Estudio de algunos sistemas CACSD

Rodrigo Moreno Serrano
Universidad de los Andes
e-mail: ro-moren@uniandes.edu.co
José Tiberio Hernández
Universidad de los Andes
e-mail: jhernand@uniandes.edu.co

Resumen

En este articulo se pretende mostrar el estado del arte en sistemas de apoyo al diseño de sistemas de control mostrando los aspectos que se deben tener en cuenta para el diseño de un sistema de este tipo. Al final se muestran los campos de investigación en este tema.

Palabras y frases clave

CACSD, CACE, CAD, control de procesos, diseño de sistemas de control, sistemas de apoyo al diseño. ingeniería del control asistida por computador, diseño de sistemas de control asistido por computador.

1. Introducción

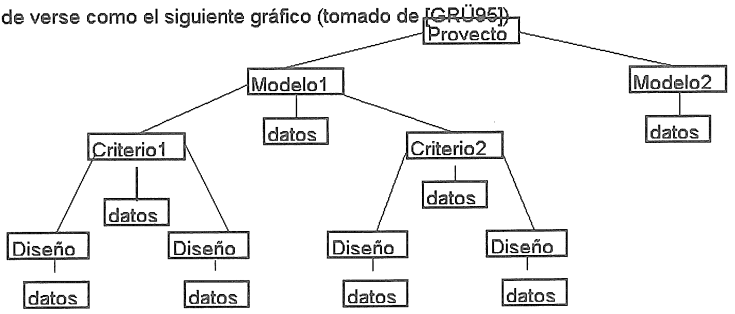
A medida que se ha desarrollado la ingeniería del control, se han desarrollado métodos cada vez más maduros que permiten diseñar mejores controladores teniendo en cuenta muchos más factores del proceso. Sin embargo, estos son extensivos en cálculos numéricos. Paralelamente, los problemas a solucionar crecieron en el número de variables (interrelacionadas) a controlar en el proceso. Por estas razones, se crean herramientas de apoyo por computador al diseño de sistemas de control.

En este documento se pretende hacer una revisión comparativa del estado del arte en aplicaciones de ayuda al diseño de sistemas de control de procesos. En un futuro se piensa estudiar la posibilidad de realizar una herramienta de este tipo usando la experiencia y el trabajo anterior del grupo de investigación DFAC de la universidad de los Andes en el diseño e implantación de sistemas CAD/CAM (e.g. [HER94]) y teniendo como referencia esta revisión.

En la siguiente sección se hará un breve recuento de las actividades del ingeniero de control cuando se enfrenta a un problema (para tenerlo en cuanta en el momento de evaluar las aplicaciones). En la tercera sección se darán los lineamientos generales que debe tener una herramienta de apoyo al diseño de sistemas de control. En la cuarta parte se comentarán los diferentes servicios que debería ofrecer al usuario este tipo de herramientas. En la quinta parte se dará una clasificación de los métodos de diseño usados en estas herramientas. En la sexta parte se hará una breve exposición de los productos estudiados. En la séptima parte se hará un análisis de los sistemas estudiados en la sexta parte y por último se darán algunas conclusiones del trabajo.

2. Proceso de diseño de sistemas de control

Cuando el diseñador se enfrenta al diseño de un sistema de control tiene que realizar básicamente tres actividades: diseñar el modelo del proceso, diseñar el sistema de control en sí y analizar todo el conjunto (o por partes) para verificar la validez de las restricciones planteadas en el diseño. En el proceso de diseño, estas tres actividades se entremezclan y no son independientes puesto que la una requiere de la otra. Una forma de ver el proceso de diseño típico puede verse como el siguiente gráfico (tomado de [GRÜ95]):



En cuanto al diseño en sí, en [NG93] se propone que el proceso es un ciclo en el que se genera un número de soluciones candidatas, se evalúa cada una de ellas y se critican los resultados de esto de lo cual pueden salir nuevas especificaciones para el siguiente ciclo o se puede decidir parar la búsqueda de soluciones. Este proceso es llevado a cabo con la colaboración tanto del diseñador como del computador.

Un sistema CACE (ingeniería de control asistida por computador) intenta ser una herramienta más genérica que una CACSD (porque la última estaría dedicada solo al diseño), sin embargo, adelante se tratarán las palabras CACE y CACSD como sinónimas porque están intimamente relacionadas.

