjustaVariableDeControlPID**	
Superclases:	
Subclases:	
Descripción: Determinar los valores de los ajustes necesarios en las variables de control a partir de los errores en las variables controladas y del criterio de control.	
Responsabilidades: - Determinar los ajustes en las variables de control	**Colaboradores:** - VectorDeEstado(4) - RegistroDePasoActivo(11) - ProgramaDeEnsayos (12) - TablaCriteriosDeControl (16) - Registro de Lógica Activa (17) - ControlaMotorFreno (7)

Clase: 19 **SeleccionaLogicaActiva**	
Superclases:	
Subclases:	
Descripción: Seleccionar la lógica más conveniente, según el vector de estado, los criterios de control y el programa de ensayos.	
Responsabilidades **-** Determina la lógica más conveniente	**Colaboradores :** - VectorDeEstado(4) - RegistroDeOperaciones (8) - RegistroDePasoActivo(11) - ProgramaDeEnsayos (12) - TablaCriteriosDeControl (16) - Registro de Lógica Activa (17)

Clase: 20 **AdministradorPuertaDeEntrada**	
Superclases:	
Subclases:	
Descripción: Recibir y transmitir mensajes a través de la puerta de comunicaciones	
Responsabilidades: **-** Opera puerta de comunicaciones	**Colaboradores:**

Clase: 21 ExtrapoladorDerivador	
Superclases:	
Subclases:	
Descripción: Extrapolar y derivar valores sobre una secuencia de datos conocidos.	
Responsabilidades: - Extrapolar una cierta función - Derivarla en el punto extrapolado	**Colaboradores:**

4.6.14 Diagrama de clases

Tras lo señalado, y una vez que con la ayuda de las fichas *CRC* se ha revisado y eventualmente ajustado la distribución de las responsabilidades del sistema entre sus clases, debe proseguirse con la representación del diagrama de clases. Al igual que lo que ocurre con el *Diagrama de Entidad Relación*, el *diagrama de clases* es una representación estática de la estructura de un sistema. Por ello es conveniente revisar las similitudes y diferencias entre ambos diagramas.

Debe recordarse que el *DER* fue aquí desarrollado en el marco del *análisis esencial,* mientras que el *diagrama de clases* responde al *paradigma de objetos*, lo que establece una base para realizar esta comparación.

En ambos diagramas, deberían estar representados los mismos depósitos de datos y sus atributos, sólo que en el *diagrama de clases* se incorporan todos los objetos que están asociados con los métodos que pueden transformar esos atributos.

Por razones de espacio y claridad de representación, el *diagrama de clases* es desdoblado representándose por separado el contacto de los objetos que son externos al sistema. Se obtiene así un *diagrama de contexto* orientado a objetos. Además, para facilitar su identificación se usan diferentes colores.

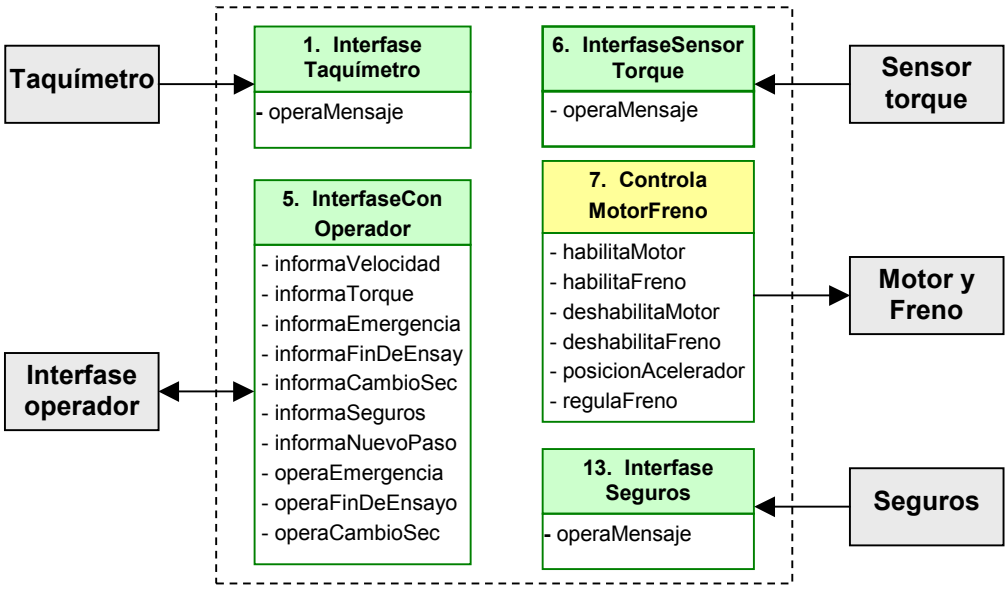


Figura 44: Diagrama de Clases / Contexto

Diagrama de clases

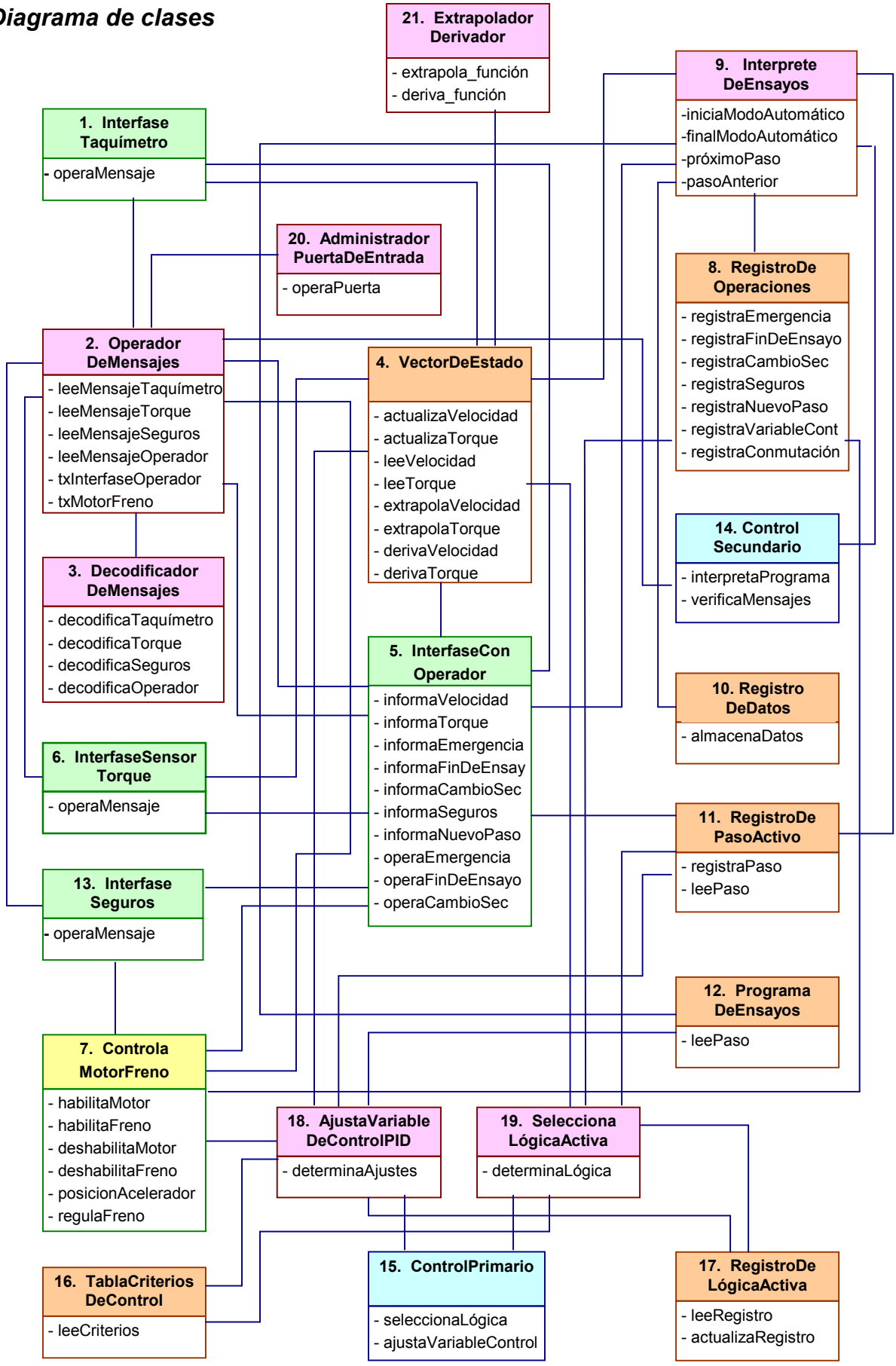


Figura 45 : Diagrama de Clases

En la Figura 45, se presenta el diagrama de clases completo y en ambos diagramas se utiliza la siguiente convención en la asignación de colores:

- Objetos externos al sistema
- Objetos en contacto con elementos externos: entradas
- Objetos en contacto con elementos externos: salidas
- Objetos activados regularmente (temporales)
- Depósitos de datos
- Transformación de datos: procesos

5 Diseño e implementación

5.1 Diseño arquitectónico

El proceso de diseño arquitectónico comprende el establecimiento de un marco de trabajo estructural básico para el sistema, lo que implica identificar sus componentes principales y la comunicación entre ellos (Sommerville, 2002). Así, uno de los aspectos principales que deben ser considerados es la forma en que se vincularán entre sí los componentes que ya fueron identificados en el esquema de la Figura 24, considerando para ello los límites que fueron establecidos en el punto 4.6.1.

Todas las funciones requeridas para resolver el problema planteado pueden ser presentadas globalmente en un esquema como el de la Figura 46, a partir del cual se identifican fácilmente los principales elementos involucrados. Obsérvese que, si bien todos ellos forman parte del sistema, la especificación de requerimientos desarrollada en el punto 4.6 se refiere exclusivamente a la unidad de control y, por lo tanto, sólo se incluyen las funciones que fueron identificadas como básicas para esta actividad. Todas los demás componentes fueron reconocidos como parte de su contexto.

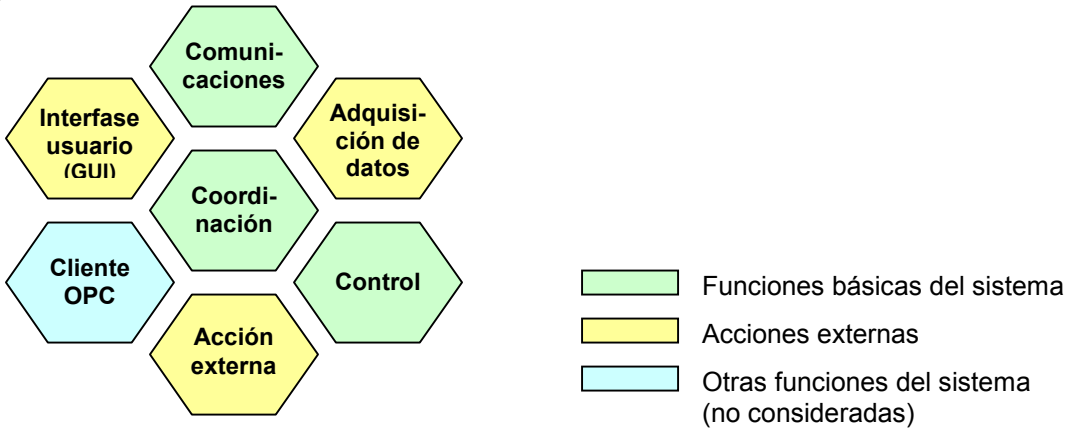


Figura 46: Funcionalidades y componentes

Para resolver este tipo de problemas se presentan dos opciones, los sistemas de control centralizados o los sistemas distribuidos.

En el primer caso, la totalidad de los elementos del sistema y sus componentes externos forman parte del hardware de la unidad de control. Aquí, las entradas y salidas analógicas y digitales están a cargo de placas de circuitos, que

son coordinados por la CPU del sistema central. Estas placas contienen los conversores A/D, conversores D/A, multiplexores, contadores y otros elementos.

En el caso de los sistemas distribuidos, las capacidades de percepción, acción y supervisión están descentralizadas y conectadas en red al sistema central a través de uno o más vínculos. Para esta comunicación, se utilizan protocolos propietarios que son habitualmente implementados en base a la norma RS-485, que presenta la ventaja de su relativa inmunidad a ambientes ruidosos (ruidos de radiofrecuencia).

Para este trabajo, se adoptó la segunda opción, es decir la de un sistema distribuido. Si bien un análisis comparativo de las dos opciones mencionadas excede el alcance de este trabajo, puede anticiparse que ambas son válidas y representan arquitecturas que son adoptadas con frecuencia en la actualidad.

A fines de implementar la arquitectura seleccionada se reconocieron los siguientes elementos principales: a) subsistema de adquisición de datos, b) subsistema de accionamiento exterior, c) interfase con el operador y d) unidad de control. El subsistema de adquisición de datos reúne al taquímetro, sensor de torque y seguros. Cabe aquí destacar que los controladores lógicos programables (*PLC*) representan, en la actualidad, una opción muy difundida para implementar los periféricos de percepción y acción (adquisición de datos y accionamiento exterior). En estos casos, se recurre a computadoras personales (*PC*) para implementar la interfase con el operador y la unidad de control. En el esquema de la figura 47 se representa la arquitectura adoptada usando, para mayor claridad, el mismo código de colores que en la Figura 46.

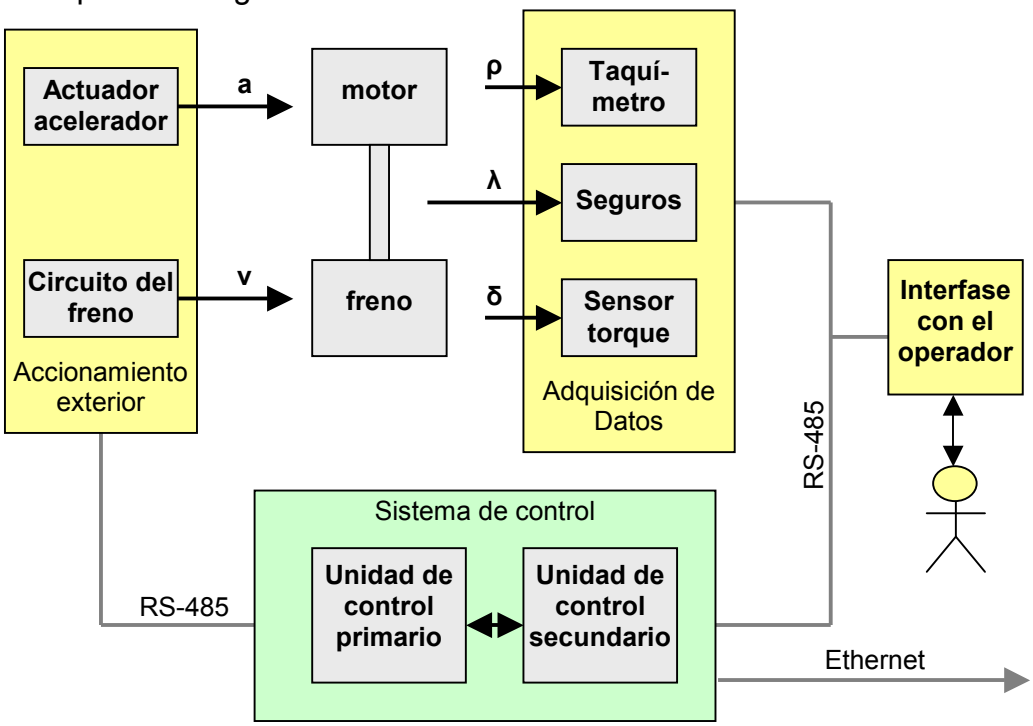


Figura 47: Arquitectura del sistema

En este tipo de arquitecturas, la unidad central coordina las comunicaciones, habilitando y consultando en forma sucesiva a los diferentes elementos de manera de evitar colisiones (*polling*). El vínculo ethernet es a su vez implementado como un cliente *OPC* de manera de asegurar completa compatibilidad con los standards del

mercado y facilitar su supervisión por parte de sistemas *SCADA* (*Supervisory Control And Data Adquisition*).

5.2 Diseño intermedio

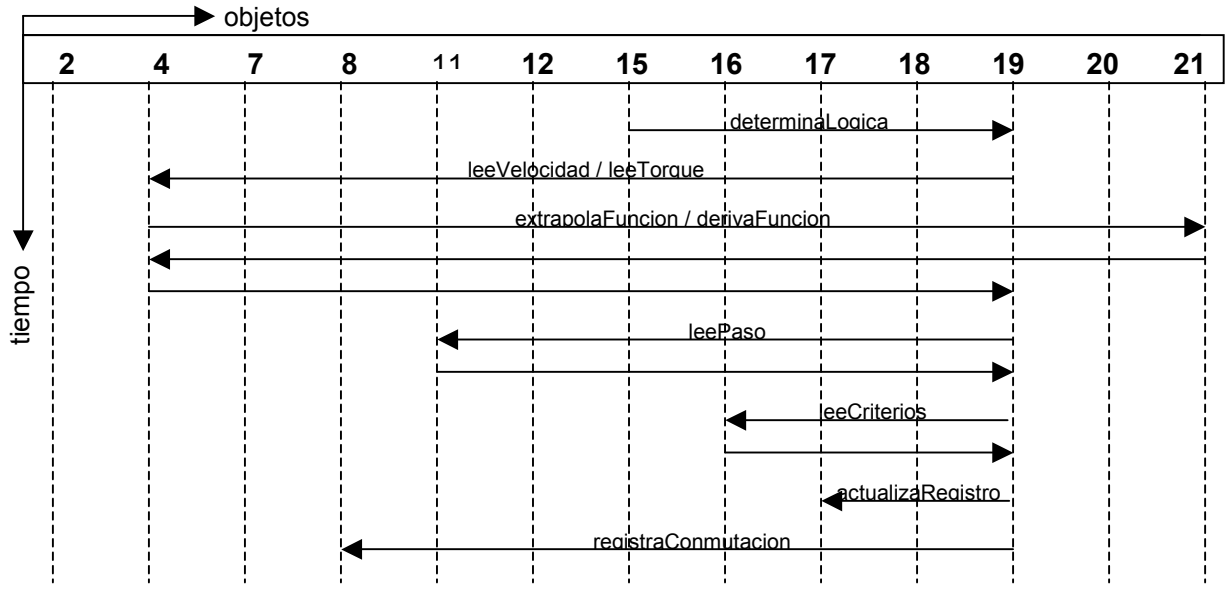
El diseño intermedio involucra un refinamiento de la solución presentada en la etapa anterior, y para ello se revisará la sucesión de operaciones que han sido previstas al identificarse las actividades principales en el modelo conceptual.

