Sistemas Embebidos: Simplificación de Descripción de Máquinas de Estado Finito

Eduardo D. Cohen, Esteban D. Volentini, Juan Pablo Gruer
Facultad de Ciencias Exactas y Tecnología, Universidad Nacional de Tucumán, Tucumán, Argentina.

Resumen

Este artículo propone herramientas para mejorar la descripción y simplificación del modelo de funcionamiento de sistemas embebidos por medio de máquinas de estado finito (MEF) . Se basa en determinadas propiedades que se encuentran en la mayoría de ellos. Los autómatas responden frecuentemente a un gran número de entradas, sin embargo, en general muy pocas de ellas se tienen en cuenta en cada estado para generar una reacción del sistema. Por otra parte, es frecuente que un sistema responda de manera similar, no solo a una entrada, sino a un grupo de ellas desde varios estados, por ejemplo en el caso de múltiples sensores que podrían desencadenar idéntica reacción. También se ha notado que muchas veces los sistemas pasan por ciertas secuencias de estados, permaneciendo muy poco tiempo en ellos. Un buen aprovechamiento de estas características permite jerarquizar la descripción de una MEF mediante un diagrama de estado más simple, en el contexto de la metodología de "dividir para conquistar".

Palabras clave: máquinas de estado finito, métodos de modelado, sistemas embebidos.

Embedded Systems: Simplification of Finite State Machines Description

Abstract

This work proposes some tools to describe and simplify Finite State Automata (FSA) that represent the behavior of an embedded system. This approach is based on typical features of this kind of systems, i.e. in a given state of a FSA only a few among a large set of system inputs are taken into account. In addition to this, usually the system presents identical reaction to many inputs in different states. Many systems present transitions through certain sequence of states staying only a short time on them. It is possible to take advantage of these features, allowing a hierarchical description of a FSA by means of a simpler state diagram. This approach belongs to the "divide and conquer" strategy.

Keywords: *finite state automata, modeling methods, embedded systems.*

Introducción

Con justa razón se dice que escondida en una buena descripción se encuentra la solución al diseño del sistema. Por esta causa resultará muy útil todo esfuerzo que se invierta en lograr un modelo simple que contenga la mayor precisión y detalle posible en la especificación.

La descripción debe expresar con total claridad y sin ambivalencias el comportamiento final del sistema a diseñar, por ello se dice que *se comienza desde el final*.

Cuando se comienza desde el final, en general se parte de unos pocos estados y al refinar la descripción, se van agregando estados y entradas adicionales, conduciendo así a un diagrama cada vez más complejo. Se termina así en una representación en la que lo fundamental está escondido por los detalles. La cantidad de componentes significativos – estados en este caso – que un diagrama debería contener para poder ser analizado apropiadamente debería ser reducido, **Maier and Rechtin (**2000) y **Keating** (2011).

Se postula la existencia de características comunes a la mayoría de los sistemas, que permitan proponer una simplificación en su metodología de descripción. Para ello se propone considerar los siguientes argumentos para aplicar el criterio de "dividir para conquistar":

a) Un sistema puede contener un número muy grande de estados, sin embargo, por lo general, una gran cantidad de ellos se activan con escasa frecuencia y representan una proporción baja del tiempo total de funcionamiento del mismo. Esta característica debería permitir dividir el análisis

en estados transitorios y estados estables. En general la funcionalidad de estos dos tipos de estado es diferente, con los estados transitorios más relacionados a transiciones complejas y poco frecuentes, en los que la seguridad y confiabilidad del sistema esté más comprometida. Por otro lado, los estados permanentes representan la respuesta del sistema a condiciones normales, responsables de la mayor parte de las especificaciones de un usuario. En consecuencia, sería importante proveer mecanismos de descripción diferenciados que permitan concentrarse en la importancia particular de ambos casos.

b) La cantidad de entradas al sistema podría ser grande, sin embargo, muchos estados solo consideran un conjunto reducido de estas entradas, siendo el resto indiferentes a los mismos. Estas condiciones de indiferencia deberían permitir mayor simplicidad en la descripción y realización del sistema.

c) En general, existen conjuntos de entradas para la cual el sistema presenta idéntica salida. Si bien el conjunto de estas entradas puede variar de estado en estado, la reacción del sistema es igual para todas ellas. Tómese como ejemplo un sistema con sensores de alerta que provocan una condición de alarma. En distintos estados de funcionamiento del sistema, el conjunto de sensores podría cambiar, pero no así la condición de alerta, que constituye la reacción de salida del sistema.