3. Clasificación de los métodos de diseño de sistemas de control

En [JAM92] se clasifican los métodos en tres dependiendo del tipo de sistema al que aplica: métodos para sistemas SISO (single input - single output), métodos para sistemas MIMO (multiple input - multiple output) y métodos para sistemas de muy alto orden.

Por otra parte, en [NG93], los métodos de diseño son clasificados en tres grandes grupos dependiendo de la cantidad de información analítica que se posea del sistema:

- Métodos de síntesis: se pueden aplicar cuando el problema está bien formulado en el sentido de que las especificaciones no son ambiguas y el problema planteado es implementable.
- Métodos basados en procedimientos: se pueden aplicar cuando las especificaciones no son exactas pero se tiene una buena cantidad de experiencia en ese tipo de problemas (e.g. usando métodos iterativos).
- Métodos basados en búsqueda: consiste en que dado un conjunto de clases de controladores, se busca cual es el que más se ajusta a los requerimientos del problema y con cuales parámetros.

4. Requerimientos básicos de un sistema CACE

Según [Mac89] cualquier sistema de ayuda por computador (para cualquier área de la ingeniería) debe permitir al usuario las hacer las siguientes cosas:

"construir, analizar, navegar, buscar, comparar y evaluar, razonar y hacer hipótesis, sintetizar, diseñar, manipular y modificar, experimentar, catalogar, grabar y restaurar".

Mientras que el ambiente debe ser:

"fácilmente comprensible por un individuo, amplio en el dominio en que trabaja, modular (con un número manejable de partes), predecible en su comportamiento, integrado y con relaciones coherentes entre sus partes, útil con acceso eficiente a la información relevante, tolerante a errores y debe permitir ver el efecto de ellos, extensible y adaptable, y autodocumentado."

Un sistema CAD para sistemas de control debe ayudar al ingeniero a realizar tres tareas fundamentalmente ([GRÜ95]): modelaje de la planta a controlar, análisis y simulación tanto de la planta como del conjunto planta/control y diseño del control de la planta; dado que estas son las actividades que el diseñador realiza cuando se enfrenta a un problema de control.

Debido a la gran cantidad de acciones que debe permitir un paquete CACSD, algunos han decidido proponer arquitecturas abiertas en las cuales se pueden introducir un número ilimitado de herramientas que se interrelacionan entre sí usando las facilidades que se ofrecen en el ambiente donde se implantan [GRÜ95] [RUT95].

Los productos que tienen arquitectura abierta deben cumplir los siguientes requerimientos [BAR93]: debe proveer un buen conjunto de facilidades básicas de modelaje, debe proveer un solo paradigma (e.g. la aplicación debe seguir el paradigma OO), debe proveer herramientas simples fácilmente integrables, debe ofrecer extensibilidad y configurabilidad (i.e. adicionar nuevas herramientas fácilmente y debe poder configurarlas el diseñador), debe ofrecer interoperabilidad entre herramientas, debe proveer soporte para otras aplicaciones (abiertas), debe asegurar su apertura (i.e. no debería ser necesaria la inclusión de herramientas propietarias).

5. Componentes de un CACE abierto

Teniendo en cuenta que la idea de un sistema de este tipo es ofrecer interoperabilidad entre aplicaciones de tal forma que se puedan integrar fácilmente nuevas herramientas especializadas y crear nuevas con las ya existentes, en [BAR93] se propone un esquema en el cual un CACE está compuesto de los siguientes componentes:

5.1. Servicios de modelaje

En [BAR93] se propone dividir los servicios de modelaje en tres capas:

- Bus de datos con modelo neutral

En este nivel se debe manejar un modelo lo más general posible en el sentido de que pueda ser transformado a cualquier otro tipo de modelo fácilmente. Las herramientas deben comprender ese modelo puesto que se deben comunicar entre sí mediante este bus de datos. El esquema de representación más usado para este propósito es el

de ecuaciones diferenciales algebraicas porque es el que más se acerca a este propósito. Un lenguaje que permite expresar dichas ecuaciones es DS-Block (DS-Block permite expresar dichas ecuaciones usando bloques de control) [OTT92]. En este documento vamos a suponer que el esquema de representación usado como modelo neutral son las ecuaciones diferenciales algebráicas.