Obsérvese que, en la forma que está concebida la arquitectura del sistema, los objetos *ControlPrimario* (15) y *ControlSecundario* (14) deben operar en forma independiente, con frecuencias compatibles con la rapidez de respuesta requerida en cada caso.

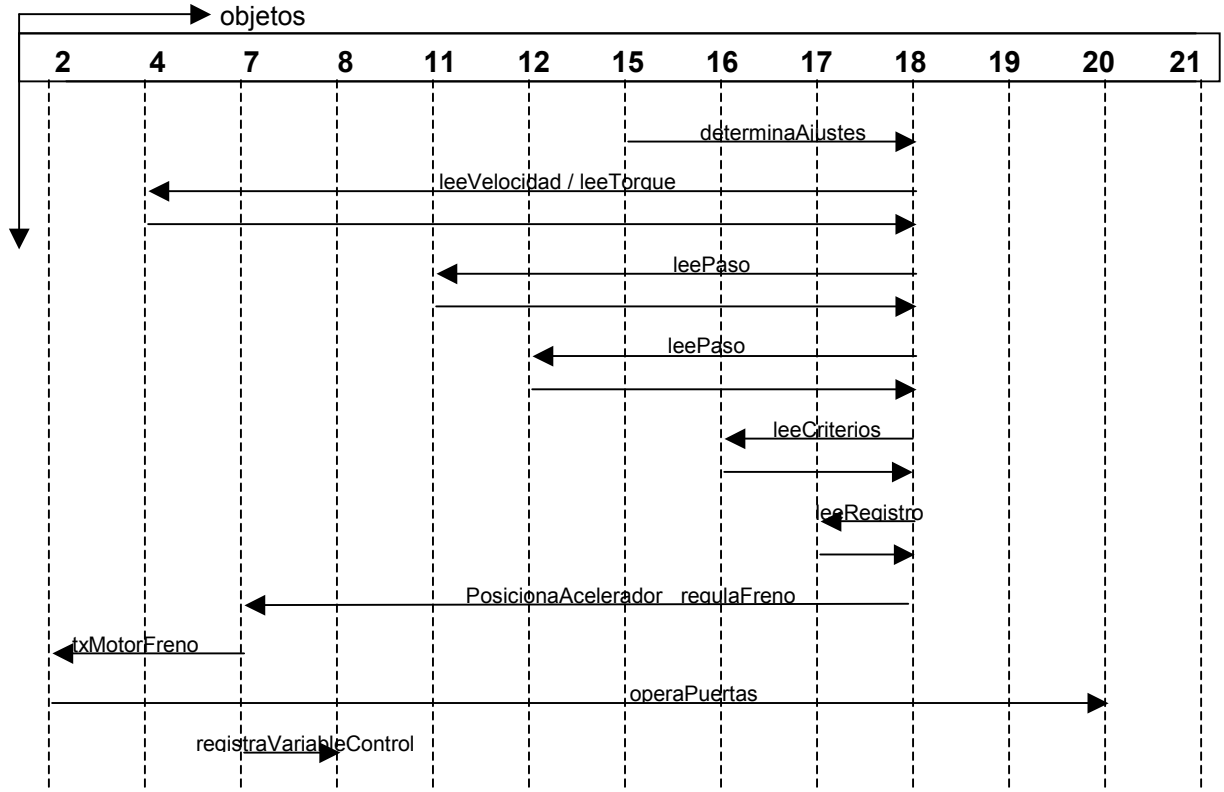
La frecuencia de operación del *ControlPrimario* encuentra un límite en la capacidad de muestreo del taquímetro y sensor de torque, ya que sólo se justifica una acción de control si en el vector de estado se dispone de datos actualizados para las variables controladas. Esto significa que la respuesta del sistema estará supeditada a la frecuencia de operación de estos instrumentos, y siempre existirán valores límites a partir de los cuales la estabilidad del objeto controlado no podrá ser garantizada. Es también importante observar que las constantes de las opciones de realimentación *PID* presentadas en las figuras 7, 8 y 9 (K_p, T_r y T_d) dependen de la frecuencia de operación de la unidad de control, por lo que una vez que sus valores han sido definidos, esta frecuencia no puede ser alterada. En resumen, *ControlPrimario* es un objeto activo que debe siempre operar a una frecuencia preestablecida.

Se comienza por presentar los diagramas que corresponden a la Unidad de Control primario, identificando en cada caso las actividades esenciales cumplidas, que están a cargo de los objetos *SeleccionaLógicaActiva* (19) y *AjustaVariable ControlPID* (18),

Unidad de Control Primario (15): Selecciona lógica de control (evento No. 9)

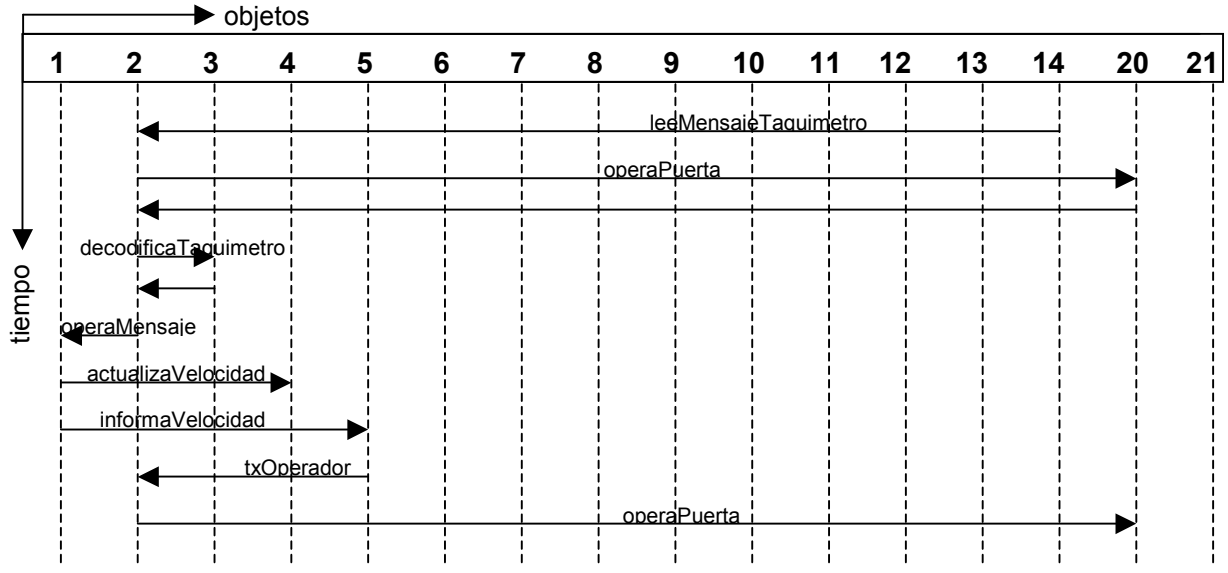


Unidad de Control Primario (15): Opera sobre el conjunto motor freno (evento No. 8)

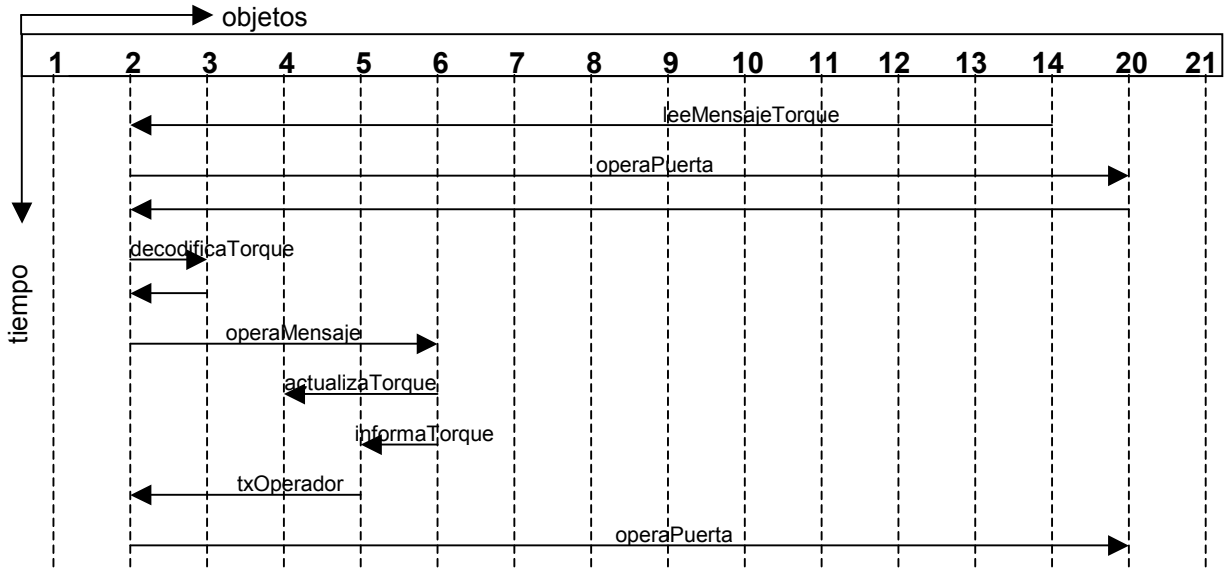


Muy diferente es el caso del objeto *ControlSecundario,* que tiene una responsabilidad activa y otra reactiva. La primera corresponde a la interpretación de los programas de ensayos y operará a muy baja frecuencia, ya que la unidad de tiempo de un paso de ensayo es el segundo. La segunda responsabilidad se refiere a la recepción de mensajes del exterior (*interpretaMensajes*), lo que conduce a una operación reactiva y asincrónica que estará estimulada por las interrupciones originadas en la puerta de comunicaciones. A continuación, se presentan los diagramas de secuencia que corresponden a la unidad de control secundario.

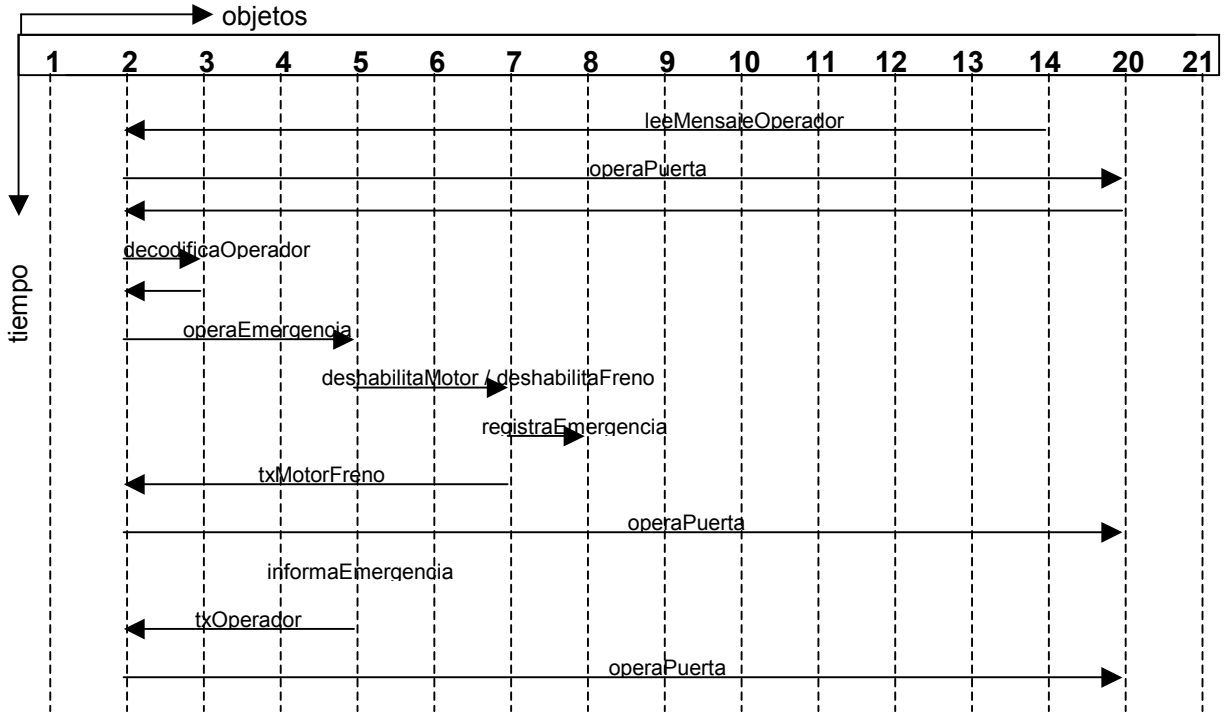
Unidad de Control Secundario (14) : Nuevo valor de velocidad (evento No.1)



Unidad de Control Secundario (14) : Nuevo valor de torque en el freno (evento No.2)

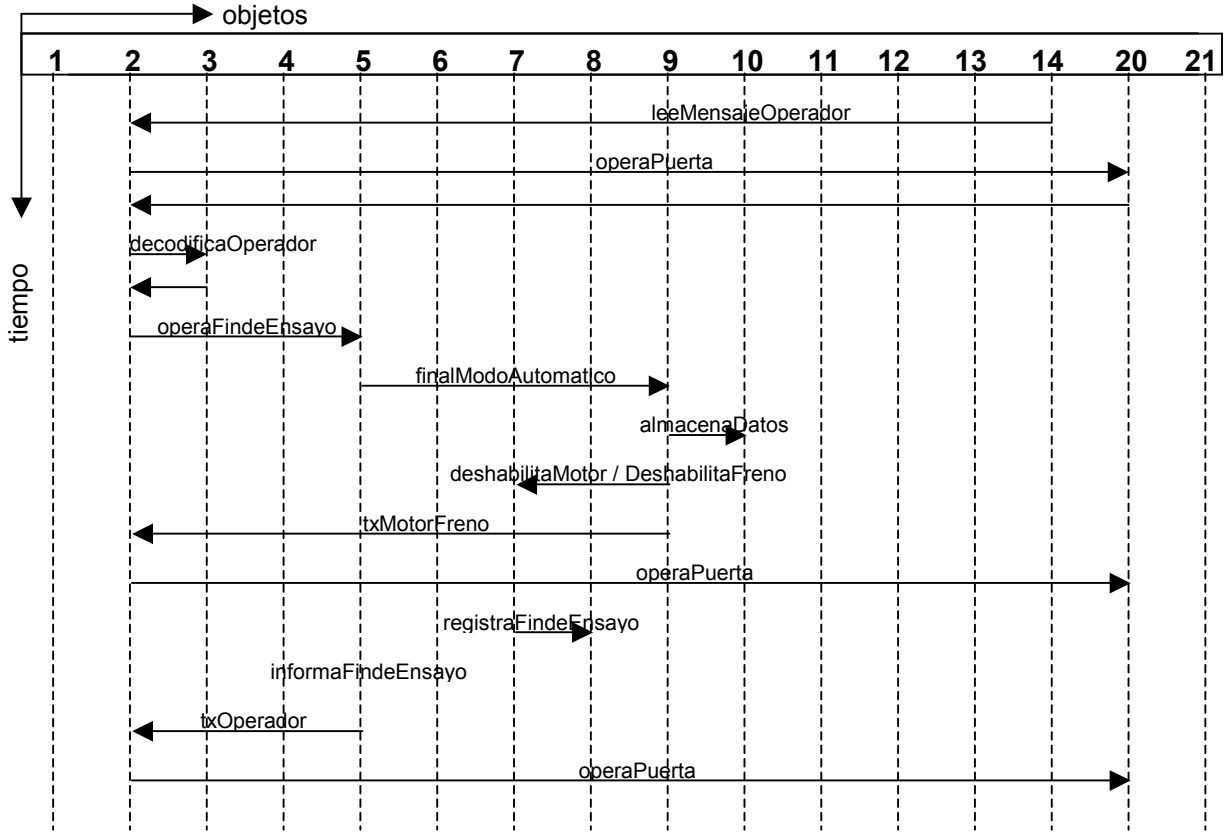


Unidad de Control Secundario (14): Detiene el motor por parada de emergencia (Evento No.3)

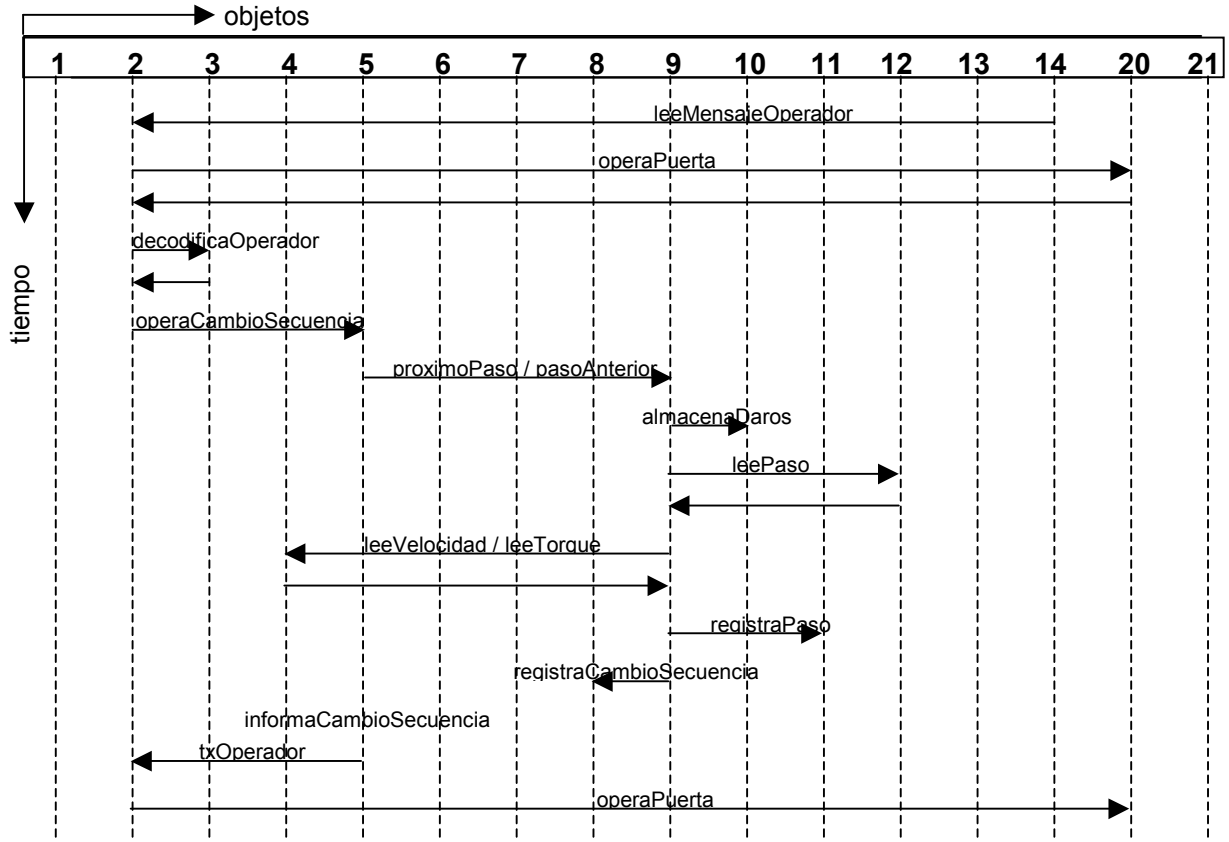


Estos primeros tres diagramas de secuencias corresponden a la recepción de mensajes del taquímetro, sensor de torque y detención del motor por parada de emergencia. Este último mensaje proviene de la interfase con el operador, al igual que los mensajes de detención del motor y cambio de la secuencia de ensayos, que se presentan a continuación. Por último, se representan los diagramas de secuencia correspondientes a la detención del motor por seguros e interpretación del programa de ensayos. Tal como ya fue anticipado, este último objeto cumple una actividad temporizada, verificando regularmente el cumplimiento de la secuencia de ensayo prevista en el programa.