Para una mayor claridad en la descripción de la metodología, se empleará el caso de un sistema de alarmas domiciliario, simplificado, que cumple con las condiciones previamente descriptas.

Caso de estudio: sistema de alarma domiciliario

Un sistema de alarmas, del tipo domiciliario, podría describirse mediante un diagrama de estados muy general que contenga 3 estados generales: inactivo, desarmado y armado.

Si se perfecciona un poco la especificación, sería necesario agregar distintos modos de armado a la lista anterior:

a) *Armado presente: personas permanecen en el interior del recinto para lo cual se requiere que determinados sensores sean ignorados por el sistema.*

b) *Armado ausente: todos los sensores activados.*

c) *Armado presente forzado*: todos los sensores activados, excepto algunos que presentan desperfectos.

d) *Armado Ausente forzado*: similar al caso previo, desactivando sensores con desperfectos que no deben generar reacción en el sistema.

e) *Armado temporario*: algunos sensores se activan al cabo de un tiempo, para poder salir del domicilio, mientras que otros no. Este estado es un estado intermedio para pasar al armado ausente, o armado forzado ausente.

f) *Desarmado temporario:* similar al caso anterior, para ingresar al domicilio.

Considerando que para realizar una transición de un estado a otro, por ejemplo desde el estado “armado presente” al estado “desarmado” es necesario oprimir una secuencia de teclas (ENTER + 5 teclas numéricas), un detalle más fino del diagrama de estados mostraría seis estados intermedios para desarrollar esta actividad.

Es claro que la cantidad de estados crece rápidamente a medida que se avanza en el detalle de la descripción. El diagrama final resultaría, así, muy engorroso.

En muchos casos el diagrama de estados exige armar “secuencias” de estados para desarrollar cada actividad (por ejemplo, el ingreso de una clave, como se describiera previamente). Ello podría inducir al pensamiento secuencial en el diseño del sistema, lo cual es muy propenso a errores: concentrarse solo en la secuencia de eventos fundamentales para llevar a cabo una actividad, olvidando tener en cuenta la posible influencia de otros eventos que pudieran ocurrir mientras dure la misma, induciendo así a un diseño incorrecto.

Ejemplificando un error típico del pensamiento secuencial: si al desarmar el sistema de alarmas solo se tuviera en cuenta las seis teclas a ingresar para pasar al estado desarmado, bastaría que un intruso oprima una tecla para dejar al sistema detenido en espera de la siguiente tecla, quedando el mismo sin reacción frente a la activación de sensores.

De lo anterior surge que no solo es necesario agregar una multiplicidad de estados, sino que en cada uno de ellos debe analizarse la respuesta a todos las entradas posibles, con lo cual el diagrama resulta aún más complejo.

En consecuencia resulta necesario generar herramientas y definiciones, con un grado más alto de abstracción, tales que simplifiquen la descripción del sistema que se desee realizar.

Definiciones preliminares

Estímulo y Reacción

Se define como *estímulo* a toda combinación en las variables de entrada que provoque una *reacción* en uno o más estados del sistema. La reacción puede consistir en al menos una de las acciones siguientes: (a) variación en la salida del sistema, que se denominará *reacción externa* y (b) transición a otro estado, *reacción interna*. Cabe aclarar que un mismo estímulo puede generar reacciones distintas en distintos estados del sistema.

Una combinación de las variables de entradas al sistema no necesariamente constituye un estímulo en un estado dado: puede no generar una reacción.

A modo de ejemplo, en el caso del sistema de alarmas que se encuentra en "estado armado", un sensor de intrusión que se activa provoca el sonido de las sirenas y por tanto es un estímulo. Si en esta situación, el sensor se desactiva (un intruso cierra una puerta luego de haber ingresado), el sistema podría no producir ninguna reacción, por lo que este cambio en la entrada no sería un estímulo.

Nótese que en el caso precedente, la presencia del estímulo produce una reacción externa, pero no así una reacción interna.

Asimismo, el ingreso de una misma clave de activación/desactivación producirá una reacción diferente del sistema para el caso de que el mismo estuviera armado o desarmado.

Estados Temporarios y Permanentes

Se denomina *estado temporario* a todo aquel en que el sistema permanece solo por un tiempo cuya duración es corta y acotada. Una señal de "time-out" es una forma típica de acotar el tiempo de duración de un estado temporario.