- Descripción del modelo en alto nivel

El poder de representación de las ecuaciones diferenciales algebráicas es muy grande y esta es la razón por la cual se usa como bus de datos con modelo neutral. Sin embargo, los lenguajes que se usan en éste nivel no son prácticos para los dieñadores de los modelos porque no facilitan la estructuración de los modelos (i.e. no tienen facilidades para dividir los modelos en submodelos). Por la misma razón, los lenguajes del bus de datos con modelo neutral no ofrecen facilidades para la reutilización de modelos.

Por estas dos razones, una herramienta CACE debería proveer un lenguaje de modelaje de más alto nivel al diseñador que le permita dar una mejor estructura a sus modelos, de modo que pueda identificar las partes de la planta en el modelo de una forma más clara. Además debe permitir ver al diseñador una parte de la planta en el modelo con el nivel de detalle que se desee.

Por otro lado, estos lenguajes deben proveer una mayor reutilización de los modelos.

Los paquetes CACSD en la actualidad ofrecen lenguajes de alto nivel que se basan en bloques que pueden contener a su vez subbloques y ,por otro lado, se encuentran los lenguajes OO como Omola.

- Objetos de datos de control (CDO)

Algunos métodos de análisis y diseño de sistemas de control usan y generan datos de tipos como ecuaciones de estado, funciones de transferencia, señales en tiempo y frecuencia (éstas dos últimas no pueden ser representadas fácilmente con las dos capas anteriores), etc, sin importarles los detalles del modelo del sistema. Si solo se tuvieran las dos capas descritas anteriormente, las herramientas que manipulan estos tipos de datos tendrían que tener traductores entre los modelos que se manejan en las otras dos capas y estos tipos de datos. Para centralizar este tipo de traducciones en un solo punto, se recomienda tener una capa de objetos de datos de control.

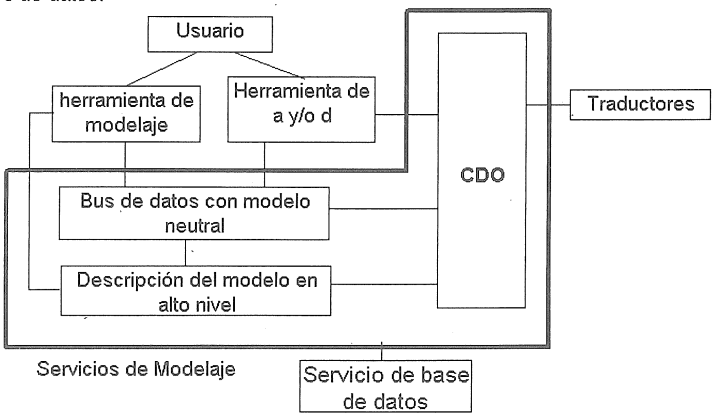
En las aplicaciones estudiadas hay dos formas de implementar esta capa: la primera es definir TADs por cada objeto y la otra forma es crear clases en el esquema de clases de la base de datos OO (si esta disponible).

Adicionalmente, al igual que en las herramientas CASE, se debe proveer un mecanismo de versionamiento de los modelos para ayudar al refinamiento del mismo por parte del diseñador.

Las tres capas se pueden ver en la siguiente figura (tomada de [BAR93]).

5.2. Servicios de manejo de base de datos

Todo el versionamiento de modelos, diseños, etc. deberían guardarse en bases de datos que puedan ser accedidas en cualquier momento por el (los) diseñador(es), por ello es recomendable apoyarse en un manejador de bases de datos.



5.3. Servicios informáticos

Existen otros servicios que debe ofrecer una herramienta CACE que dependen más de la arquitectura informática de las aplicaciones que de los servicios CAD del paquete. Sin embargo, estos deben ser tomados en cuenta al implementar una herramienta de este tipo. A continuación se describen algunas características de ellos:

5.3.1. Servicios de manejo de tareas

Debido a que se tienen diferentes herramientas en el ambiente, la aplicación debe tener un manejador de tareas cuya función es enrutar los comandos que le llegan a las herramientas a las que les corresponde ejecutarlas. De esta forma, este manejador provee la integración de las diferentes herramientas con el ambiente de la aplicación.