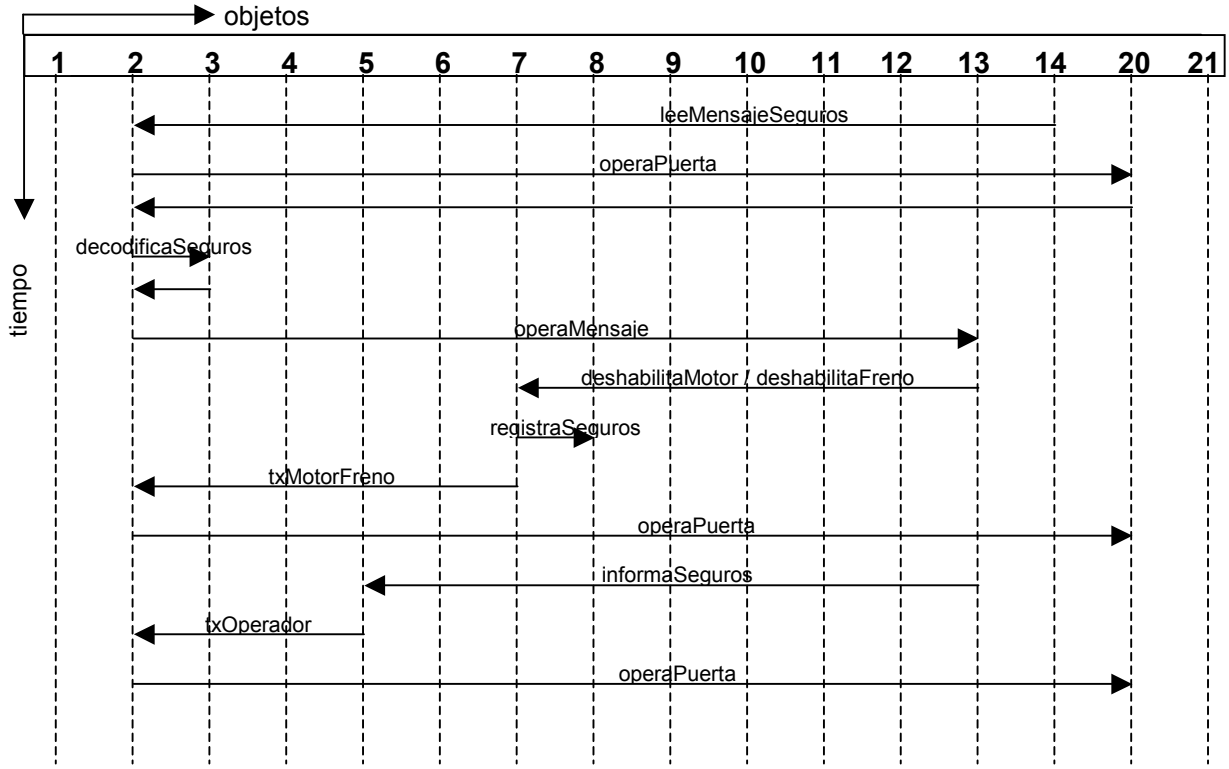
Unidad de Control Secundario (14) : Final de ensayo (evento No.4)



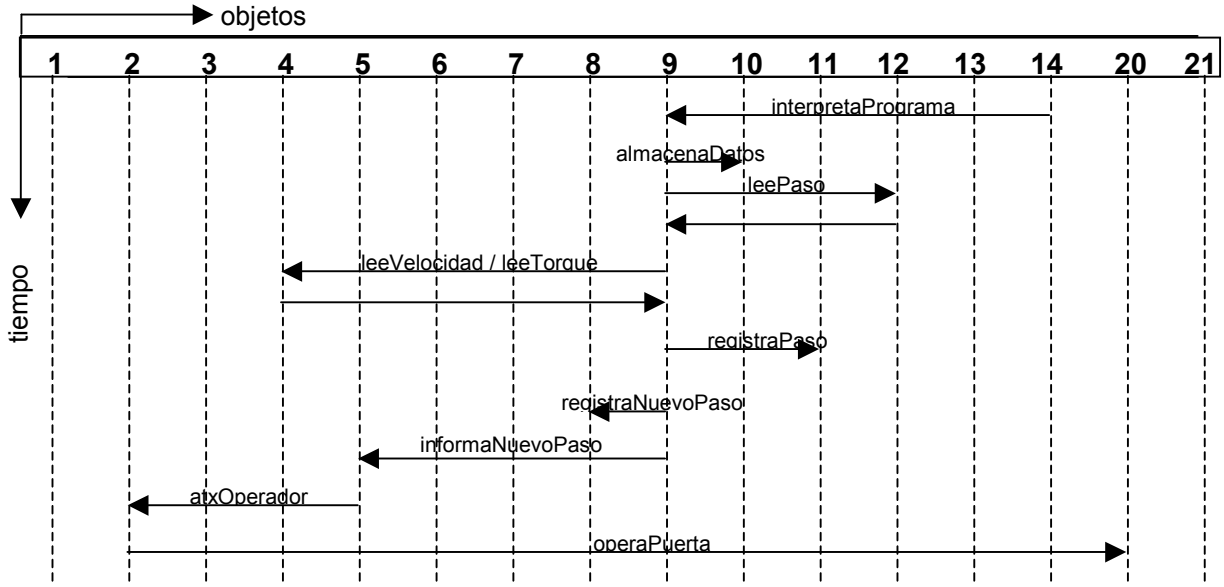
Unidad de Control Secundario (14) : Altera secuencia de ensayo (evento No.5)



Unidad de Control Secundario (14) : Detención del motor por seguros (evento No.6)



Unidad de Control Secundario (14) : Transita a nuevo paso de ensayos (evento No. 7)



En este punto, es conveniente destacar que los patrones de diseño (design patterns), que vienen teniendo una gran difusión en todos los campos de aplicación de la ingeniería de software, también ocupan un lugar cada vez más importante en el diseño intermedio de sistemas de tiempo real. Si bien los sistemas de tiempo real operan sobre una gran diversidad de plataformas, y con finalidades muy diferentes, hay soluciones que son consideradas “clásicas” y con muy pocas variantes son universalmente aceptadas para resolver problemas específicos. En particular, pueden mencionarse los subsistemas de comunicaciones, implementación de lógicas de control e interfases con el usuario como las áreas en las que con mayor

facilidad se adoptan patrones de diseños normalizados. Sin embargo, estas soluciones normalizadas son en su mayoría conceptuales y no hay todavía catálogos de patrones de diseño de uso general (Zalewski, 2002), por lo que al reconocerlas como patrones se está más bien haciendo referencia a enfoques que han demostrado ser efectivos y son considerados como standards "de-facto". Además, un sistema de tiempo real involucra también numerosas operaciones convencionales, que encuentran una fuente de posibles soluciones en los catálogos de patrones de diseño de aplicación general (Gamma et Al, 1995).

5.3 Diseño detallado

En concordancia con los mismos límites ya comentados, se centra el diseño detallado en las unidades de control primario y secundario. Para este tipo de aplicación y este nivel de definición, los *autómatas finitos* representan, en muchos casos, el mejor medio para documentar sus capacidades y particularidades operativas.

5.3.1 Unidad de control secundario

La clase que representa la unidad de control secundario dispone de dos métodos. Éstos, que están destinados a interpretar los programas de ensayos y a recibir los mensajes provenientes del exterior, se describen a continuación.

a. Intérprete de ensayos

El intérprete de ensayos es un módulo activo que interpreta regularmente las sentencias que definen un programa de ensayos, tal como fue presentado en el punto 4.4.8. La actividad cumplida por este módulo queda convenientemente representada con el grafo de la Figura 48 y el significado de cada transición es presentado en la Tabla 9.

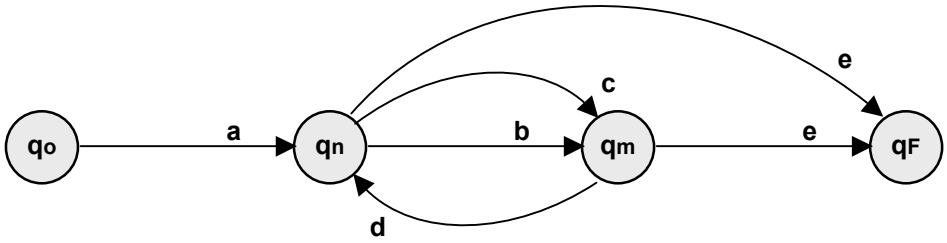


Figura 48: Intérprete de comandos

entrada	comando	evento	Descripción de la función de transición
a	Inicia ensayo	4	$\mathbf{f}(q_o,a)=q_n$; n=1
b	Transición normal a próximo paso	7	$\mathbf{f}(q_n,b)=q_m$; m = n + 1 (próximo paso)
c	Avanza a próximo paso	5	$\mathbf{f}(q_n,c)=q_m$; m = n + 1 (próximo paso)
d	Retrocede a paso anterior	5	$\mathbf{f}(q_m,d)=q_n$; n = m – 1 (paso anterior)
e	Final del ensayo	4	$\mathbf{f}(q_n,e)=q_F$; $\mathbf{f}(q_m,e)=q_F$

Tabla 9: Transiciones en el intérprete de comandos

En las instalaciones para ensayos de motores que están destinadas a pruebas de larga duración se incluyen, en el programa de ensayos, otros tipos de

instrucciones, que están destinadas a permitir ciclos cerrados anidados y otras facilidades. En estos casos, los intérpretes de ensayos quedan representados por *autómatas a pila*, y los programas son definidos por lenguajes Tipo 2 (Chomsky).

b. Recepción de mensajes del exterior

Tal como fue propuesto en el diseño arquitectónico del sistema, la unidad de control secundario está conectada a un bus de comunicación serial RS-485 por el que recibe los mensajes provenientes del taquímetro, sensor de torque, seguros e interfase del operador. Estas unidades son presentadas en el esquema de la Figura 49:

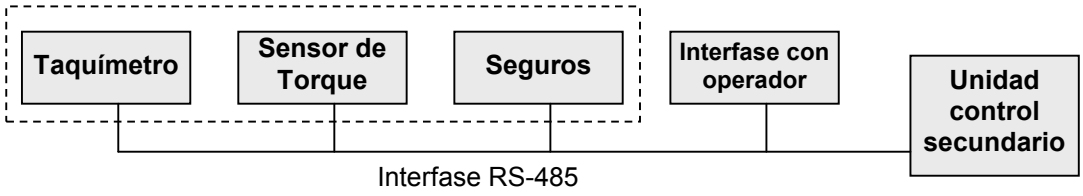


Figura 49: interconexión de bus serial

A los fines de esta comunicación, se adopta un protocolo común, que prevé múltiples comandos para el caso de la interfase con el operador, tal como el presentado en la Tabla 10. Para representar la recepción de estos mensajes se recurre también a un *autómata finito*, que es representado en la Figura 50. La principal función de este autómata es la validación de la integridad y consistencia de los mensajes recibidos. Obsérvese que, con el fin de facilitar la interpretación de las transiciones previstas, los símbolos que pertenecen al alfabeto de entrada del autómata son identificados en el encabezamiento de las columnas de la Tabla 10.

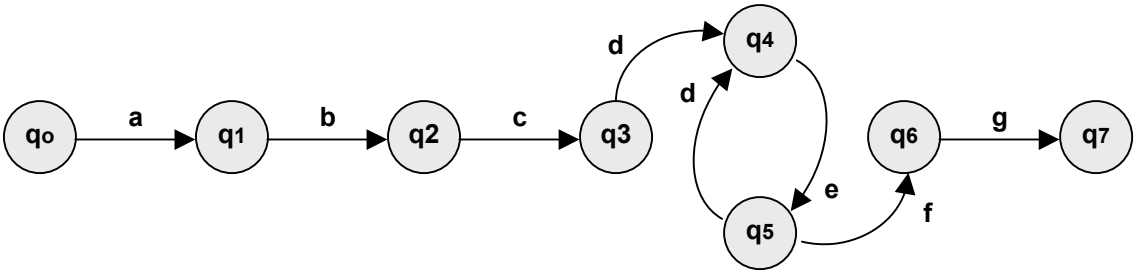


Figura 50: Validación de mensaje

a	b	c	d	e	d	...	f	g
Inicio de mensaje	Origen del mensaje	Cantidad de caracteres	Cdo	Argu-mento	Cdo	....	CRC	Fin de mensaje

Tabla 10: Protocolo de comunicaciones

5.3.2 Unidad de control primario

La unidad de control primario fue definida en la Figura 4 como una unidad conmutada de control. En efecto, se trata de una unidad activa que tiene por finalidad seleccionar la lógica de control más conveniente en cada caso y ajustar regularmente las variables de control.

a. Selección de la lógica activa

La selección de la lógica activa es convenientemente representada por un *autómata finito*, cuyo conjunto de estados queda definido por las opciones de realimentación que están disponibles en cada caso, resumidas en la Tabla 11.

Estas opciones de realimentación ya fueron definidas en los esquemas de las Figuras 33, 34, 36 y 37. Obsérvese que, para implementar estas opciones, se dispone a su vez de los esquemas de realimentación definidos como PID, PI-D y PID-D de las Figuras 7, 8 y 9; por lo que, en caso de considerarlas, la cantidad de combinaciones posibles sería sustancialmente mayor.

Sin embargo, el verdadero problema no es la definición del conjunto de estados de este autómata, sino su función de transición, es decir las condiciones bajo las cuales transitará de un estado a otro, y a las que ya se hizo referencia en las expresiones 24 y 27 al definirse el *control conmutado* en el punto 2.3.1. Algunas de estas transiciones son obvias, y pueden ser anticipadas. Otras, por el contrario, sólo serán descubiertas a partir de un detallado proceso de evaluación del comportamiento de la unidad de control y del objeto controlado. Esto significa que, en la práctica, la función de transición no puede ser completamente definida con anticipación; y por ello es conveniente posibilitar su progresiva actualización. Una alternativa, quizás la más ventajosa, es asociar estas transiciones a los resultados de reglas de producción que son almacenadas en un archivo de configuración, que será leído al iniciarse la operación del sistema. Así, en este archivo, que es denominado *tabla de criterios de control*, bajo la forma de reglas de producción se incorporarán nuevas transiciones a medida que la experiencia lo haga aconsejable.

Estado	Descripción	Variable de Control (VC)	Variable de Proceso (VP)	Condición especial
q_0	Motor descargado a velocidad mínima	Ninguna	Ninguna	Arranque y detención
q_1	Motor libre o con frenado constante	Posición del Acelerador	Velocidad	Tensión de frenado constante
q_2		Posición del acelerador	Torque	
q_3	Motor a carga constante	Tensión en el Freno	Velocidad	Acelerador en posición constante
q_4		Tensión en el Freno	Torque	
q_5	Operación General (Caso 1)	Posición del Acelerador	Torque	Ajuste simultáneo de la tensión de frenado y posición del acelerador
		Tensión en el Freno	Velocidad	
q_6	Operación General (Caso 2)	Posición del Acelerador	Velocidad	
		Tensión en el Freno	Torque	

Tabla 11: Condiciones de Control

En resumen, la selección de la lógica de control está a cargo de un autómata finito (Ec. 21) que es representado en la Figura 51, donde $\mathbf{S} = \{q_o, q_1, q_2, q_3, q_4, q_5, q_6\}$, $S_o = q_o$, $S_F = q_o$ y $\mathbf{\Sigma} = \{a, b, c, d, e, f, g, h, i, j, k, l\}$.

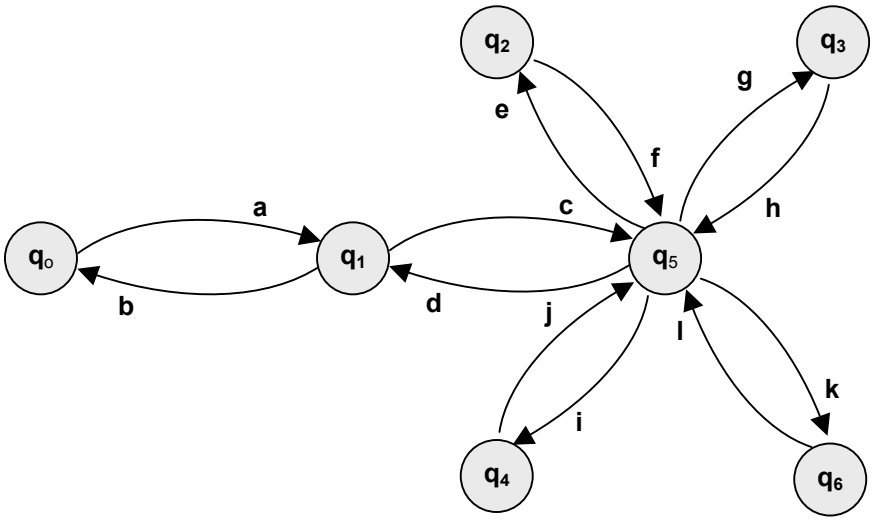


Figura 51: Selección de la lógica de control

Este autómata se comporta como una máquina de Moore, en donde, sus salidas están directamente relacionadas con los estados adoptados en cada instante de tiempo. No es por lo tanto necesario representar estas salidas. Como ya fue expuesto, la función de transición de este autómata queda vinculada a las reglas de producción que son leídas de la tabla de criterios de control y fue anticipada conceptualmente por las expresiones 26 y 27 del punto 2.1.4.

b. Ajuste de las variables de control

Este método debe determinar los errores en cada una de las variables de proceso y luego, a partir de los esquemas de realimentación que correspondan a cada caso (lógica activa), calcular las magnitudes de los ajustes que deben introducirse en las variables de control. Luego, a través de la interfase RS-485, estos ajustes son comunicados a los elementos de accionamiento exterior, que son el actuador de acelerador y el circuito de excitación del freno.

Este método debe disponer de todas las opciones de control y realimentación, para ser aplicadas en cada caso, pero su implementación no ofrece dificultades especiales.