Un *estado permanente*, por el contrario, es aquel en el cual el sistema puede ubicarse por un lapso de tiempo no acotado. Los estados temporarios pueden concebirse como parte de un camino intermedio hacia un estado estable.

Actividades

Se define una *actividad* como la secuencia de estímulos, estados temporarios y reacciones externas del sistema que le permite realizar una transición entre dos estados permanentes: el estado origen y el estado destino.

Dado que la actividad está compuesta por una secuencia de estados temporarios, su duración es acotada. Para mayor generalidad, la actividad incluye implícitamente la transición de retorno al estado origen en caso de que el lapso de tiempo, desde que comenzara, haya superado el límite establecido para la misma.

Considérese el caso en que el sistema de alarmas, descripto previamente, se encontrara en el "estado armado presente". Como ejemplo de actividad podría considerarse "desarmar", que consiste en el ingreso de una clave para permitir realizar una transición del estado "armado presente" a "estado desarmado". Si la clave se ingresara de manera incompleta el sistema retornaría al estado origen ("armado presente") una vez transcurrido el tiempo máximo establecido para esta actividad.

En función del número de estímulos, es posible clasificar las actividades en:

a) Atómicas. Son actividades compuestas por un único estímulo y por tanto no pueden ser interferidas por otra actividad previo al cambio de estado. Un ejemplo en el sistema de alarmas: la activación de un sensor de intrusión cuando el sistema está armado produce el disparo de la alarma. Estas actividades son de muy corta duración en relación a los tiempos que transcurren entre estímulos.

b) No atómicas. Se trata de actividades compuestas por una sucesión de estímulos y sus correspondientes reacciones. Las actividades no atómicas podrían ser exclusivas o no. Actividades exclusivas, al igual que las atómicas, no pueden ser interferidas por otra actividad. Por el contrario, las actividades no exclusivas podrían interactuar con otras actividades, previo a completar el cambio de estado. Teclear los dígitos que componen una clave es un ejemplo de actividad no atómica. Es claro que esta actividad debe ser no exclusiva para evitar justamente el error (pensa-

miento secuencial) descripto en el caso de estudio.

Reducción de estados temporarios mediante actividades

Considérese el caso de una transición del sistema entre dos estados permanentes: S_o y S_n, a través de una secuencia de n-1 estados intermedios temporarios, como se ilustra en Fig. 1.

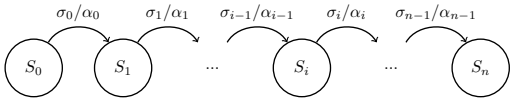

Fig. 1. Secuencia de estados temporarios entre dos estados permanentes.

El sistema pasa desde S_0 a S_n a través de los estados temporarios (S_1, S_2, …, S_{n-1}) merced a una secuencia de estímulos (σ_o; …; σ_{n-1}), generando las salidas (a_0; a_1; …; a_{n-1}). Esta secuencia puede ser simplificada mediante su reemplazo por una actividad, A, que consiste justamente en la secuencia completa de dichos estímulos, dando como resultado el diagrama simplificado de Fig. 2.

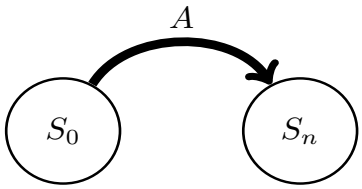

Fig. 2. Transición mediante la actividad A.

Cuando la actividad no es exclusiva, entonces existe al menos un estado intermedio desde el cual se puede pasar a otro fuera de aquellos reemplazados por la secuencia A, mediante una actividad distinta, B, como se ve en Fig. 3.

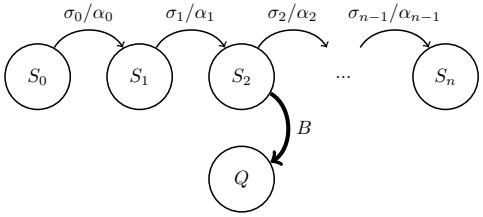

Fig. 3. Actividad no exclusiva.

En este caso podría producirse alguna de las siguientes alternativas:

a) Para llegar al estado final, S_n, será necesario retornar al estado S_0 y esperar a que se produzca nuevamente la secuencia de todos los estímulos que configuran la actividad A. En este caso se dirá que B *aborta* a A.

b) Se realiza la secuencia de estados que comienza con la actividad B y posteriormente se retorna al estado en que se encontraba la secuencia "A", al momento de llegar el primer estímulo de B, S_2 en el ejemplo de Fig. 3, en este caso se dice que B *coexiste* con A.