5.3.2. Servicios de interfaz con el usuario

En cuanto a interfaz se refiere, debe cumplir los requerimientos dados en el apartado 4. Cada herramienta puede tener su propia interfaz pero deberían ser similares para no confundir al diseñador.

5.3.3. Servicios de comunicación entre aplicaciones

Estos servicios permiten comunicar a todas las herramientas y el ambiente entre sí. Usar modelos estándares como el OSI facilita la distribución de la aplicación en una red de computadores. De esta forma el manejador de tareas se apoyaría en esta capa dando mayor independencia a las herramientas.

5.4. Herramientas

Estas herramientas deberían ser abiertas en el mismo sentido que se ha venido discutiendo para integrarlas fácilmente entre sí y con el ambiente de la aplicación. Entre los tipos de herramientas que se dispondrían estarían:

5.4.1. Herramientas de simulación

Estas herramientas, en la mayoría de casos necesitan conocer el modelo del sistema y/o el modelo del control además de las condiciones iniciales del sistema y los parámetros específicos del simulador. También pueden tenerse simuladores que calculen diagramas típicos de sistemas como el cálculo del diagrama de Nichols.

5.4.2. Herramientas de métodos específicos de diseño

Las herramientas de este tipo ayudan al ingeniero a sintetizar compensadores usando diferentes estrategias y métodos. Deberían haber herramientas de cada uno de los tipos de diseños estudiados en el apartado 3 para el diseño de controladores.

5.4.3. Herramientas de análisis numérico

Aquí están muchos de los métodos de análisis de todos los tipos de sistemas.

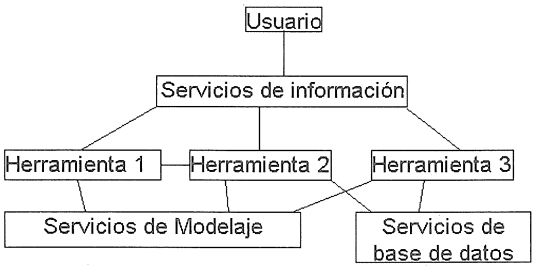
5.4.4. Herramientas de manipulación simbólica

Las herramientas de este tipo ayudan al análisis de sistemas no lineales, sistemas lineales (e.g. cálculo de matrices inversas en función de la frecuencia) y sistemas discretos y proveen soporte a sistemas expertos.

5.4.5. Herramientas de documentación

Las herramientas de este tipo ayudan al ingeniero a mantener información sobre los proyectos, además de ayudarle a generar reportes, resultados, etc.

De esta forma, la estructura de una aplicación CACE sería como la que se muestra en la figura:



6. Estudio de productos CACSD

En [JAM92] se dividen los sistemas CACSD en dos grandes bloques: los basados en MATLAB y los que no se basan en él.

6.1. Sistemas basados en MATLAB

En los sistemas basados en MATLAB como MATRIXx, CTRL-C, SCILAB y el mismo PC-MATLAB; la ayuda al modelaje de la planta es mínimo puesto que el ingeniero de control debe convertir el modelo de su planta al lenguaje matricial que entienden estos paquetes. Esto tiene la ventaja de que la herramienta maneja solo un tipo de

dato, las matrices complejas, y así desde el punto de vista de la aplicación, es más fácil dar más ayuda en los otros dos campos que debe cubrir (análisis y diseño). Sin embargo tiene las siguientes desventajas:

- al cambiar de espacio de representación (al pasar del modelo dado por el modelador al modelo equivalente matricial) se puede perder la semántica del modelo, además de que dependiendo del modelo original, la transformación al modelo matricial puede no ser exacto y así perder información del proceso.
- el ingeniero debe darle una interpretación a los datos del paquete ajustados a su problema (e.g. los vectores en ciertos contextos pueden referirse a polinomios y en otros casos si se refieren a vectores).
- el mantenimiento del modelo puede ser engorroso.
- es muy difícil reutilizar el modelo en otros problemas similares.
- la definición de bloques no lineales no es muy sencillo.
- no es fácil manejar las técnicas de diseño que se basan en la representación del sistema en el dominio de la frecuencia.