5.4 Programación

Un sistema es el producto final que materializa el objetivo de un proceso de desarrollo de software y su finalidad es cumplir con las especificaciones previstas en sus requerimientos. Para ello, el sistema debe ser escrito en cierto lenguaje, y por lo tanto, es natural que el punto de partida de la fase de programación sea la selección de dicho lenguaje.

Ya fue anticipado que en esta selección inevitablemente influirá la preferencia y experiencia del o de los programadores. Sin embargo, se trata de una decisión técnica que debe apoyarse esencialmente en argumentos objetivos. Para ello, en el punto 3.3.1 de este trabajo se identificaron los atributos que son relevantes en los lenguajes destinados a sistemas de tiempo real.

En este caso, el lenguaje seleccionado fue Delphi, que es una implementación orientada a objetos de Pascal. Las respuestas que ofrece Delphi a los atributos requeridos son las siguientes:

a. Control de tiempo

Se dispone de una resolución de 1 ms en la gestión del tiempo, pudiendo generarse secuencias de eventos que tengan como mínimo este período. El valor indicado es suficiente para los objetivos de este sistema.

b. Concurrencia

Soporta la definición de procesos "multihilo" y la activación periódica de objetos a partir de eventos temporales, brindando el soporte necesario para procesos concurrentes.

c. Seguridad

Se dispone de herramientas de "debugging" y directivas en el compilador que permiten un control minucioso del código. En caso de producirse un error, el programa ejecutable brinda referencias precisas sobre su causa, por lo que se facilita la obtención de productos finales sólidos y confiables.

d. Portabilidad

Delphi genera programas ejecutables completamente autocontenidos, lo que facilita la portabilidad del sistema dentro del entorno de plataformas operativas Windows, sin necesidad de librerías externas para su posterior operación. Además, la portabilidad a otras plataformas está asegurada ya que, para los sistemas UNIX y LINUX, existe una versión de Delphi denominada Kylix. Por último, hay también una versión multiplataforma y completamente libre de Delphi llamado Lazarus que tiene su origen en la comunidad del software libre. El nombre Lazarus proviene del Oráculo de Delfos.

e. Eficiencia

Delphi, al igual que Pascal, es un lenguaje fuertemente tipeado, y esto contribuye a su rapidez en tiempo de ejecución. Además, se trata de un compilador y *link-editor* que disponen de diversas opciones de optimización, lo que permite generar programas ejecutables muy eficientes.

Adicionalmente, pueden mencionarse otros atributos que no están asociados a la operatividad en tiempo real de los sistemas, pero que sí están relacionados con la metodología de diseño propuesta. Éstos son:

f. Orientación a objetos

Delphi es un lenguaje claramente orientado a objetos, con un buen manejo de clases, herencia y polimorfismos. Cubre todas las necesidades de un desarrollo complejo, bajo una rigurosa implementación del paradigma de objetos.

g. Vigencia y compatibilidad

Las recientes versiones de Delphi demuestran su vigencia, y en todos los casos, puede comprobarse una excelente compatibilidad con versiones anteriores.

5.5 Implementación

Para poder operar el simulador del sistema de control de ensayos de motores, es indispensable la implementación de todos los módulos que ya fueron

representados en la Figura 24 y descriptos en detalle en el punto 4.4. La implementación de estos módulos presenta los siguientes aspectos particulares:

a. Unidad de Control

Tanto en la especificación de requerimientos como en la fase de diseño, se puso el foco en los principales componentes de esta Unidad, que son los de control primario y secundario. Su implementación contempla los aspectos particulares ya definidos en estas fases de especificación y diseño.

b. Conjunto motor-freno

El comportamiento del motor y freno es representado por redes neuronales, y para acoplar ambos elementos, se opta en esta oportunidad por un eje rígido. Por lo tanto, la respuesta del conjunto se obtiene integrando la ecuación 83 del punto 4.4.3.b.

c. Taquímetro

El taquímetro se activa periódicamente, según lo enunciado en el punto 4.4.4, y tiene a su cargo la conversión de la señal de velocidad, tal como fue detallado en el punto 4.5.2.

d. Sensor de Torque

El sensor de torque se activa también periódicamente, según ya fue detallado en el punto 4.4.5. A los fines de concretar las lecturas del torque del freno se debe convertir su señal según se describe en el punto 4.5.1.

Por otra parte, ya fue establecida la imposibilidad de medir el torque en el motor y por esta razón su valor es determinado a partir de la ecuación 89. Para ello, se calcula la aceleración del motor mediante la ecuación 72.

e. Seguros

La función de la unidad de seguros fue descripta en el punto 4.4.6 y sus señales son convertidas según lo previsto en 4.5.3.

f. Interfase con el operador

Sus características fueron enunciadas en el punto 4.4.9. Por tratarse de una interfase de simulación, se ha previsto una representación gráfica central sobre un plano cartesiano ortogonal y diversas opciones que están destinadas a desplegar ventanas que detallan el estado de los diferentes procesos numéricos (Anexo A).

6 Evaluación del simulador

Tal como ya fue señalado en la consideración del marco metodológico, el desarrollo de un sistema debe contemplar una serie planificada de pruebas. Su objeto es comprobar que, bajo todas las condiciones posibles de operación, cada módulo cumplirá adecuadamente con sus responsabilidades y que, el producto final, será capaz de exhibir un correcto funcionamiento (punto 3.4 y Figura 20). Si bien este control progresivo es siempre recomendable, reviste particular importancia en aquellos sistemas que incluyen la implementación de algoritmos complejos.

Debe observarse que, a pesar de que este documento ha centrado su atención en el diseño de la unidad de control, han debido también diseñarse e implementarse los demás módulos de software, que son necesarios para simular el comportamiento del conjunto motor-freno y los elementos de adquisición de datos. Todos estos componentes principales ya fueron identificados en el punto 4.4 de la fase de definición de requerimientos, su interconexión se presentó en el esquema de arquitectura del sistema en la Figura 47 y el alcance de la implementación de cada uno fue resumido en el punto 5.

La secuencia ordenada de pruebas previstas, para la verificación del sistema en su conjunto y de sus componentes, son las siguientes:

a) Testeo del algoritmo de integración de las ecuaciones del movimiento
b) Testeo de los algoritmos de extrapolación y derivación
c) Testeo de las funciones de aproximación del comportamiento del motor y freno
d) Evaluación de la simulación de la respuesta dinámica del conjunto motor-freno en ausencia de acciones de control.
e) Evaluación de la estimación del torque del motor a partir del torque medido en el freno.
f) Evaluación de la interacción del conjunto motor-freno y la unidad de control.

Con el fin de disponer de valores de referencia para el cálculo de errores, en las evaluaciones previstas en los puntos “a”, “b” y “c” se utilizarán expresiones matemáticas conocidas y, que tengan un comportamiento similar al esperado en los elementos reales. Debido la disponibilidad de valores de comparación, se otorga gran importancia a la exhaustiva evaluación de estos procesos que pueden ser contrastados con soluciones matemáticas exactas. Por el contrario, la posterior evaluación del desempeño del sistema en su conjunto encerrará muchas dificultades por la poca disponibilidad de resultados. En efecto, sólo para algunas condiciones de operación se dispondrá de valores experimentales.

Para cada una de las pruebas previstas en la secuencia de verificación, se ha hecho una breve selección de resultados, que son presentados a continuación junto con una descripción de las expresiones o valores tomados como referencia.

6.1 Integración de las ecuaciones diferenciales del movimiento

Para la evaluación del desempeño de los algoritmos de integración de las ecuaciones del movimiento, y la selección final del método más conveniente, se implementó una expresión que contiene dos funciones armónicas, de distinta frecuencia, y donde una de ellas tiene una amplitud variable, tal como la siguiente:

$$y = A.e^{-B.t} . \cos(\alpha.t) + D.\text{sen}(\beta.t) \quad (103)$$

donde

$$y' = - B.A.e^{-B.t} . \cos(\alpha.t) - \alpha.A.e^{-B.t} . \text{sen}(\alpha.t) + \beta.D.\cos(\beta.t) \quad (104)$$

es decir que

$$y' = f(y, t) = - B.y - \alpha.A.e^{-B.t} . \text{sen}(\alpha.t) + D.(\beta.\cos(\beta.t) - \text{sen}(\beta.t)) \quad (105)$$

con las siguientes constantes características:

T_1 = período de la armónica principal

T_2 = período de la armónica secundaria

α = frecuencia de la armónica principal ($\alpha = 2.\pi / T_1$)

β = frecuencia de la armónica secundaria ($\beta = 2.\pi / T_2$)

A = amplitud de la armónica principal

B = amortiguamiento

D = amplitud de la armónica secundaria

En la Figura 52, se representa la función de comparación propuesta. Esta es evaluada en un intervalo de 30 segundos y se representan los valores que corresponden a dos períodos diferentes de la armónica secundaria.

En este gráfico están superpuestas las curvas de la función original (Ec.103) y las que corresponden a los resultados de integrar su derivada (Ec.105) por el método de Runge-Kutta de 4º orden (Ec. 36 y 37). La escala del grafico no permite apreciar el error, que en este caso estuvo siempre por debajo del 0,5%. Además, al evaluarse la función en períodos prolongados de más de 10 minutos, se pudo comprobar el buen desempeño del método de integración, ya que el error siempre se mantuvo acotado por debajo del valor indicado.

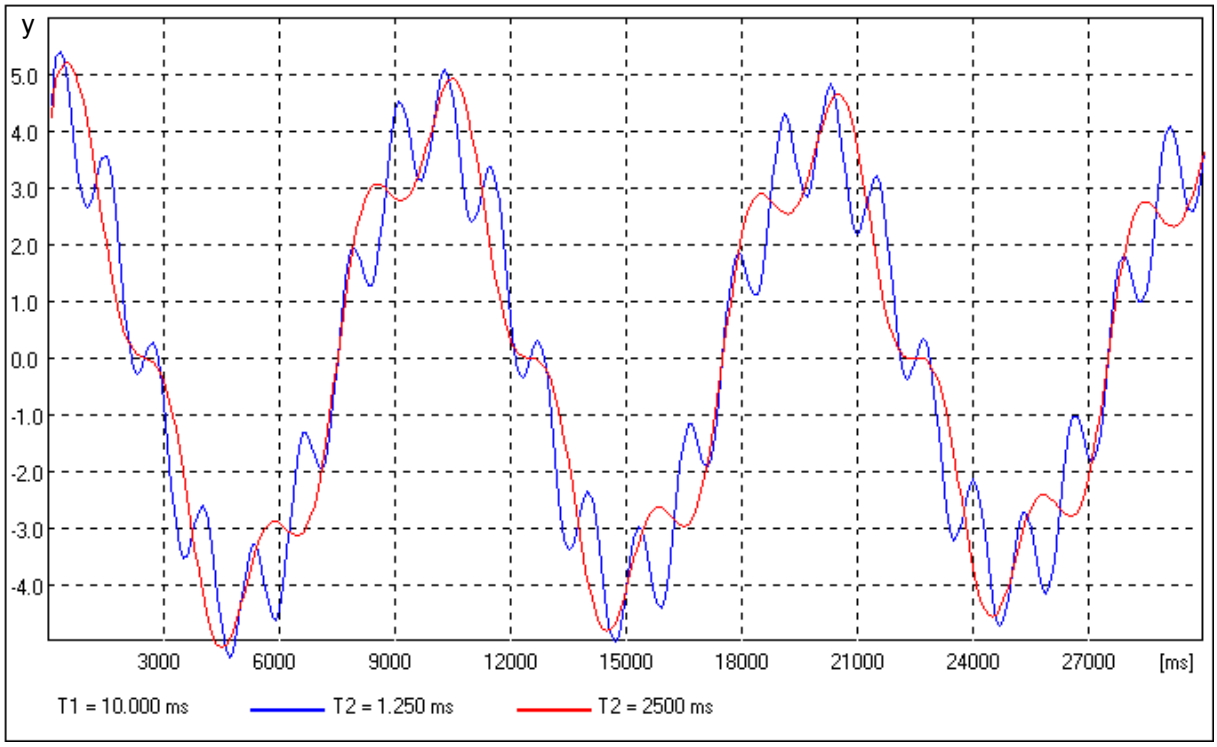


Figura 52: Función de evaluación de la integración numérica

También se evaluó el comportamiento de estos métodos de integración numérica con diversas relaciones entre los períodos de las dos armónicas. En la Figura 53, se resumen los resultados obtenidos con diferentes intervalos de integración ($T_3 = \Delta t$) y distintas relaciones entre los períodos T_1 y T_2.

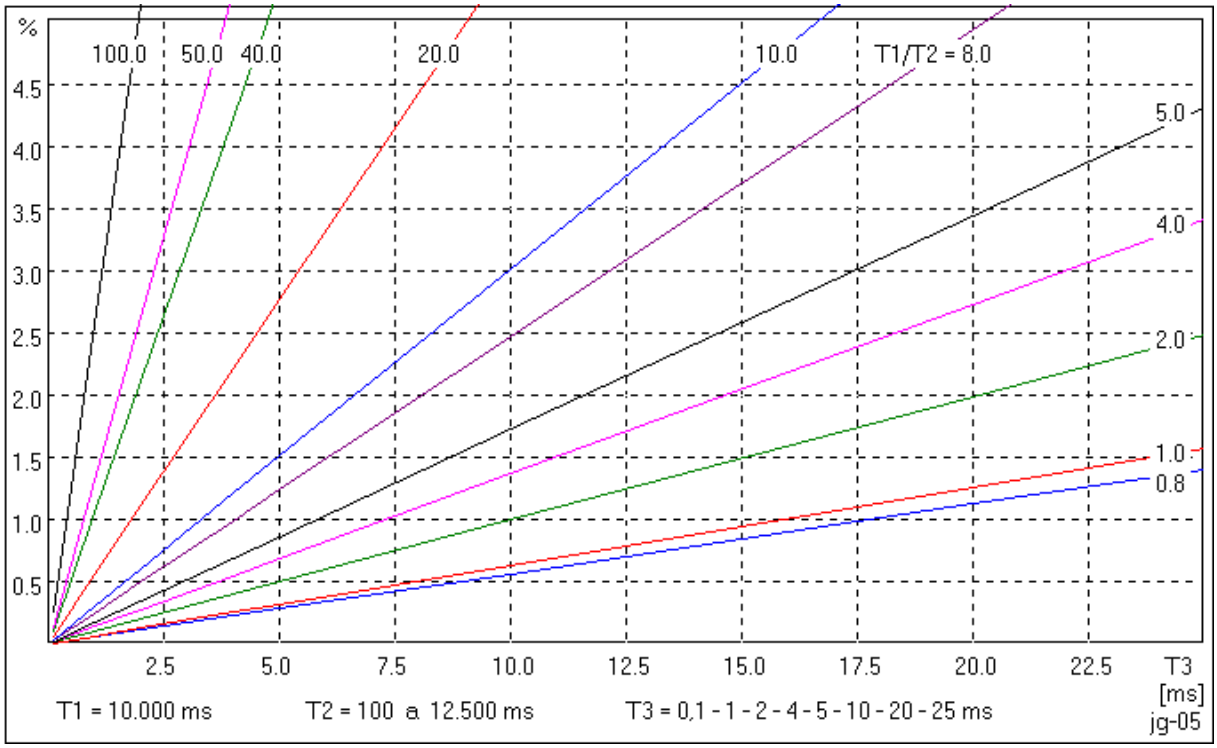


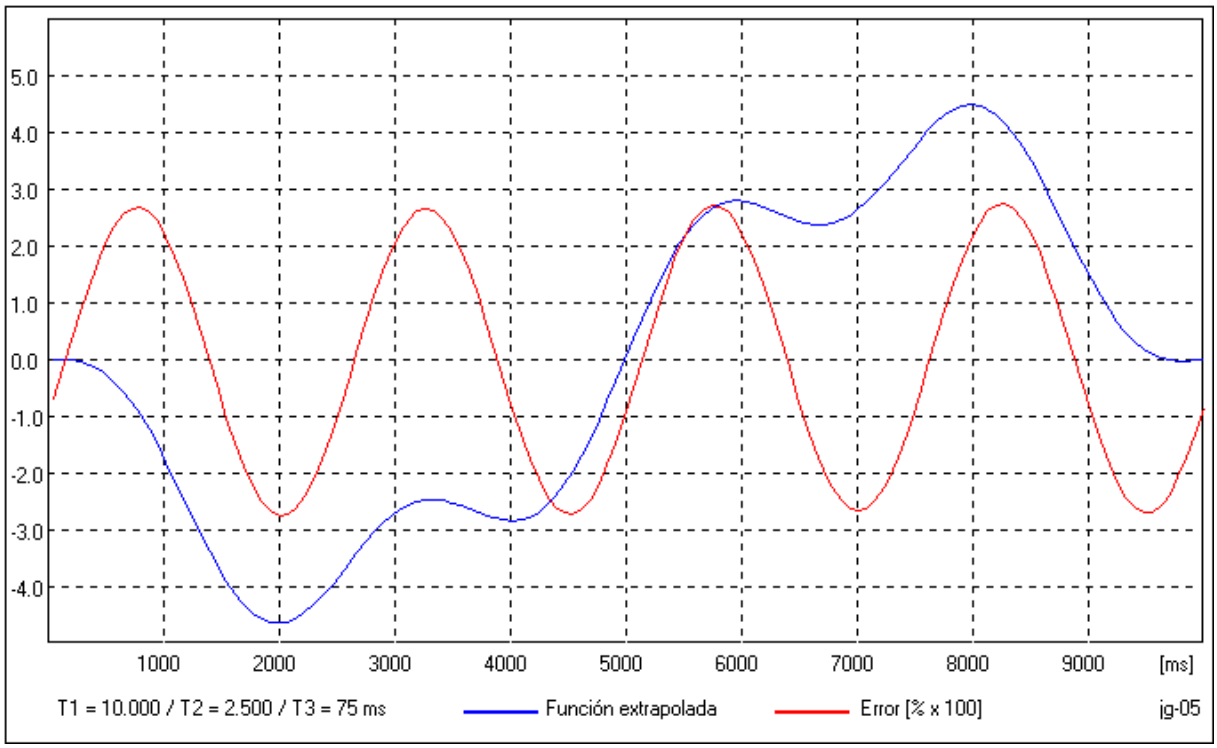
Figura 53: Evolución del error con el intervalo de integración T_3

En este gráfico puede observarse que, cuando la relación entre los períodos T_1/T_2 aumenta, el error depende significativamente del intervalo de integración. No obstante, los intervalos que son propios de los procesos de tiempo real están en el orden de 1 o 2 milisegundos; por lo que los errores obtenidos con estos intervalos son satisfactorios en todos los casos. Obsérvese el comportamiento claramente lineal que manifiestan estas curvas de error para todas las relaciones T_1/T_2 .