Los casos anteriores se pueden representar también con cartas de estados, **Harel** (1987), que justamente tienen por objetivo mejorar el clásico diagrama de estados permitiendo jerarquizar su descripción. Así una secuencia de estados podría representarse con un "superestado". A diferencia de la representación citada, la del presente artículo no requiere el uso de herramientas profesionales de trabajo.

El uso de actividades es más simple ya que deja la interacción con otras actividades para un paso posterior en la estrategia de "dividir para conquistar". Por otra parte, en un sistema de lógica programada, las actividades se prestan para su realización mediante hilos de programa. La solución a problemas de interacción entre tareas han sido analizadas y propuestas en el campo de sistemas de tiempo real, por ej. **Laplante and Ovaska** (2012). En un próximo artículo, los autores presentarán una propuesta innovadora para modelar y determinar los tipos de interacción entre tareas.

Unificación de estados

Cada estado debe su existencia a que describe un comportamiento diferencial del sistema en relación a los estímulos. En efecto, cuando el sistema pasa de un estado a otro, algunos estímulos presentes en el estado anterior podrían no provocar ninguna reacción en el nuevo estado, y por tanto no estar representados en el mismo. Lo opuesto también es válido: pueden aparecer estímulos en el nuevo estado que en el estado previo eran inexistentes. Resulta, por tanto, que existe una percepción diferencial de estímulos en cada estado del sistema.

Si cada *actividad* se encargara de modificar la percepción del sistema a los estímulos y cada estímulo percibido provocara, en un determinado grupo de estados, S, idéntica salida del sistema, podría obtenerse un diagrama de estados más simple, lo que se muestra a continuación.

Sea un sistema en el que existe un subconjunto, S, de estados tales que: (a) la

reacción externa al mismo estímulo percibido sea idéntica y (b) cualquier transición debida a idéntico estímulo en S puede conducir a un único estímulo E que no pertenezca a S, entonces los estados de S se pueden unificar en un solo estado, U, dando por resultado una simplificación del diagrama de estados.

Nótese que los estímulo puede o no percibirse desde distintos estados de S, pero en caso de ser percibidos deben cumplir con las condiciones del párrafo anterior.

La prueba es inmediata ya que al unificar los estados de S en un único estado U, las transiciones entre ellos solo pueden darse de la siguiente manera:

a) Autolazos en U que cambian la percepción a los estímulos, equivalen a transiciones entre estados de S.

b) Transiciones de salida desde estados de S, a estados externos a este subconjunto, ocasionadas por el mismo estímulo, producen, por definición, idéntica reacción del sistema. En consecuencia conducen al mismo estado externo E, con igual reacción externa, por lo que pueden ser reemplazadas por una única transición desde U a E.

c) Transiciones de entrada a cualquier estado de S, originadas por un mismo estímulo desde un único estado externo W, pueden ser reemplazadas por una única transición de W a U, ya que las siguientes reacciones serán idénticas, de acuerdo a los dos puntos anteriores..

Dado que cada estímulo provoca igual reacción del sistema en los estados de S, también lo harán en U, con lo cual el sistema no varía su comportamiento al reemplazar S por el único estado U.

Obsérvese que la simplificación permite que los estados en S se unifiquen en U incluso si no reaccionan a la misma secuencia de entradas, y por tanto no requiere relaciones de equivalencia entre estados, como es el caso de algoritmos conocidos, **Hopcroft** (1971).

Cambios a la percepción del sistema

Para implementar cambios a la percepción del sistema cuando ocurren transiciones en su diagrama de estado, es suficiente con realizar una operación de enmascaramiento de estímulos.

Tómese como ejemplo el caso de hasta 64 posibles estímulos, organizados en una matriz, ME, de 8x8, **Cohen et al.** (2015). Cada posición de un bit en esta matriz representa con un “1” o un cero la presencia o ausencia del estímulo correspondiente.

Se define a la matriz de percepción, MPi,j a aquella que la actividad Ai imponga sobre el sistema en el estado j, tal que contenga un “1” la posición que tiene en ME el estímulo que debe ser percibido y “0” en caso contrario.