Hay algunas herramientas basadas en MATLAB que lo usan como motor de procesamiento numérico pero que permiten hacer modelaje por diagramas de bloques de forma amigable al usuario y algunos tienen una biblioteca de bloques genéricos en los cuales sólo se ajustan unos cuantos parámetros para definirlo (e.g. SIMULINK).

6.2. Sistemas no basados en MATLAB

Los demás CACSD tienen su propio lenguaje de especificación del modelo, como es el caso de Omsim [MATT95], o son más un ambiente de integración de herramientas externas de forma que el módulo de modelaje puede ser externo, como es en cierta forma ANDECS [GRÜ95].

6.2.1. CADACS (Computer aided design and analysis of control systems) [CAD95]

CADACS es un producto desarrollado en la Universidad Ruhr Bochum de Alemania. Es un sistema abierto en el sentido en que el usuario puede diseñar sus algoritmos e incluirlos en la biblioteca de funciones del sistema. Posee el servicio de base de datos en donde se guardan los modelos de la planta y todos los datos necesarios para trabajar en el problema de control (no maneja versionamiento). Tiene un módulo de tiempo real para el cual tiene un control adaptativo y un extractor de señales que es útil para la síntesis de modelos.

De los sistemas estudiados, es el que más facilidades ofrece para el modelaje: ofrece 10 tipos de representación y tiene una amplia biblioteca de utilidades de extracción de módelos que usan aproximaciones y estimación de parámetros. También provee algunas utilidades de diseño tanto del control como de observadores.

Ofrece herramientas básicas de procesamiento como funciones de manejo y análisis de señales, funciones de transferencia, respuestas de frecuencia, variables de estado y matrices de fracciones polinomiales.

Una de sus desventajas es que el lenguaje que usa es bastante críptico y dificulta la interacción con el usuario. El lenguaje es interactivo e interpretado.

6.2.2. Omsim [MATT95][AND95]

Omsim es el simulador del lenguaje orientado por objetos de modelaje de sistemas dinámicos "Omola". Ambos fueron desarrollados en el instituto de tecnología de Lund en Suecia. Omola es orientado a bloques de control, i.e. la unidad básica de modelaje es un bloque de control.

Omola es un lenguaje orientado por objetos de tal forma que conceptos como herencia, polimorfismo, etc. se encuentran disponibles en su sintaxis.

En Omola es posible definir clases de bloques. Los atributos de estos pueden ser de varios tipos:
- objetos de otras clases (estos atributos se llaman componentes)
- variables del bloque (posiblemente para ser usadas como parámetros)
- ecuaciones que describen el comportamiento del bloque.
- conexiones entre terminales (entradas o salidas) del bloque y sus subbloques
- definición de eventos (discretos)

En Omola el comportamiento de un bloque de control no está dado solamente por las ecuaciones que lo modelan, sino que además se deben añadir una serie de restricciones (que no hacen parte del modelo) para saber cuales variables de las ecuaciones se deben calcular y cuales están dadas. Así por ejemplo, la ecuación que modela totalmente el movimiento de un cuerpo es:

$$m\, d^2x/dt^2 = F$$

sin embargo, si se quiere simular el bloque correspondiente, hay que agregar dos restricciones (usando herencia por especialización) para despejar la otra . Por ejemplo: m=mo y x=xr(t) y al simular se despejaría la fuerza. Adicionalmente se permite definir las variables de entrada y salida cuando se están creando las clases contendedoras de la clase en cuestión.

Sin embargo ellos mismos admiten que estas dos facilidades del lenguaje son difíciles de resolver para el caso general puesto que implica la reorganización de las ecuaciones para poder resolverlas (si es posible) y esto a su vez implica, para efectos prácticos, la limitación del tipo de ecuaciones permitidas en el lenguaje. En Omsim implementan estos conceptos con algunas restricciones por las razones dadas (i.e. no funciona para cualquier ecuación en la versión actual). tiene la restricción de que las ecuaciones que modelan el bloque deben ser diferenciales algebráicas.

Uno de los problemas que se presentan es la necesidad de definir valores iniciales para efectos de simulación, pero en muchos casos algunos de estos valores no están disponibles fácilmente al usuario. Omsim tiene el lenguaje OCL (Omola command language) para hacer guiones para procesamiento de una secuencia de comandos de Omsim en batch. No tiene servicio de base de datos y no es abierto en todos los servicios de la discusión hecha arriba.