En todas los evaluaciones realizadas se compararon los resultados obtenidos con los métodos de Runge-Kutta de 4º orden (Ec. 36 y 37), Runge-Kutta-Butcher de 5º orden (Ec.38) y el método de pasos múltiples de Milne, obteniéndose siempre resultados prácticamente idénticos. Esta circunstancia permitió prescindir del método de paso variable de Runge-Kutta_Fehlberg (Ec. 41), lo que fue muy ventajoso por las dificultades que encierra la implementación de los métodos de paso variable en los entornos de tiempo real. Finalmente, para esta implementación se seleccionó el método de Runge-Kutta_Butcher.

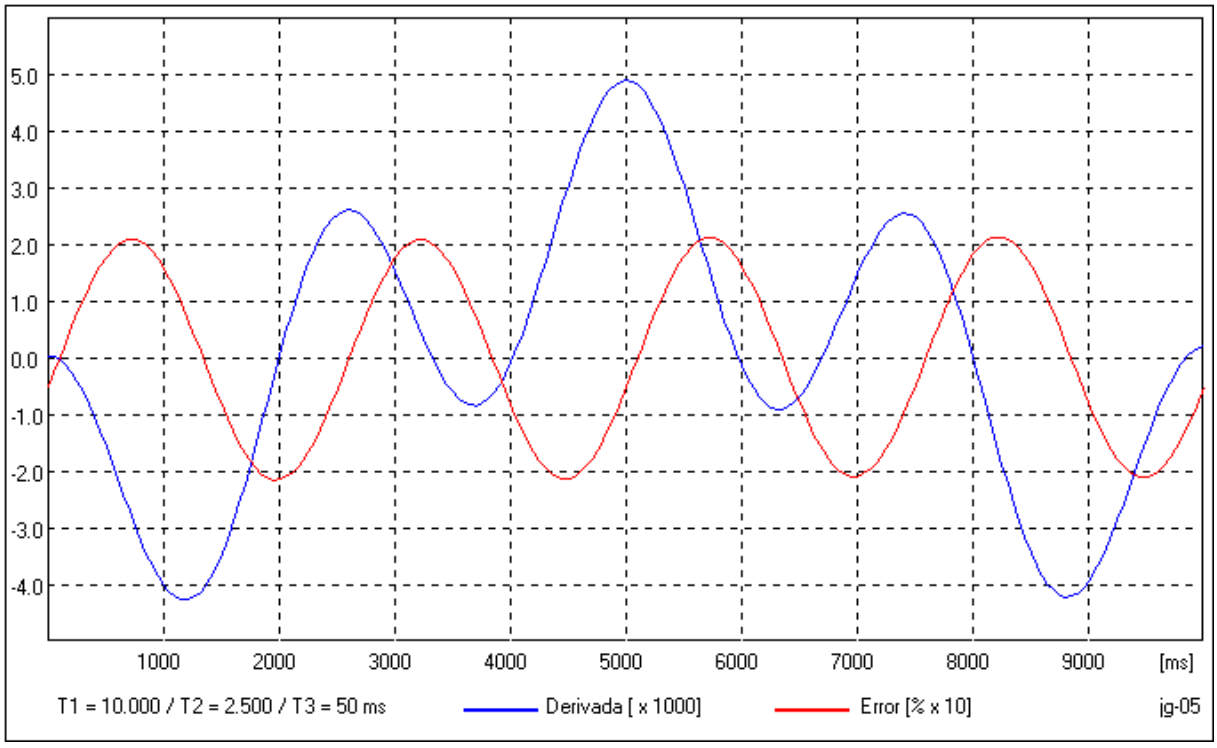
6.2 Extrapolación y derivación en puntos extrapolados

Debe recordarse que, la necesidad de conocer en un mismo instante de tiempo los valores provenientes de diferentes fuentes, como son el taquímetro y el sensor de torque, hace necesaria la extrapolación de estas funciones en el entorno del último punto conocido. Este entorno no será nunca mayor que el intervalo con que estos valores son muestreados, por lo que para la evaluación del algoritmo se toma como referencia el caso representado en la Figura 15. Aquí se utilizan cuatro puntos conocidos y el valor es extrapolado a una distancia Δt del último punto que es igual al intervalo de muestreo, lo que significa que $\delta = 1$.



Figuras 54: Extrapolación de un punto de la función

Esta extrapolación fue implementada por el método de los mínimos cuadrados con un polinomio de tercer grado (Ec.64); y para evaluar los resultados se utilizó la misma función que fue empleada con la integración numérica (Ec.103).



Figuras 55: Derivación numérica de la función extrapolada

La derivación de la función en el punto extrapolado fue también realizada por mínimos cuadrados (Ec.72) y algunos de los resultados obtenidos en intervalos de 10 segundos son representados en las Figuras 54 y 55.

La Figura 54 presenta superpuestas la función de referencia y los valores extrapolados en un intervalo de 10 segundos. Como ya ocurrió con anterioridad, la magnitud de los errores son lo suficientemente pequeños como para que no sea posible distinguir entre ambas curvas. En el mismo gráfico se presenta la curva del error relativo que fue multiplicado por cien para posibilitar su visualización. Puede observarse que los mayores errores están en la proximidad de los valores extremos de la función.

Por su parte, la Figura 55 representa las curvas de la derivada de la función y la de su error relativo, que aquí fueron respectivamente multiplicadas por mil y por diez para posibilitar su visualización. En este caso, los mayores valores de la curva del error relativo están en proximidad de los puntos de inflexión de la curva que representa la derivada.

Un aspecto muy importante que debe ser considerado es el comportamiento del error cuando aumenta el intervalo de tiempo entre los valores conocidos Δt y la distancia al valor extrapolado. Para ello, se hicieron varios estudios que dieron lugar a gráficos como el representado en la Figura 56, que corresponde al caso en que la relación entre los períodos de ambas funciones es $T_1/T_2 = 4$. Obsérvese que el mayor error en la derivada es aquí menor a 0,1% para un Δt menor a 40 ms, y que el mayor error en la evaluación de la función es aproximadamente unas cincuenta veces más chico. En un caso extremo en que el período del muestreo fuese de 100 ms (diez valores por segundo), el error relativo en la derivada sería del orden de 1,7 % y de 0,085 % en la función extrapolada.

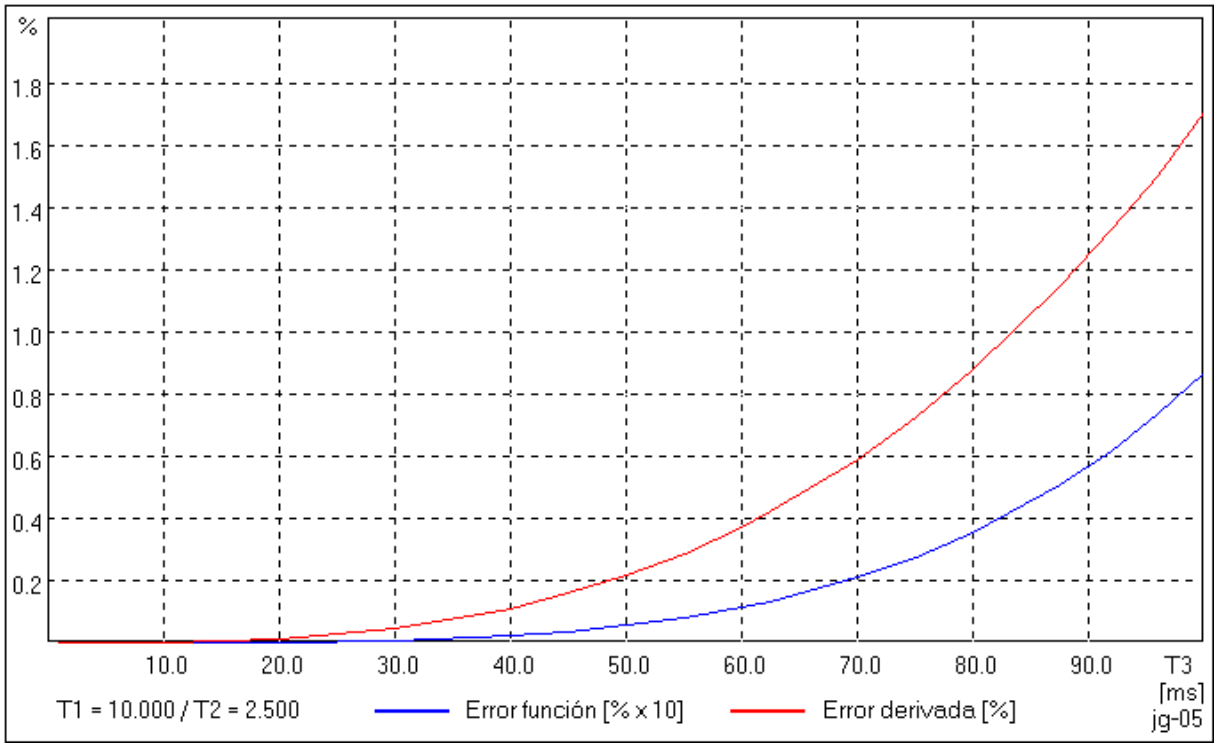


Figura 56: Comportamiento de los errores con diferentes intervalos entre los puntos conocidos

A partir de los estudios realizados, se concluye que la extrapolación y la derivación por el método de los mínimos cuadrados permite obtener muy buenas aproximaciones. Por ser los errores suficientemente bajos, se considera que el método es adecuado para ser implementado en este trabajo.

6.3 Aproximación de las funciones del motor y freno

Las redes neuronales artificiales serán aquí empleadas para aproximar el comportamiento del motor y del freno dinamométrico. Para ello, en cada caso debe seleccionarse una arquitectura conveniente. A su vez, para determinar los valores de sus parámetros, debe someterse luego cada red a un proceso de entrenamiento.

Como ocurrió con los casos anteriores, se utilizan expresiones matemáticas conocidas para evaluar el comportamiento de las funciones de aproximación a ser adoptadas. En este caso, en que la arquitectura más apropiada para cada función debe obtenerse a partir de sucesivas pruebas, la utilización de funciones conocidas para el proceso de entrenamiento es altamente recomendable. Con este fin, se adoptan las funciones siguientes para representar al motor y al freno:

$$T_m(\omega, a) = (A_0 + A_1*\omega + A_2*\omega^2 + A_3*\omega^3 + A_4*\omega^4)*(1 - A_m\, a) \qquad (106)$$

$$T_r(\omega, v) = (B_0 + B_1*\omega + B_2*\omega^2 + B_3*\omega^3 + B_4*\omega^4)*(1 - B_r\, v) \qquad (107)$$

donde $A_0 = 0.4$, $A_1 = 1.1$, $A_2 = -0.8$, $A_3 = 0.0$, $A_4 = 0.0$, $A_m = 0.8$
y $B_0 = 0.1$, $B_1 = 0.1$, $B_2 = 0.2$, $B_3 = 0.0$, $B_4 = 0.6$, $B_r = 0.5$

Por tener dos argumentos, las funciones anteriores quedan representadas por superficies sobre ejes cartesianos. Para facilitarse su visualización, en la Figura 57 son presentadas en perspectiva.

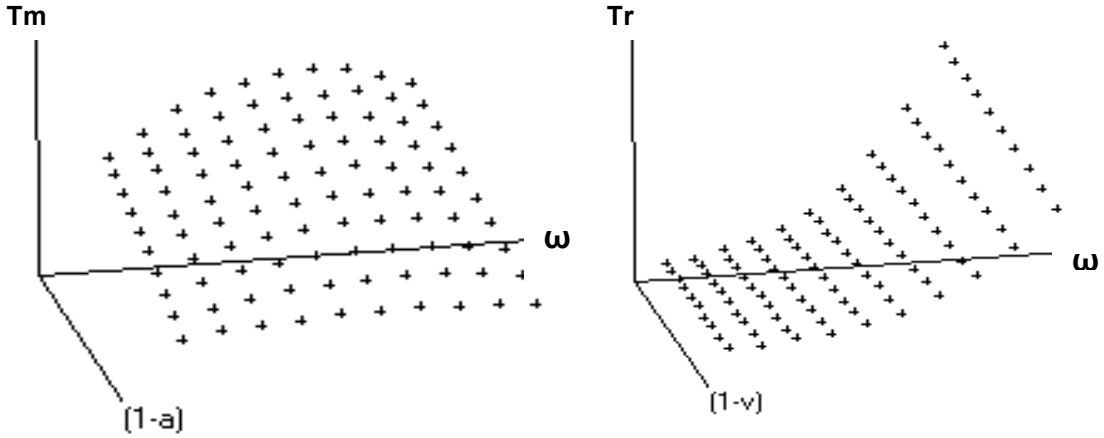


Figura 57: Curvas teóricas del motor y freno

El proceso de selección de la arquitectura más conveniente para cada red es laborioso, ya que pueden utilizarse varias capas ocultas y cada capa puede a su vez contener un número de células que debe ser determinado. Para estos casos se adoptaron redes con dos capas ocultas, tales como la representada en la Figura 10, realizándose los procesos de entrenamiento en las siguientes condiciones:

a) En razón de las características de las funciones a ser aproximadas, las redes tenían dos unidades en la capa de entrada y una en la de salida. Se trabajó con valores de entrenamiento normalizados en el rango 0.0 a 1.0.

b) Se exploraron diversas combinaciones, con tres a quince unidades en cada capa oculta. Cada caso fue identificado por un código que incluye las unidades de cada capa separadas por guiones, tal como 2-N-M-1.

c) El entrenamiento fue realizado con el método de *backpropagation*, que fue definido en el punto 2.3.3.

d) Se adoptó la técnica denominada *Adaptive Back Propagation* (Ec. 63) definida en el punto 2.3.5, que ajusta el factor de aprendizaje **η** durante el proceso de entrenamiento. Se evitó así la necesidad de hacer pruebas con diferentes valores para encontrar el factor más conveniente.

e) Las pruebas realizadas demostraron la conveniencia de utilizar un factor de momentum **β**, con valores superiores a 0,80.

f) Todos los procesos de entrenamiento fueron realizados a partir del mismo conjunto de pesos iniciales, que fueron generados en forma aleatoria.

g) Con las ecuaciones 106 y 107 se definieron conjuntos de pares de valores de entrenamiento y conjuntos de evaluación. Estos últimos están destinados a comprobar la capacidad de generalización de las diferentes configuraciones.

h) Los conjuntos de entrenamiento y evaluación tuvieron entre 80 y 120 puntos, tanto en el caso del motor como del freno.

i) Durante los procesos de entrenamiento, se evaluaron tanto los errores obtenidos con el conjunto de entrenamiento como con el conjunto de evaluación.

En la Figura 58, se representa la evolución del error durante el proceso de entrenamiento de redes con diferentes configuraciones. Los datos de entrenamiento son de la función del motor y corresponden a 100 puntos distribuidos regularmente sobre el plano X-Y. Se utilizó un coeficiente de momentum **β** = 0,95 y el coeficiente de aprendizaje fue variable a partir de un valor mínimo **η** = 0.08.

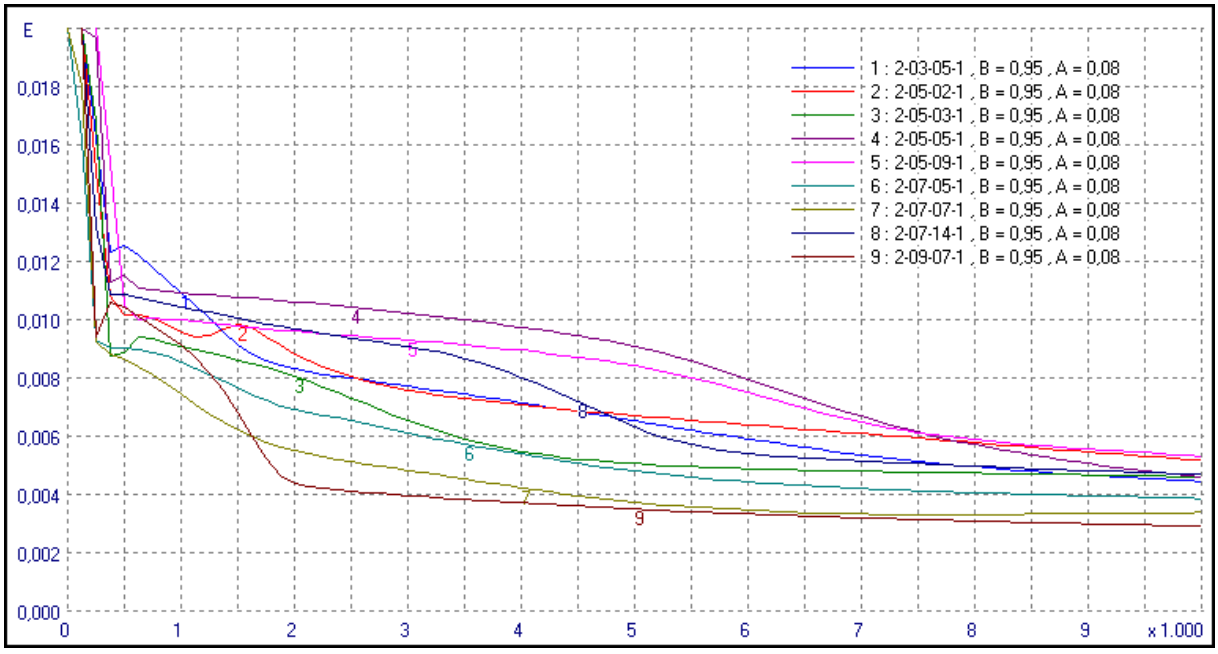


Figura 58: Evolución del error en el proceso de entrenamiento de redes con diferentes arquitecturas

En la Figura 59, se presenta un proceso similar, sólo que en este caso destinado a explorar el factor de momentum **β** más conveniente para una red en particular. Por ello, se trabaja siempre con la misma configuración y un valor de **η** = 0.10. Los datos de entrenamiento corresponden también a la función del motor.