La matriz de estímulos percibidos por el sistema en el estado Sj al finalizar la Ai , MEP i,j , se obtiene mediante una simple operación de enmascaramiento, dado por la siguiente ecuación lógica.

$$MEP_{i,j} = ME \wedge MP_{i,j} \qquad (1)$$

El sistema de entrada de datos se encarga de mantener actualizada a ME y a $MEP_{i,j}$. La última actividad ejecutada determina la matriz de percepción vigente en el sistema en un momento dado.

Es importante destacar que el enfoque para la simplificación de estados es diferente a aquellos relacionados a modelos de autómatas no completamente especificados, **Higuchi and Matsunaga** (1996). Ello se debe a que las matrices de percepción aseguran que las variables de entrada no especificadas para un estado cualquiera, Si, no podrán estar presentes cuando dicho estado esté activo, quedando así totalmente especificado el comportamiento del sistema.

Aplicación al Caso de Estudio

Para ejemplificar un diagrama de estado simplificado por actividades, considérese el sistema de alarmas presentado como caso de estudio. El mismo se representaría mediante un diagrama compuesto por múltiples estados y transiciones. El diagrama de estados simplificado se muestra en Fig. 4 y se describe a continuación.

Inicialmente el sistema no habilita ningún sensor, reacciona solo a las teclas que permitan ingresar una clave determinada, correspondiente a la actividad “*clave especial*”. El sistema se encuentra en el estado *INACTIVO*.

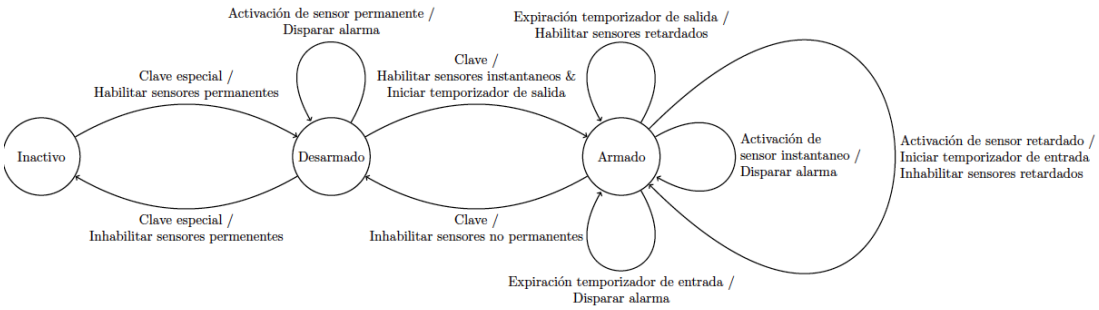

Fig. 4 Diagrama de estados simplificado del sistema de alarmas.

La actividad "*clave especial*" coloca al sistema en el estado "desarmado", adicionando la percepción de los sensores permanentes (fuego y anti-desarme). Un estímulo que provenga de sensores percibidos provocará el encendido de las sirenas. El sistema se encuentra en el estado D*ESARMADO*.

Si se realizara la actividad de "*armado presente*", se habilitaría también la percepción de sensores externos a la casa, permitiendo el desplazamiento de personas en su interior. Nuevamente un estímulo percibido provocará el encendido de las sirenas. El sistema se encuentra en el estado "*ARMADO PRESENTE*".

El lector podrá comprobar, a modo de ejercicio, que el estado "*ARMADO PRESENTE FORZADO*" es similar en todo al estado "*ARMADO PRESENTE*" sólo que en este caso la actividad de transición no habilita los sensores, que por mal funcionamiento o alguna otra razón, permanecen activos. Lo mismo ocurre con el resto de las modalidades para armar el sistema, por lo que es posible unificarlas en un único estado "*ARMADO*". Se construye así el diagrama de estados general de Fig. 4, donde por razones de simplicidad no se exhibe la totalidad de las actividades del sistema.

En todos los casos en que se arma al sistema, un grupo de sensores no deben habilitarse durante un determinado tiempo, a fin de permitir que el usuario pueda salir del domicilio sin que se dispare la alarma. Para ello se establece una matriz de percepción que no incluya a este grupo de sensores "retardados" y se habilita una cuenta de tiempo – TIMER1, pasando al estado *ARMADO*. Una vez TIMER1 llega a su fin se procede a cambiar nuevamente la matriz de percepción del sistema para habilitar estímulos de esos sensores.