6.2.3. CAMeL (Computer aided mechatronic laboratory) [RUT95]

CAMeL es un ambiente basado en procesos para sistemas de control. Es un CACSD abierto en el sentido descrito arriba. Fue desarrollado en la Universidad de Paderborn de Alemania.

Tiene un lenguaje de modelaje llamado DSL (dynamic system language) que pretende ser lo más genérico posible en el sentido de que pueda modelar prácticamente cualquier sistema. Además DSL tiene facilidades de integración. Este lenguaje es orientado a bloques de control, i.e. la unidad básica de modelaje es un bloque de control.

DSL no ofrece un lenguaje orientado por objetos sino que simplemente para cada tipo de bloque define una serie de parámetros, una serie de entradas y salidas (aquí si se diferencian), una serie de variables de estado y un conjunto de ecuaciones de estado que definen el comportamiento del bloque. Usando DSL se pueden definir bloques que estan compuestos por otros definiendo la forma de interconexión entre ellos.

Solo permite que las ecuaciones que modelan el bloque sean diferenciales y/o algebraicas (en la forma de ecuaciones de estado).

Ofrece interfaces de comunicación con otros paquetes CACSD como MATLAB y otros. Para interactuar con las herramientas externas de control, ofrece dos formas de organizar los procesos de diseño: una en la cual el usuario define un pseudocódigo para definir procesos batch y la otra forma que ofrece es ejecutar cada acción en un proceso diferente y definir la sincronización entre ellos. Esta última forma ayuda a la distribución del producto y al uso de máquinas paralelas (usa PVM para ello). Adicionalmente, posee un modulo de simulación de ecuaciones no lineales llamado TRANSIENT. No tiene servicio de base de datos.

6.2.4. ANDECS (Analysis and design of controlled systems) [GRÜ95]

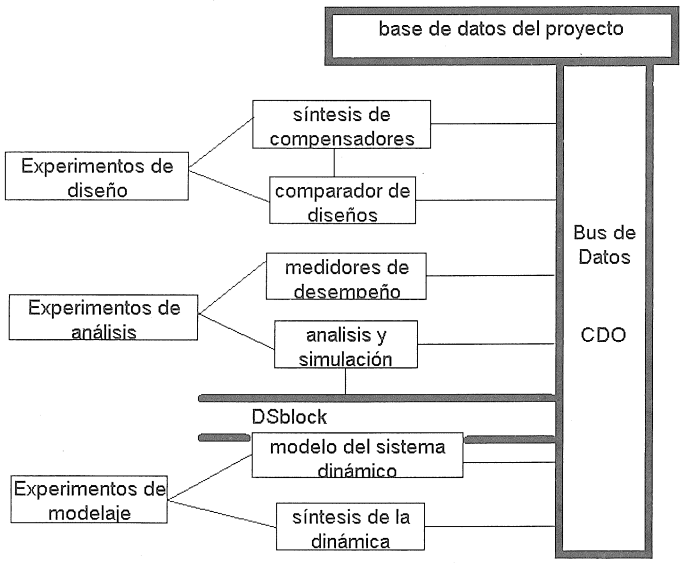
ANDECS es un ambiente basado en procesos para sistemas de control. Es un CACSD abierto en el sentido descrito en las primeras secciones. Fue desarrollado en el instituto de robótica y sistemas dinámicos DLR de Alemania.

Posee una base de datos llamada RSYST que permite la persistencia de datos complejos. También maneja un sistema muy básico de versionamiento. La comunicación entre las herramientas se hace por medio de la base de datos únicamente. Se usa PVM para la comunicación entre procesos.

Para definir modelos la idea es que el usuario lo diseñe en el paquete de su predilección y luego lo convierta a objetos de datos de control (CDOs). Estos CDOs son los que van en la base de datos. A otro nivel se tiene DSblock (Dynamic system block) [OTT92] que es un protocolo estándar para comunicar los métodos numéricos genéricos con la definición de los modelos. En este punto es que se encuentran los conversores a otros dominios de modelaje porque es en cierta forma un modelo computacional neutral (de hecho se tiene un traductor a Dymola, el antecesor de Omola).