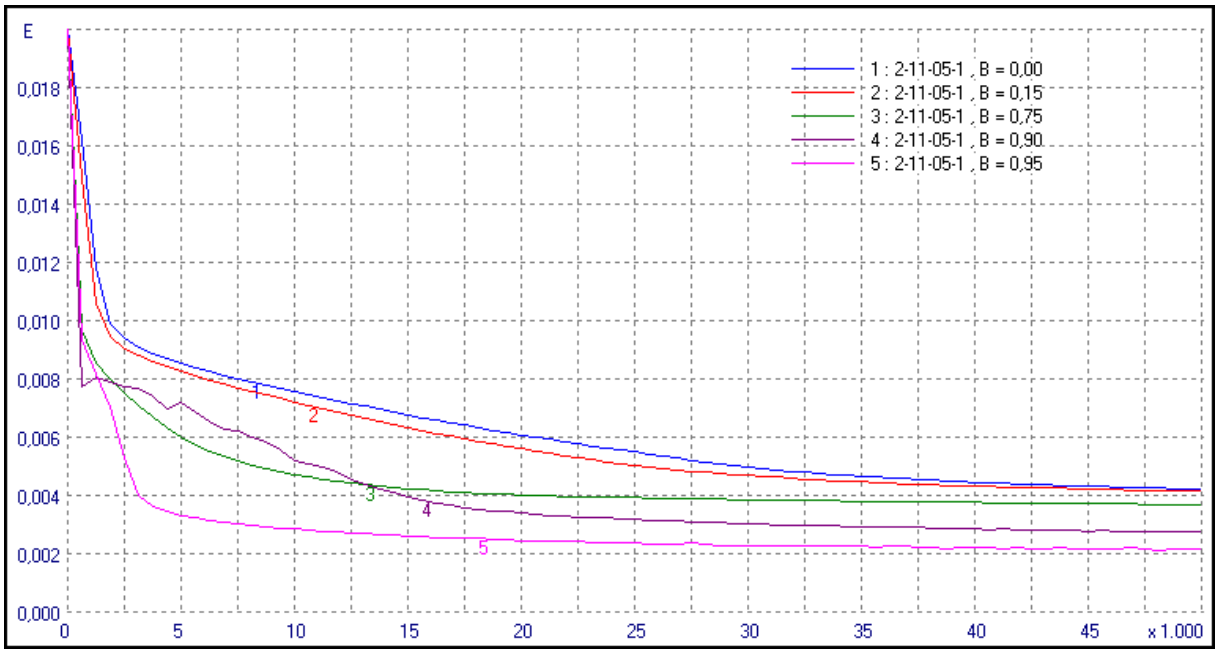


Figura 59: Evolución del error en el proceso de entrenamiento con diferentes valores de β

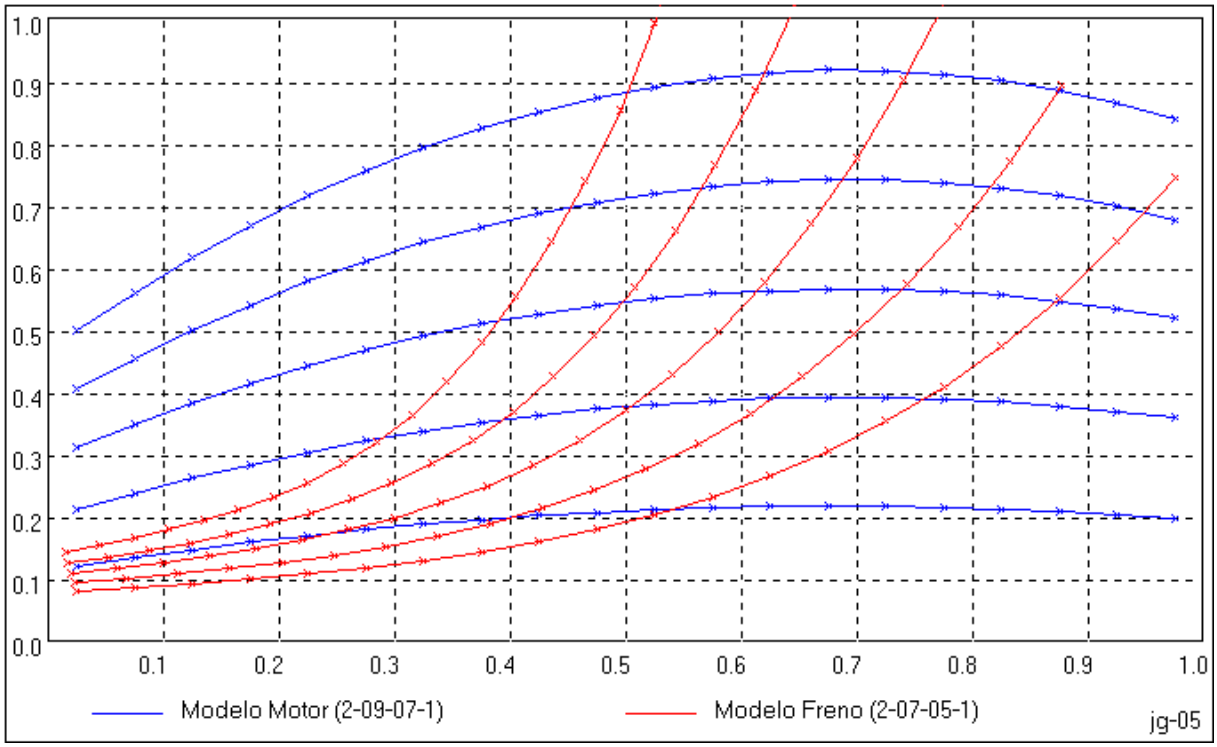


Figura 60: Reproducción por redes neuronales de las curvas del motor y freno

Como resultado de todas las pruebas realizadas, se concluyó que las redes más convenientes tienen las configuraciones presentadas en la siguiente tabla:

Representación	Unidades el cada capa			
	Entrada	Oculta 1	Oculta 2	Salida
Motor	2	9	7	1
Freno	2	7	5	1

En la Figura 60, se representan las curvas del motor y freno correspondientes a cinco condiciones de acelerador y tensión de frenado, obtenidas con estas redes. Al igual que en otros casos anteriores, la escala del gráfico no permite distinguir los puntos obtenidos con las expresiones matemáticas de aquellos calculados con los modelos neuronales.

6.4 Evaluación del comportamiento del conjunto motor-freno

Para estas primeras evaluaciones del proceso de simulación de la respuesta dinámica del conjunto Motor-Freno, se opta por un eje rígido. Por lo tanto, el modelo dinámico queda representado por la ecuación 83 del punto 4.4.3.b, que debe ser integrada en el tiempo.

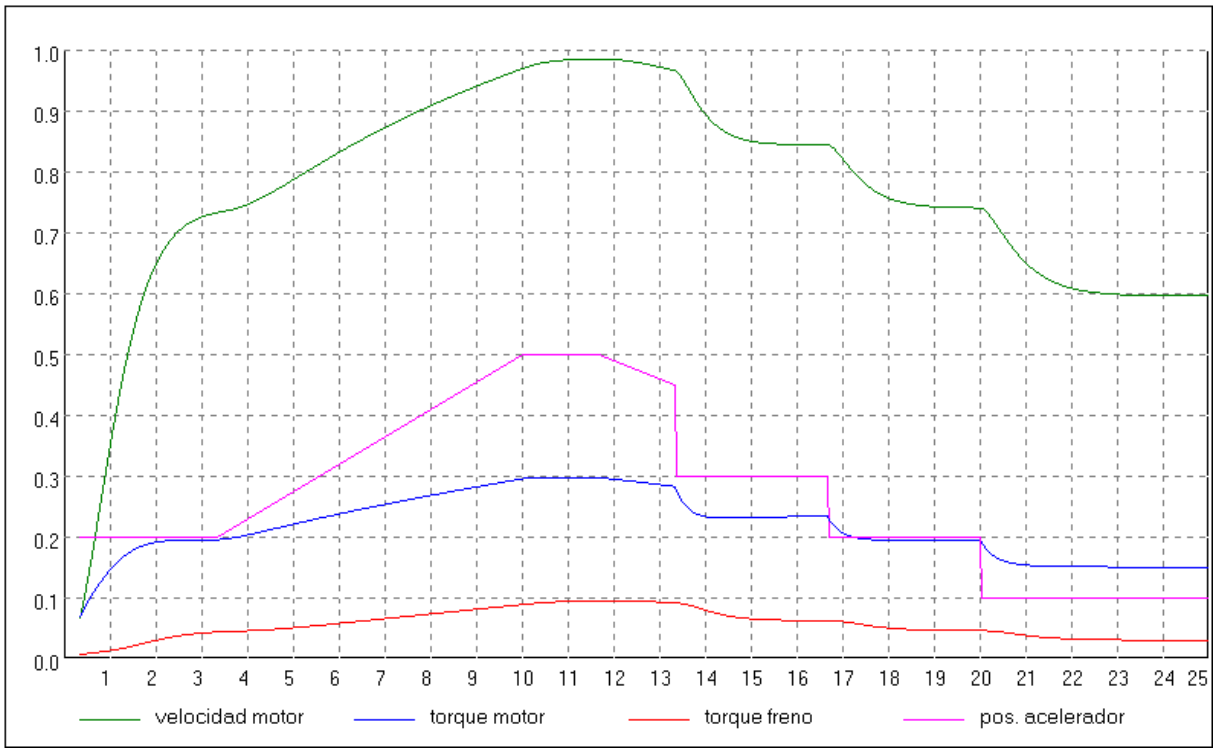


Figura 61: Respuesta del conjunto motor-freno a un cambio de posición del acelerador

Para estudiar el comportamiento de este proceso numérico, se hicieron numerosas pruebas, resultando todas ellas satisfactorias. En la Figura 61, se reproducen los resultados de una de estas evaluaciones, que corresponde en este caso a una aceleración que combina un escalón y una rampa para llevar el motor a su máxima velocidad. Luego, el acelerador desciende una rampa y tres escalones sucesivos, dejando el motor al 60% del valor anterior. En estos casos se trabajó con la constante $T_1 = 0$ (Figura 26). En este gráfico se representan: la posición del

acelerador, el torque entregado por el motor, la velocidad angular y el torque del freno, todas ellas durante un intervalo de 25 seg. Aquí son presentados todos estos valores como magnitudes adimensionales, es decir, referidos a un valor máximo unitario. De esta forma, el gráfico admite una única escala en las ordenadas que facilita su interpretación.

6.5 Evaluación del torque del motor a partir del torque del freno

Una vez que fue comprobado que el comportamiento dinámico del motor y freno es correctamente simulado, se procede a evaluar los resultados de la determinación del torque del motor a partir de los valores del torque del freno. En todos los casos se mantiene un acoplamiento rígido vinculando ambos elementos principales.

Como se recordará, la imposibilidad de medir directamente el torque del motor hace necesaria su evaluación indirecta a partir de otros parámetros. Para ello, debe calcularse la aceleración del motor (Ec. 72), para posteriormente determinar el valor de su torque con la ecuación 89 del punto 4.4.5. Éste es un punto crítico que tendrá mucha importancia en el desempeño de la unidad de control, razón por la cual se hacen numerosas pruebas tendientes a confirmar que el error obtenido está en todos los casos dentro de un rango aceptable.

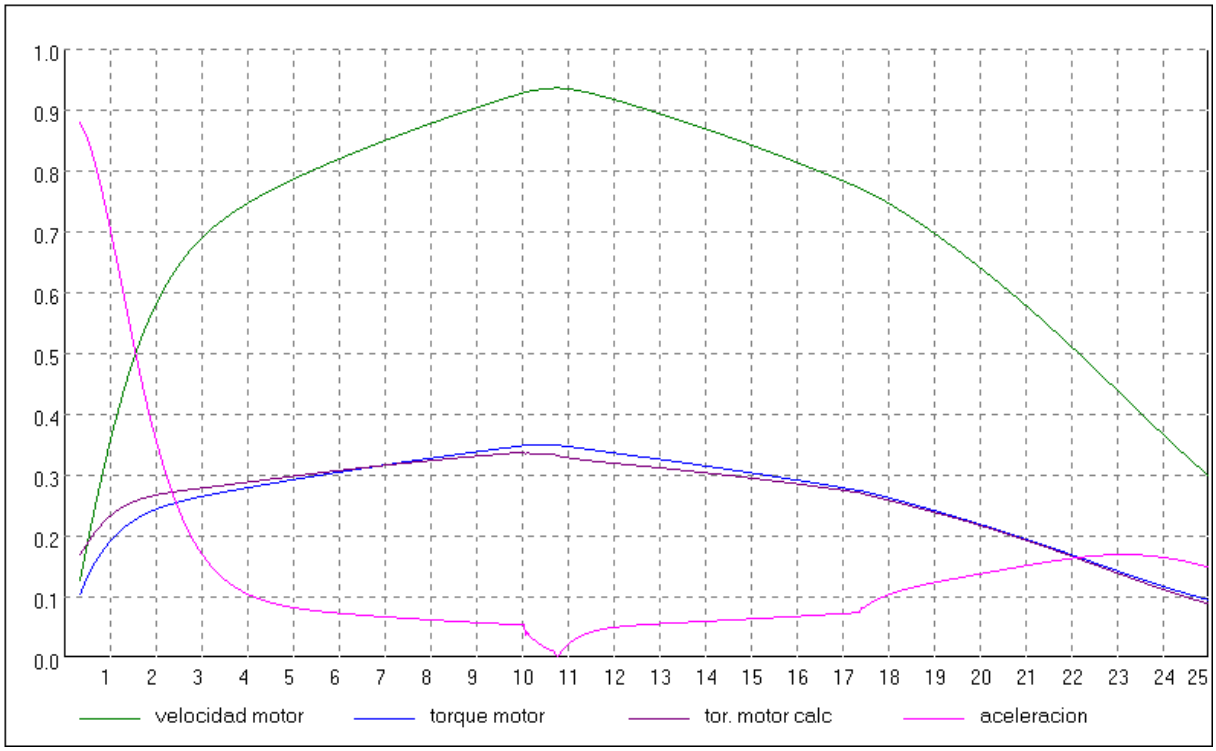


Figura 62: Evaluación del torque del motor a partir del torque del freno

En la Figura 62, se presentan resultados correspondientes a una curva suave de aceleración y desaceleración del motor, que se desarrolla en un intervalo de 25 seg. En el mismo gráfico se presentan las curvas de aceleración, torque real del motor y torque del motor determinado en forma indirecta. La aceleración fue calculada a partir de la velocidad, y es representada en valor absoluto, para no tener que ampliar la escala del gráfico al campo de los números negativos. Como se

observa, el error obtenido en la determinación del torque es suficientemente pequeño en la zona en que se operará con la unidad de control. Aquí también, como en todos los casos, se representan magnitudes adimensionales.

6.6 Evaluación de la interacción del conjunto motor-freno y la unidad de control

El último paso en este proceso de simulación es la evaluación de la unidad de control operando sobre el conjunto compuesto por el motor y freno. Algunos de los resultados obtenidos son presentados en las Figuras 63 y 64, en las que se representan la velocidad, el torque del motor y el torque del freno. Los resultados obtenidos estuvieron, en todos los casos, dentro de los valores esperados.

En el primer ejemplo se lleva un motor al 80% de velocidad y se lo estabiliza en ese valor con una carga de aproximadamente el 60%, representado en el gráfico por el punto A. Para ello se trabaja en forma simultánea sobre el acelerador y freno, de acuerdo a lo previsto por el estado q_5 de la Tabla 11. Tal como fue establecido, la operación bajo el estado q_5 prevé una acción simultánea sobre ambas variables de control. Luego, operando en el estado q_1 se trabaja sólo con el acelerador, para reducir la velocidad y estabilizarlo al 45% de la velocidad máxima, representado por el punto B.

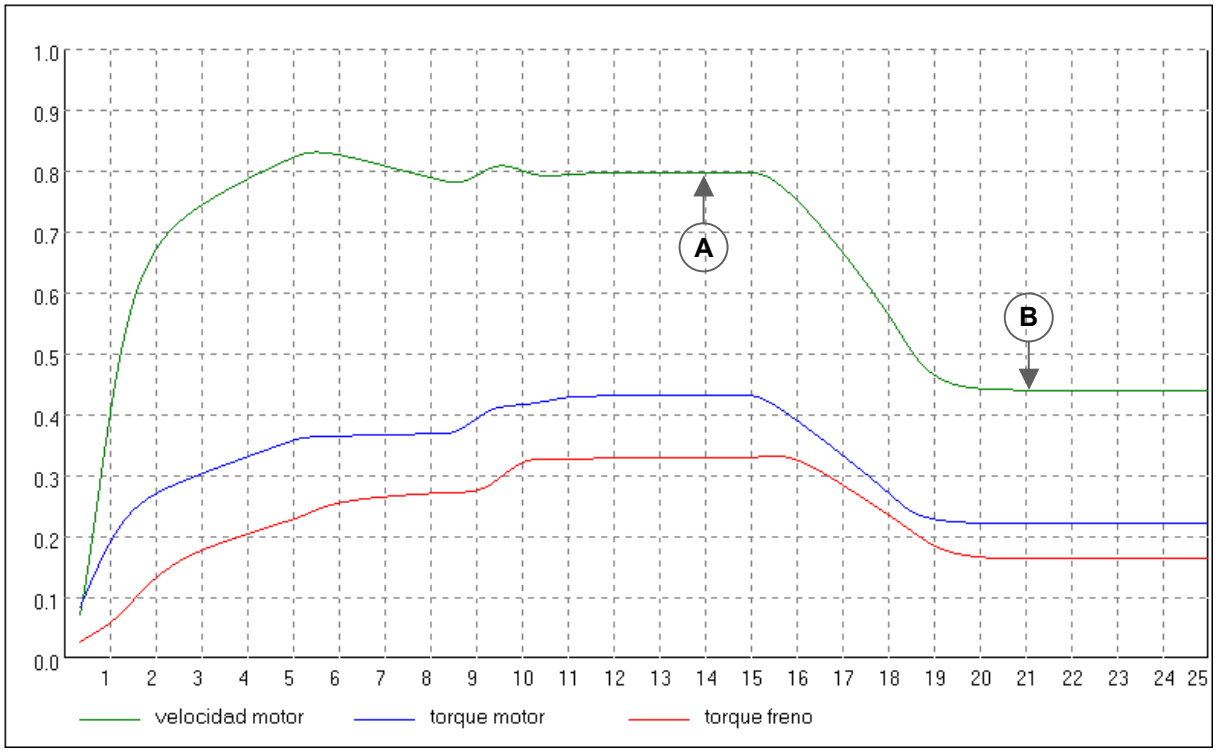


Figura 63: Respuesta del conjunto motor-freno ante cambios en las condiciones requeridas

En el segundo caso se lleva suavemente el motor al 85 % de la velocidad con un torque del 60%, como se representa en el punto C de la Figura 64. Aquí la unidad de control operó alternativamente entre los estados q2 y q3, es decir sobre una sola variable de control por vez. Nótese que son las mismas acciones de control que las previstas en el estado "q_5", sólo que en este primer caso las acciones son simultáneas y se cumplen en paralelo. Como puede observarse, y ya fue anticipado, para pasar de una condición a otra sobre el plano de estados se dispone de

numerosas variantes. Finalmente, el motor es desacelerado hasta alcanzar la condición definida por el punto D.

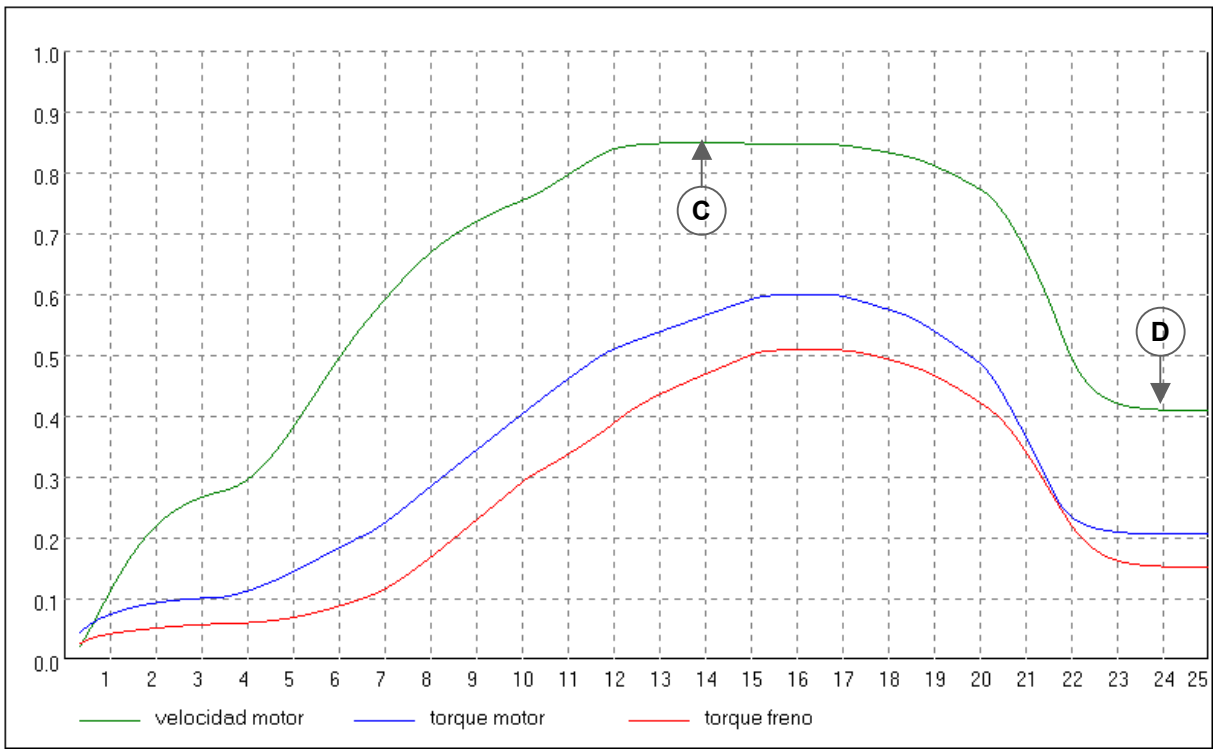


Figura 64: Respuesta del conjunto motor-freno ante cambios en las condiciones requeridas

En las obtención de estos resultados se consideró que tanto el motor como el freno comienzan a responder en forma inmediata, una vez alteradas sus variables de control. Esto equivale a decir que el parámetro T_1=0 en ambos casos (Figuras 26 y 28),

6.7 Resultados obtenidos y su proyección futura

Las pruebas realizadas permitieron comprobar que el simulador desarrollado, reproduce correctamente el comportamiento de un conjunto motor-freno, su unidad de control y los dispositivos auxiliares de adquisición de datos y accionamiento. Para llegar a esta conclusión fueron evaluados, por separado, cada uno de los algoritmos utilizados y finalmente se hizo lo propio con el simulador en su conjunto. En todos los casos, los resultados fueron los esperados.

Con este desarrollo se dispone de lo que podría denominarse una *plataforma* de simulación de ensayos de motores, capaz de reproducir los componentes principales de un sistema de ensayos y, en particular, su unidad de control.

Sin embargo, debe observarse que esta plataforma tendrá que ser *configurada* para simular adecuadamente el ensayo de cada motor en particular. Para ello, deben definirse los parámetros que son propios de la instalación de ensayos y los que son específicos de cada tipo de motor.

La configuración de la instalación incluye los siguientes pasos:

a) Entrenar la red neuronal con datos característicos que serán obtenidos experimentalmente del freno dinamométrico.

b) Conocer el momento de inercia del rotor del freno.

c) Conocer el torque necesario para arrastrar el freno a velocidad constante, a fin de determinar las fuerzas de rozamiento estática y dinámica.

d) Conocer el tiempo de respuesta del freno (constante T_1 , Figura 28).

e) Conocer las frecuencias de muestreo de la velocidad y torque.

Para el caso del motor, este proceso de configuración incluirá:

a) Entrenar la red neuronal, con datos característicos a ser obtenidos del motor a ser simulado.

b) Conocer el momento de inercia rotatorio equivalente del motor.

c) Conocer el torque necesario para arrastrar el motor a velocidad constante, a fin de determinar las fuerzas de rozamiento estática y dinámica.

d) Conocer el tiempo de respuesta del acelerador y del motor respecto del acelerador (constantes T_a y T_1, Figura 26).

Una vez ajustados los parámetros del sistema, éste podrá reproducir correctamente el ensayo de cierto motor en todas sus condiciones de operación. Luego, el esfuerzo se debe orientar a evaluar y completar la tabla de criterios de control, a través de la definición o ajuste de las reglas de producción. Se podrán así obtener transiciones más rápidas y respuestas menos oscilantes.

Tal como puede observarse, a partir de su configuración, el simulador quedará disponible para reproducir ensayos de motores y su interacción con la unidad de control. Nótese que, para representar fenómenos vibratorios, el modelo dinámico debe ser representado por el sistema de ecuaciones diferenciales (Ec. 78 a 80), en reemplazo de la ecuación diferencial que representa al sistema rígido (Ec. 83).

7 Conclusiones

Una unidad conmutada digital de control, destinada a operar sobre un sistema mecánico, fue el eje de este trabajo. En su definición se reunieron conceptos de sistemas híbridos, teoría de control y ciencia de la computación. Su capacidad de conmutar, para adoptar la lógica de control más apropiada en cada caso, quedó a cargo de un *Autómata Finito*. Sin embargo, esta función pudo haber sido cumplida por una red de *Petri*, la que hubiese sido preferible en caso de haberse presentad la necesidad de sincronizar procesos paralelos o concurrentes. Por no ser éste el caso, finalmente, se seleccionó para esta función al mencionado *Autómata Finito*. Esta capacidad de conmutación está destinada a permitir la estructuración dinámica de diferentes lógicas, que incluyen opciones de realimentación y cálculo del ajuste de las variables de control. A partir de estos conceptos, se concibió un modelo “abierto”, cuya verdadera capacidad de conmutación se apoya en una tabla de criterios de control y donde estos criterios toman la forma de reglas de producción.

Los sistemas de computación de este tipo son desarrollados e implementados por profesionales del área de informática, o por especialistas en distintas disciplinas tecnológicas, normalmente ingenieros de las distintas especialidades. Los primeros

saben cómo se debe desarrollar un sistema y deben elicitar los conocimientos relativos al dominio del problema tratado, mientras que, por el contrario, los especialistas tecnológicos conocen perfectamente el dominio del problema pero no necesariamente saben mucho de ingeniería de software. Aquí cabe recordar que estos últimos trabajan desde que fue presentado el primer compilador Fortran, desarrollando por lo tanto sistemas desde antes que la propia Ingeniería de Software exista. En todo este tiempo, estos especialistas han establecido toda una cultura que caracteriza sus desarrollos, los que giran en torno a modelos evolutivos con muchos puntos de contacto con lo que hoy se denominan "metodologías ligeras", tales como el "eXtreme Programming (XP)" o "*SCRUM*".

Sin embargo, en muchos casos, la excesiva confianza que brinda el buen conocimiento sobre el dominio del problema parecería justificar la ausencia de una apropiada planificación o de un diseño suficientemente detallado y por lo general los resultados son muy elocuentes, manifestándose en presupuestos y plazos excedidos, en la escasa calidad del producto final o en enormes dificultades para su mantenimiento.

¿Cómo justificar que profesionales habituados a planificar cuidadosamente los trabajos que son propios de su especialidad, hagan todo lo contrario cuando se trata de desarrollar software? Una posible respuesta a este interrogante lleva a formular a su vez otra pregunta: ¿han contemplado las técnicas y metodologías dominantes la necesidad de prever el desarrollo de este tipo de sistemas?

En respuesta a esta inquietud y, tal como fue expresado al enunciarse el segundo de los objetivos particulares de este trabajo, la necesidad de disponer de un procedimiento más acorde a la naturaleza de los sistemas de tiempo real llevó a proponer y ensayar una nueva metodología para su modelado. Se trata en realidad de una combinación de técnicas ampliamente conocidas, que tiene como centro al *análisis esencial,* y donde se recurrió a las responsabilidades del sistema para establecer el nexo con el modelo de objetos.

Con la definición de estos marcos, teórico y metodológico, quedaron establecidas bases sólidas para alcanzar el objetivo central de este trabajo, que fue el desarrollo de un *simulador de tiempo real* de una unidad de control. Así, estas ideas fueron aplicadas en el desarrollo del modelo y los resultados obtenidos superaron las expectativas, permitiendo una identificación rigurosa de las clases y sus relaciones a través de un proceso que permite ser recorrido hacia atrás, asegurando su completa trazabilidad. Obviamente, aunque esta propuesta debe aún ser perfeccionada y aplicada a sistemas de distinta magnitud y naturaleza, en primera instancia demostró ofrecer una respuesta adecuada al problema planteado. En efecto, se piensa que este procedimiento no sólo es apropiado para modelar *sistemas de tiempo real*, sino que podría ser utilizado ventajosamente para todo tipo de sistemas, particularmente técnicos. En ese caso, se estaría brindando una respuesta efectiva a la señalada necesidad de una metodología de modelado acorde a la naturaleza de estos sistemas.

Finalmente, el modelo fue implementado y sus primeras aplicaciones parecen indicar su buen desempeño, tal como se concluye de los resultados presentados y sus evaluaciones. Por tratarse de un sistema abierto, las experiencias que se obtengan de su aplicación podrán ser empleadas para mejorar progresivamente su

desempeño, ajustando sus criterios de control o incorporando otros nuevos. Así, una aplicación intensiva permitirá identificar y corregir eventuales debilidades.

A partir de haberse alcanzado el objetivo de este trabajo, se abren numerosos nuevos caminos. En uno de ellos, la incorporación de propiedades elásticas al modelo mecánico permitirá reproducir fenómenos vibratorios y evaluar su incidencia en la calidad del desempeño de la unidad de control. En otro, podrá dotarse de cierta *inteligencia* a la unidad de control, de manera de permitirle mejorar su desempeño a partir de sus propias experiencias, convirtiéndolo en un sistema de control *adaptativo inteligente*. Como puede comprobarse, con este trabajo se dieron los primeros pasos en un camino que a poco de andar comienza a ofrecer múltiples opciones para continuarlo, algunas de ellas insospechadas. La intención es seguir adelante.

Bibliografía

Ingeniería de Software

1. Antonelli L., Oliveros A.; Rossi G. (1999); "*Baseline Mentor, An Application that Derives CRC Cards from Lexicon and Scenarios*", XXVIII JAIO, II Workshop Iberoamericano en Ingeniería de Requerimientos, WER'99, Buenos Aires, Argentina, septiembre 9 y 10.
2. Antonelli L.; Oliveros A. (2001); "*Traceability en la Etapa de Elicitación de Requerimientos*", Workshop de Ingeniería de Requerimientos, Buenos Aires, Argentina, noviembre 22 y 23, Pg. 1-19.
3. Beck K.; Cunningham, W. (1989); "*A Laboratory for Teaching Object-Oriented Thinking*", Proceedings OOPSLA'89, Pg. 1-6.
4. Becker L., Pereira C., Gergeleit M.; Nett E. (1999); "*An Integrated Environment for the Complete Development Cycle of an Object-Oriented Distributed Real-Time System*", Proceedings of 2nd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'99), Saint-Malo, France,Pg.165.
5. Bellin D.; Suchman S. (1997); "*The CRC Card Book*", Addison-Wesley.-
6. Boehm B. (1984); "*Verifying and Validating Software Requirements Specifications*", IEEE Software, Vol. 1, N° 1, Pg. 75-88.
7. Bollella G.; Gosling J. (2000); "*Real-Time Specification for Java*", Computer, Junio, Pg. 47-54.
8. Bustos Reinoso, G. (1999); "*Modelado Orientado a Objetos: Una Evaluación Crítica*", en Revista Facultad de Ingeniería, Nº 10, Octubre, Facultad de Ingeniería, Pontificia Universidad Católica de Valparaíso, Valparaíso.
9. Constantine L. (1989); "*Object-Oriented and Structured Methods: Towards Integration*" *American Programmer*, Vol. 2, N° 7-8, agosto, Pg. 34 - 40.
10. Constantine L. (1995); "*What do users want? Engineering Usability into Software*", Window Technical Journal, December.
11. Del Bianco V., Lavazza L., Mauri M.; Occorso G. (2003); "*Towards UML-based formal specifications of component based real-time software*", ETAPS, European Joint Conferences on Theory and Practice of Software, Polonia.

12. Douglass B. (2000); "*Real Time UML*", Second Edition, The Component Software Series, Addison-Wesley.
13. Drake, González, Harbour, Gutiérrez; Palencia (2001); "*UML-MAST : Visual Modeling and Analysis Suite for Real-Time Applications with UML*", http://ctrpc17.ctr.unican.es/umlmast.
14. Gamma E., Helm R., Jonson R.; Vlissides J. (1995); "*Design Patterns: Elements of Reusable Object-Oriented Software*", Addison-Wesley.-
15. Gao Q., Brown L.; Capretz L. (2003); "*UML Extensions for Real-Time Control Systems*", Proceedings of the 42nd IEEE Conference on Decision and Control, Pg. 5932-5938.
16. Huang, Dongping, Sarjoughian; Hessam (2004); "*Software and Simulation Modeling for Real-time Software-intensive Systems*", Proceedings of the 8th IEEE International Symposium on Distributed, Simulation and Real-Time Applications (DS-RT'04).
17. Jacobsen I. (1992); "*Object-Oriented Software Engineering: A Use Case Driven Approach*", Addison-Wesley, Reading, Mass.
18. Jacobson I., Booch G.; Rumbaugh J. (1999); "*The Unified Software Development Process*", Addison Wesley.
19. Leite J.; Oliveira A. (1995): "*A Client Oriented Requirements Baseline*", Proceedings of RE 95, Second IEEE International Symposium on Requirements Engineer, UK, IEEE Computer Society Press, Pg 108-115
20. Leonardi M., Rossi G.; Leite J. (1998); "*Un modelo de hipertexto para la especificación de Requisitos*", Anais do WER98 - Workshop em Engenharia de Requisitos, Maringá-PR, Brasil, Outubro 12, 1998, pp 119-128.
21. Meyer B. (1997); "*Object-Oriented Software Construction*", Prentice Hall,2º Ed.
22. McMenamin S.; Palmer J. (1984); "*Essential Systems Análisis*", Prentice Hall.
23. Mills D. (2002); "*What's the Use of a Use Case?*", Software Education Associates Ltd, Wellington, New Zealand.
24. Oliveros A. (2001); "*Tópicos de Ingeniería de Software I*", Apuntes de clase, Magister de Ingeniería de Software, UNLP-UBP.
25. Page-Jones M. (1992); "*The Synthesis Method*", Panel Report: From Events to Objects, Addendum to the Proceedings, OOSPLA'92, Vancouver, Canada.
26. Prieto M. (2002); "*Técnicas y Herramientas de Desarrollo*", Apuntes de clase, Magister de Ingeniería de Software, UNLP-UBP.
27. Rubin K.; Goldberg A. (1992); "*Object Behavior Análisis",*Communications of the ACM, Vol.35, No. 9, Pg.47-62.
28. Ruble D. (1997); "*Practical Analysis & Design for Client / Server & GUI Systems*", Prentice-Hall Inc.
29. Rumbaugh J., Blaha M., Premerlani W., Eddy F.; Lorensen W. (1991); "*Object-Oriented Modeling and Design*", Prentice-Hall Inc.
30. Satzinger J. (1993); "*Introduction to Essential Systems Analysis*", Department of Management, University of Georgia, USA.

31. Sommerville I. (2002); "Ingeniería de Software", 6a edición, Addison Wesley.-

32. Wieringa R. (1998); "*A Survey of Structured and Object-Oriented Software Specification Methods and Techniques*", ACM Computing Surveys, Vol.30, No. 4, December, Pg. 467-471.

33. Wirfs-Brock R., Wilkerson B.; Wiener L. (1990); "*Designing Object-Oriented Software",* Prentice Hall.

34. Yourdon E. (1993); "*Modern Structured Analysis*", Prentice-Hall Inc.

35. Zalewski J. (2002); "Real-Time Software Design Patterns", 9th Polish Conference on Real-Time Systems, Ustron, Poland, September, Pg. 23-42.

Ensayos de motores

36. Arsie I., Pianese C.; Rizzo G. (1998); "*Enhancement of Control Oriented Engine Models Using Neural Network",* Proc. of the 6th IEEE Mediterranean Conference on Control Systems, Italy.

37. Arsie I., Marotta F., Pianesse C.; Risso G.(2001); "*Information Based Selection of Neural Networks Training Data for S.I. Engine Mapping",* SAE Technical Paper No. 01P23-196.

38. Borman G., Myers P.; Uyehara O. (1971); "*Some Problem Areas in Engine Simulation*", Paper No. 710172, SAE Automotive Engineering Congress, Detroit.-

39. Balluchi A., Benvenuti L., Di Benedetto M., Pinello C.; Sangiovanni A. (2000); "*Automotive Engine Control and Hybrid Systems: Challenges and Opportunities*", Proceedings of de IEEE, Vol .88, No. 7, July, Pg. 888 - 912.

40. Balluchi A., Bicchi A., Caterini C., Rossi C.; Sangiovanni-Vincentelli A. (2000); "*Hybrid Tracking Control for Spark-Ignition Engines*", Proceedings of the 39th IEEE Conference on Decision and Control, Sydney, Australia.

41. Balluchi A., Di Benedetto M., Pinello C.; Sangiovanni A. (2001); "*Mixed Models of Computation in the Design of Automotive Engine Control*", Proceedings of the 40th IEEE Conference on Decision and Control, Orlando, Florida, USA, December.

42. Boehm C.; Allan J. (1973); "*Parameter Sensitivity Studies for Internal Combustion Engine Control and Design Purposes*", Proceedings of the 10th workshop on Design automation, Annual ACM IEEE Design Automation Conference, Pg. 240-246.

43. Brace C., Deacon M., Vaughan N., Horrocks R.; Burrows (2001); "*The Compromise in Reducing Exhaust Emissions and Fuel Consumption from Diesel CVT Powertrain Over Typical Usage Cycles*", University of Bath & Ford Motor Company, U.K.