Posee dos vías de acceso a la biblioteca de funciones numéricas básicas: una es la biblioteca RASP y por otra parte tiene un interpretador de MATLAB llamado Amatlab (ANDECS_MATLAB) por donde se puede tener acceso a ellas. Además, Tiene un módulo llamado MOPS (multi objective programming system) que permite al usuario crear archivos de procesamiento en batch (con posible interacción con el usuario) para hacer todo el proceso de diseño de los compensadores siguiendo el método que se describió en la segunda sección.

La estructura global del producto se presenta en la siguiente figura.



7. Análisis de los productos

7.1. Lenguajes de modelaje

Cada uno de los lenguajes estudiados tiene ventajas sobre el otro. Por ejemplo la mayor ventaja de Omola es su orientación por objetos que permite la reutilización de los modelos más fácilmente.

El problema de Omola es que al pretender dar la mayor cantidad de genericidad al modelo (para aprovechar al máximo las ventajas de un lenguaje OO) se complica un poco la comprensión del modelo, por ello sería probablemente más útil exigir más restricciones en la definición de las ecuaciones del bloque y definir totalmente las entradas y las salidas (como lo hace DSL).

DSL es un lenguaje un poco más natural (para el ingeniero de control) por cuanto es más orientado a bloques de control (que es lo que se maneja con más frecuencia en sistemas de control). Sin embargo, los modelos dados en

DSL son un poco más difíciles de reutilizar que los de Omola porque estos últimos se manejan como objetos ofreciendo mayores facilidades en la reutilización.

En cuanto a las ecuaciones de Omola, sería mejor tener la capacidad de definir métodos (quizás un método por salida) que describan el comportamiento del bloque. DSL es relativamente pobre en cuanto a las ecuaciones que permite (solo modelaje con variables de estado) y el problema es que dado que esto va en la definición del bloque, sería necesario extender el lenguaje para ello con el inconveniente de que las no linealidades no tienen una forma común de definición. Por ello el comportamiento debería estar fuera de la definición del bloques organizado por funciones que calculen cada una de las salidas del bloque y así no sería necesario diferenciar los bloques que definen bloques básicos de los que definen la interconexión de un conjunto de subbloques.

Por otra parte, en los lenguajes de modelaje debería poderse distinguir los bloques de control de los bloques de la planta y poder separarlos de forma que el cambio de estrategia de control sea mas fácil y así poder facilitar la labor de diseño asistido. A su vez, se debería poder definir esquemas de control jerárquico y descentralizado para facilitar el diseño.

7.2. Productos

Tanto CAMeL como ANDECS tienen el mismo enfoque en el sentido de crear un ambiente donde se integren herramientas externas ofreciendo servicios básicos para ello. Al estudiarlos, se puede decir que ANDECS tiene las ideas más maduradas (e.g. tiene manejo de bases de datos) aunque en algunas características, CAMeL ha profundizado la investigación (e.g. CAMeL ha hecho mayores avances en la programación de tareas puesto que permite la programación en paralelo de tareas).

Por el contrario, Omsim pretende ser una herramienta de modelaje y simulación que podría incluirse en los dos primeros y es abierto en el sentido de ofrecer modelos genéricos que podrían integrarse con otros paquetes (como ocurre en ANDECS) hechos en Omola.

Por otra parte, CADACS pretende ofrecer la mayor funcionalidad posible dentro de él mismo, permitiendo además la programación de nuevos algoritmos por parte del usuario. En este sentido es menos abierto que los dos primeros, pero el hecho de tener un buen número de algoritmos y servicios lo hace, en cierta medida, más fácil de manejar. Adicionalmente tiene el módulo de tiempo real que puede ser muy útil tanto para la captura de señales como para la implementación de ciertos controladores, además de tener la posibilidad de crecer a sistemas supervisores de procesos.

A continuación se muestra una tabla comparativa de los productos estudiados bajo los parámetros estudiados en las primeras secciones del documento. También se muestra que el único paquete con módulo de tiempo real es CADACS que puede llegar a ser interesante en la implementación de controles por computador.