44. Bykhovsky I. (1972); "*Fundamentals of Vibration Engineering* ", Ed. MIR, Moscú.

45. Den Hartog J. (1964); *"Mecánica de las Vibraciones* ", CECSA.

46. Hafner M. (2001); *"Model Based Determination of Dynamic Engine Control Function Parameters",* Society of Automotive Engineers, 01FL-319, MHafner@iat.tu-darmstadt.de.

47. Hori M., Suzuki M., Nomura M.; Terashima M. (1995); "High-Performance Automotive Engine Control in Engine Tester", Industry Applications Conference, Annual Meeting, IAS '95, Pg.1572 – 1579.

48. Rubin Z., Munns S.; Moskwa J. (1997); "*The Development of Vehicular Powertrain System Modeling Methodologies : Philosophy and Implementation*", Paper No. 971089, University of Wisconsin-Madison.

49. Timoshenko; Young (1955); "*Vibration Problems in Engineering* ", Van Nostrand Co.

50. Yin X.; Ge A. (2001); "*A Dynamic Model of Engine Using Neural Network Descriptions",* IEEE, Pg 109-114.

Sistemas híbridos y sistemas híbridos de control

51. Alla H.; David R. (1987); "*Continuous Petri Nets"*, Proc. 8th Int. Workshop on Applications and Theory of Petri Nets", Zaragoza, Spain, Pgs. 275-294.

52. Alur R., Courcoubetis C., Halbwachs N., Henzinger T., Ho P., Nicollin X., Olivero A., Sifakis J.; Yovine S. (1995); "*The algorithmic analysis of hybrid systems*", *Theoretical Computer Science 138*, pages 3-34. Springer-Verlag.

53. Alur R., Courcoubetis C., Henzinger T.; Ho P. (1993); "*Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems*", *Hybrid Systems, R. L. Grossman et al., Eds., Lecture Notes in Computer Science, vol. 236, Springer-Verlag, Berlin,* 209–229.

54. Antsaklis P. (2000); "*A Brief Introduction to the Theory and Applications of Hybrid Systems*", Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications, Vol.88, No.7, pp. 879-887.

55. Branicky M. (1998); "*Multiple Lyapunov Functions and Other Análisis Tools for Switched and Hybrid Systems*", IEEE Transactions on Automatic Control, Vol.43, No.4.

56. Branicky M., Borkar V.; Mitter S. (1998); "*A Unified Framework for Hybrid Control: Model and Optimal Control Theory*", IEEE Transactions on Automatic Control, Vol.43,No.1,Pg. 31-45.

57. Branicky M.; Zhang G. (2000); "Solving Hybrid Control Problems: Level Sets *and Behavioral Programming"*, Proceedings of the American Control Conference, June, Pg.1175 – 1180.

58. Champagnat R., Esteban P., Pingaud H.; Valette R. (1998); "*Petri net based modeling of hybrid systems*", *Computers in Industry*, 36(1–2), pp. 139–146.

59. DeCarlo R., Zak S.; Matthews G. (1988); "*Variable Structure Control of Nonlinear Multivariable Systems: A Tutorial*", Proceedings of the IEEE, Vol 76, No.3.

60. DeCarlo R., Branicky M., Pettersson S.; Lennartson B. (2000); "*Perspectives and Results on the Stability and Stabilizability of Hybrid Systems*", Proceedings of the IEEE, Vol 88, No.7.

61. Demongodin I.; Koussoulas N. (1996); "*Modeling Dynamic Systems Through Petri Nets"* , Proc. IEEE –Systems, Man and Cybernetics, Pgs. 279-284.-

62. Fahrland D. (1970); "*Combined discrete event continuous systems simulation*" Simulation 14, Pg. 61–72.

63. Koutsoukos X.; Antsaklis P. (1999); "Design of hybrid system regulators", Proceedings of the 38th IEEE Conference on Decision and Control, Pg. 3990-3995.

64. Koutsoukos X., Antsaklis P., Stiver J.; Lemmon M. (2000); "Sepervisory Control of Hybrid Systems", Proceedings of the IEEE, Vol. 88, No. 7, July, Pg.1026–1049.

65. Le Bail J., Alla H.; David R. (1991); "*Hybrid Petri Nets*", Proc. 1st Int. European Control Conference, ECC91 (Grenoble, France), pp. 1472–1477.

66. Mascaros V., Garcia E., Morant F., Correcher A., Quiles E.; Blasco R. (2002); "*Modelado de Sistemas Híbridos de Control*", Dpto. de Ingeniería de Sistemas y Automática, Universidad Politécnica de Valencia, España.

67. Murata T. (1989); "*Petri Nets: Properties, Analysis and Applications"*, Proceedings IEEE, Vol 77, No.4, Pg. 541-580.

68. Utkin, V. (1977); "*Variable Structure Systems with Sliding Modes*", IEEE Transactions on Automatic Control, Vol. AC-22, No 2.

Teoría de control

69. Creus A. (1987); "*Simulación y Control de Procesos por Ordenador*", Ed. Marcombo.

70. Goodwin G., Graebe S.; Salgado M. (2001); "*Classical PID Control*", Prentice Hall.

71. Minorsky (1922); "*Directional stability of automatically steered bodies*", Journal of the American *Society of Naval Engineering*, Vol. 34, p. 284.

72. Ogata K. (2003); "*Ingeniería de Control Moderna*", 4a Edición, Pearson.-

73. Voda A.; Landau I. (1995); "*A Method for the Auto-calibration of PID Controllers*", Automatica, 31(1) 41-53.

74. Ziegler J.; Nichols N. (1942); "*Optimum settings for Automatic Controllers*", Transactions of the ASME, 64:759-768.

Ciencia de la computación

75. Arbib M. (1987); "*Brains, Machines and Mathematics*", Springer-Verlag, New York.

76. Wiener N. (1948); "*Cybernetics: or Control and Communications in the Animal and the machine*", J. Wiley & MIT Press, USA.-

Métodos numéricos

77. Alfaro V. (2004); "*Métodos numéricos para la solución de ecuaciones diferenciales ordinarias (EDO)",* Departamento de Automática, Escuela de Ingeniería Eléctrica, Universidad de Costa Rica.

78. Alvarez López J. (2002); "*Métodos GRK para ecuaciones diferenciales ordinarias",* Tesis Doctoral, Departamento de Matemática Aplicada a la Ingeniería, Universidad de Valladolid.

79. Chapra S. (2004); "*Applied Numerical Methods*",McGraw-Hill.

80. McCracken D.; Dorn W. (1964);"*Numerical methods and fortran programming",* John Wiley & Sons.

81. Rogers, D.; Adams, A. (1990); "*Mathematical Elements for Computer Graphics*", McGraw Hill.

Redes Neuronales

82. Delashmit W.; Manry M. (2003); "*New Training Algorithms for Dependently Initialized Multilayer Perceptrons* ", IEEE 0-7803-8104-1/03.

83. Delashmit W.; Manry M. (2002); "*Enhanced robustness of multilayer perceptron training*", IEEE Conference on Signals, Systems and Computers, Vol.2, Pg. 1029-1033.

84. Isasi V.; Galván L. (2003); "*Redes de Neuronas Artificiales: un enfoque práctico*", Pearson-Prentice Hall.

85. Nilsson N. (2001); "*Inteligencia Artificial : una nueva síntesis* ", McGraw Hill.

86. Plagianakos, Magoulas; Vrahatis (2002); "*Deterministic Nonmonotone Strategies for Effective Training of Multilayer Perceptrons "*, IEEE Transactions on Neural Networks, Vol.13, No.6.

87. Riedmiller M. (1994); "*Rprop : Description and Implementation Details* ", Technical Report, University of Karlsruhe.

88. Rumelhart D., Hinton G.; Williams, R. (1986); "*Learning internal representations by error propagation in Parallel Distributed Processing: Explorations in the Microstructures of Cognition* ", Vol 1, MIT Press, pp 318-362.

89. Tan Y.; Dang X. (1999); "*Generalised nonlinear PID controller based on reural networks"*, Proceedings of the 1999 Information, Decision and Control, Adelaide, Australia, February 8-10, 1999; IEEE, USA, 1999, 519-524.

90. Valishevsky A. (1998); "*Comparative analysis of different approaches towards multilayer Perceptron training*", Department of Computer Science, University of Latvia, Riga.

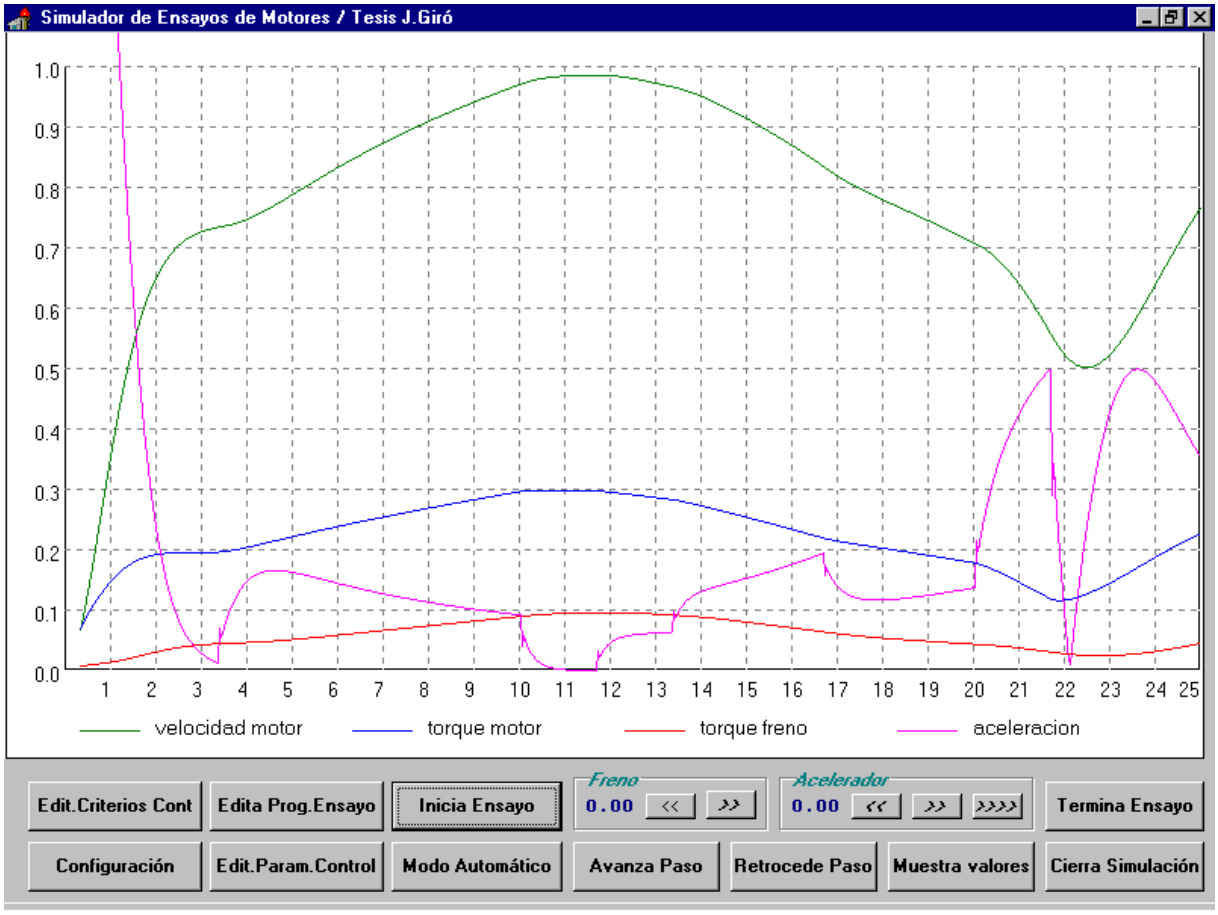
91. Wlodzislaw, Adamczak; Jankowski (1997); "*Initialization and Optimization of Multilayered Perceptorns* ", Department of Computer Methods, Copernicus University, Poland.

92. Xiao J., Chen Z., Cheng J. (1996); "*Structure Study of Feedforward Neural Networks for Approximation of Highly Nonlinear Real-valued Functions",* IEEE.- Ingeniería de software.

Otros

93. Maiztegui A., Gleiser R. (2000); "Mediciones de Laboratorio", publicación editada por los autores, Córdoba.

Anexo A: pantalla principal del simulador



Anexo B: Simbología

Sistema controlado

$\boldsymbol{w}$	: Vector de estado híbrido
$\mathbf{x}$	: Vector de setpoints
$\boldsymbol{y}(t)$	: Vector de estado de planta
$\boldsymbol{z}(t)$	: Vector de señales de control
$\mathbf{A}$	: Matriz Hurwitz
V	: Función de Lyapunov
$\mathbf{Q}$	: Opciones de conmutación
ξ	: Función de transición
e	: Vector error
g	: Acción del sistema de control
h	: Acción del propio sistema
$\rho_1(\mathbf{y},t)$	: Superficie de deslizamiento

Sistema conmutado de control

$\boldsymbol{S}$	: Conjunto finito de estados
$\boldsymbol{\Sigma}$	: Alfabeto de símbolos de entrada
$\boldsymbol{R}$	: Alfabeto de símbolos de salida
δ	: Función de transición
ϕ	: Función de salida
So	: Estado inicial ($So \in \boldsymbol{S}$)
$\boldsymbol{S}F$	: Condiciones de detención ($\boldsymbol{S}F \subset \boldsymbol{S}$)
$\boldsymbol{P}$	: Subconjuntos de $\boldsymbol{S}$

Interfases

χ^y	: Conjunto de eventos reconocidos
$\mathbf{f}_\chi$	: Regiones en el espacio de estados
ψ	: Relación eventos-alfabeto entrada
$\mathbf{u}$	: Vector de estado modelo discreto

Lógica de control

G	: Función de transferencia
Kp	: Constante proporcional
Kp / Tr	: Constante integral
Kp / Td	: Constante derivativa

Integración de ecuaciones diferenciales

ki	: Constante método Runge-Kutta
Δt	: Intervalo de tiempo
ε	: Error

Aproximación de funciones

$\mathbf{a}$	: Constantes polinomio aproximación
$\mathbf{c}$	: Términos independientes
$\mathbf{B}$	: Matriz de mínimos cuadrados

Redes neuronales

N	: Cantidad de unidades de la primer capa oculta
M	: Cantidad de unidades de la segunda capa oculta
$\{ d \}$	: vector de entradas (Dimensión Nd = 2)
$\{ o1 \}$	: vector de salida de la primera capa oculta (Dimensión N)
$\{ o2 \}$	: vector de salida de la segunda capa oculta (Dimensión M)
$\{ s \}$	: vector de salida (calculada) de la red (Dimensión Ns = 1)
$\{ v \}$	: vector de salida (esperada) de la red (Dimensión Ns = 1)
$[Wdo]$	: matriz de sinapsis entre las capas “d” y “o1” (N x Ne)
$[Woo]$	: matriz de sinapsis entre las capas “o1” y “o2” (M x N)
$[Wos]$	: matriz de sinapsis entre las capas “o2” y “s” (Ns x M)
$[\Delta Wdo]$	: ajuste de la matriz de sinapsis entre las capas “d” y “o1”
$[\Delta Woo]$	: ajuste de la matriz de sinapsis entre las capas “o1” y “o2”
$[\Delta Wos]$	: ajuste de la matriz de sinapsis entre las capas “o2” y “s”
$\{ \theta d \}$	: vector de “offset” (umbral) de las unidades de la primer capa oculta
$\{ \theta o \}$	: vector de “offset” (umbral) de las unidades de la segunda capa oculta
$\{ \theta s \}$	: vector de “offset” (umbral) de las unidades de la capa de salida
$\mathbf{f}$	: función de activación
$\{ es \}$	: vector de errores de salida (Dimensión Ns = 2)
$\{ eo2 \}$	: vector de errores de 2ª capa oculta (Dimensión M)
$\{ eo1 \}$	: vector de errores de 1ª capa oculta (Dimensión N)
$\{ 1 \}$	: vector unitario
$[x]$	: matriz diagonal correspondiente a un vector genérico $\{ x \}$
$[A]^T$	: transpuesta de la matriz $[A]$
E	: error total cometido por la red en “n” puntos
$\boldsymbol{\eta}$	: factor de aprendizaje ($0 < \boldsymbol{\eta} < 1$)
$\boldsymbol{\beta}$	: factor de momentum ($0 < \boldsymbol{\beta} < 1$)
$\boldsymbol{\mu}$	: factor de algoritmo ABP y Rprop
$\boldsymbol{\delta}$	: factor de algoritmo ABP

Dominio del problema

t : Tiempo (variable independiente)
Im : Momento de inercia del motor
Ir : Momento de Inercia del rotor
Ie : Momento de Inercia del estator
Cm : Amortiguamiento viscoso del motor
Cr : Amortiguamiento del rotor
Ce : Amortiguamiento del estator
k : Rigidez de la transmisión cardánica
Fs : Fuerza de fricción de Coulomb
θm : Ángulo girado por el motor
θr : Ángulo girado por el rotor
θe : Ángulo girado por el estator
Tm : Torque entregado por el motor
Tr : Torque absorbido por el freno
Tc : Torque leído en la celda de carga
Fc : Fuerza axial en la celda de carga
L : Distancia desde el eje del freno a la celda de carga
To : Torque deseado en el motor (set point)
a : Posición del acelerador (variable de control)
v : Tensión del freno (variable de control)
ρ : Tren de pulsos originados en una rueda dentada solidaria al motor
ω_m, ω_r : Velocidades del motor y freno (radianes / segundo)
ω_o : Velocidad mínima (radianes / segundo)
ω_L : Velocidad límite (radianes / segundo)
Ω_m,Ω_r : Velocidades del motor y freno (revoluciones por minuto)
Ωo : Velocidad deseada en el motor (revoluciones por minuto)
δ : Deformación medida en celda de carga
μ : Señal digital de habilitación del motor
η : Señal digital de habilitación del freno
λ : Señal digital de seguros
Nc : Cantidad de cilindros del motor
T_a,T_1,T_2: Tiempos de respuesta propios del motor y freno
f_m : Función periódica del comportamiento del motor
g : Función periódica del comportamiento del freno
U_m,U_f,U_t: Regiones de operación del motor, freno y transmisión cardánica
h : Factor de amortiguamiento
Nr : Cantidad de dientes en rueda dentada
P : Potencia (CV)
s_{cc} : Señales contínuas en tiempo continuo
s_{cd} : Señales contínuas en tiempo discreto
s_{dc} : Señales discretas en tiempo continuo
s_{dd} : Señales discretas en tiempo discreto

Nota: En cada sección pueden mantener vigencia símbolos de secciones anteriores. Los símbolos son redefinidos sólo en el caso en que cambie su significado.

--- o ---