Criterio	Basados en Matlab	CADACS	Omsim	CAMeL	ANDECS
1. Servicios de Modelaje					
Modelo neutral	si para sistemas lineales	si	si	si	si
Lenguaje para modelo neutral	Matlab	no	no	no	si
Lenguaje de alto nivel	la mayoría ofrecen agrupamiento de bloques	no	si	si	externo
CDOs	si usando convenciones sobre las variables de Matlab	no	si	si	si
2. Servicios de manejo de bases de datos	no	sin versionamiento	no	no	si
3. Servicios informáticos					
Manejo de tareas	usando Matlab	no	si	si	si
Interfaz con el usuario	si	si	si	si	si
comunicacion entre aplicaciones	no	baja	no	si	si
4. Herramientas					
Simulación	si	si	si	si	si
Análisis numérico	si	si	no	externas	externas
manipulación simbólica	no	no	no	externas	externas
métodos específicos de diseño e identificación.	si	si	no	externas	externas
documentación	no	si	no	externas	externas
5. Facilidades de tiempo real					
Monitoreo en tiempo real	no	si	no	no	no
Control adaptivo en tiempo real	no	si	no	no	no

8. Conclusiones

Como conclusiones podemos decir que un sistema de apoyo al diseño de sistemas de control debería tener los servicios que se describieron en los apartados 4 y 5 para apoyar el proceso de diseño descrito en el apratado 2. Debería ofrecer métodos de diseño para todos los tipos de sistemas (SISO, MIMO y alto orden) de todas las clases de métodos (de síntesis, basados en procedimientos y basados en búsqueda). Como se pudo ver, en la actualidad,

con muy pocas excepciones se tiene una herramienta con todas esas características y por el contrario hay herramientas que concentran su capacidad en un determinado tipo de servicios.

El corazón de la aplicación es el manejador de servicios de modelaje, por ello, éste podría ser el primer problema que se podría resolver al diseñar un sistema CACSD. Otro campo por el que se podría comenzar es en la capa de servicios informáticos para crear el ambiente de trabajo de la herramienta CACSD, aunque algunas partes de ella requiere tener conocimiento sobre la interacción requerida entre herramientas y la comunicación con los servicios de modelaje. También se pueden crear herramientas abiertas que se puedan integrar fácilmente en el ambiente CACSD. Para ello, se puede explorar la posibilidad de montar todos estos cálculos sobre plataformas tipo Matlab (que manipulan las matrices de una forma muy eficiente), por ejemplo usando el paquete Scilab.

Por último, se puede pensar en como integrar los módulos de tiempo real (e.g. implementación de controles y control adaptivo) al modelo presentado (que básicamente es el que se muestra en [BAR93]) y crear el esquema de comunicación del computador con la planta.

9. Referencias

[AND95] Andersson, M. "Omsim and Omola tutorial and user's manual". Instituto de tecnología de Lund, Marzo 1995(disponible via ftp anónimo).

[BAR93] Barker, A. et al. "Open arqchitecture for computer-aided control engineering". IEEE control systems, Abril 1993.

[CAD95] "CADACS - a system for computer-aided design and analysis of control systems". Univesidad Ruhr Bochum, 1995 (disponible via ftp anónimo).

[GRÜ95] Grübel, G. "The ANDECS CACE framework". IEEE control systems, Abril 1995

[HER94] Hernández, J. Morales, J. "PICAS: un CAD de propósito específico basado en rasgos para diseño de piezas quasi-axisimétricas". Revista Sistemas de ACIS, Enero-Marzo 1994.

[JAM92] Jamshidi, M. Mahmoud, T. Bahram, S. "Computer Aided analysis and design of linear control systems". Prentice Hall. 1992.

[Mac89] MacFarlane, A. Grübel, G. "Future design environments for control engineering". Automatica, Vol 25. 1989.

[MAT95] Mattsson, S. Andersson, M. "Ideas behind Omola". Instituto de tecnología de Lund, 1995 (disponible via ftp anónimo).

[NG93] Ng, W. "Perspectives on search-based computer-aided control systems design". IEEE control systems, Abril 1993.

[OTT92] Otter, M. "DSblock: A neutral description of dynamic systems". Open CACSD electronic newsletter, Mayo 1992 (disponible via ftp anónimo).

[RUT95] Rutz, R. Richert, J. "CAMeL: an open CACSD environment". IEEE control systems, Abril 1995