Para ejemplificar otras actividades, considérese que el sistema deba armarse en modo ausente: bastará con que se perciban todos los sensores, con excepción de los retardados hasta que concluya la cuenta en timer.

Si bien no está representado en Fig. 4 por razones de simplicidad, cada clave debe ser precedida por la identificación del modo de armado que se va a realizar, lo cual consiste en ingresar una función de armado mediante el ingreso de unas teclas adicionales precediendo a la clave, todo en una misma actividad de armado. Así, la actividad para armar en modo presente requeriría el tecleo de la función "P" y la clave.

En caso de que el sistema se encuentre en el estado *ARMADO*, el usuario debe acceder a la casa a través de un camino de sensores retardados para poder ingresar la clave y desarmar el sistema. No bien se activa el primer sensor retardado, se habilita un segundo timer (TIMER2), que al finalizar disparará la alarma si es que el sistema no pasa previamente al estado *DESARMADO*.

Una versión futura del sistema podría requerir armar una zona particular, Z, de la casa mientras que el resto debe permanecer desarmada. La solución para esta modificación es simple: solo es necesario definir una nueva matriz de percepción que incorpore a los sensores de Z. Se deberá definir una nueva actividad: "*armar zona*" que añadiría esta nueva matriz al sistema cuando el mismo pase al estado *ARMADO*. De este ejemplo se concluye que las herramientas de simplificación contribuyen de manera importante a la

capacidad para realizar cambios incrementales futuros al sistema.

En cuanto a las matrices de percepción, el sistema se inicializa a partir del estado *INACTIVO*, con una matriz de percepción que contiene “0” en toda posición de sensor y “1” en las posiciones correspondientes a los contactos del teclado por el cual se ingresará una clave. En consecuencia, el sistema no reaccionará a estímulos provenientes de ningún sensor.

Al pasar al estado *DESARMADO* mediante la actividad de ingreso de una clave especial, la matriz de percepción agregará un “1” por cada sensor permanente (fuego y anti-desarme).

Las matrices de percepción pueden ser fijas y se conocen al momento de especificar el diseño, tal como se describe precedentemente, o variables, que se determinan durante el funcionamiento del sistema al momento de ejecutarse la actividad que la establece. Tal el caso en que el sistema se debe armar de manera *FORZADA*, pues algunos sensores no funcionan. En ese caso, la actividad de “armado forzado”, deberá determinar qué sensores no funcionan y en función de ello incorporar un 0 en la posición correspondiente a esos estímulos en la matriz de percepción, permaneciendo el resto de la misma idéntica a la que resultaría de una actividad de armado normal del sistema.

Conclusiones

El análisis por actividades es el principal medio para representar de manera simplificada una FSM y ello se debe a las mismas:

a) cambian la percepción del sistema, lo cual permite unificar estados, reduciendo así su número,
b) contienen estados transitorios, reduciendo aún más los estados a representar y
c) proveen un nivel mayor de abstracción. El análisis de interacción entre actividades se deja para un siguiente nivel de detalle, para lo cual se puede emplear cartas de estado y variadas herramientas de software, ya que pueden realizarse como hilos de proceso en lógica programada.

Las consideraciones finales de la revisión del ejemplo del sistema de alarmas, en sección 5, muestra que la aplicación de las herramientas de simplificación, presentadas en este artículo, facilitan modificaciones incrementales futuras. Se postula que diversos sistemas podrían beneficiarse de igual manera.

Las actividades reemplazan a los estados temporarios y las transiciones entre ellos. En la mayoría de los sistemas embebidos, la funcionalidad de los estados temporarios es muy distinta a la de los permanentes. Los primeros generalmente se relacionan a casos excepcionales con transiciones en general complejas, usualmente para asegurar condiciones apropiadas de seguridad, confiabilidad y disponibilidad del sistema. Dado que las actividades constituyen hilos de proceso, su diseño y desarrollo pueden atacarse de manera razonablemente independientes, asegurando así la focalización que estos casos requieren. Para el desarrollo e interacción de estos hilos son de aplicación herramientas muy probadas en el campo de los Sistemas de Tiempo Real.

Por otro lado, los estados transitorios, incluidos en las actividades, representan por lo general condiciones inusuales y complejas, que afectan importantes propiedades del sistema: seguridad, confiabilidad y disponibilidad. Dado que las actividades pueden ser diseñadas como hilos de proceso, se las puede diseñar de manera semi-independiente, permitiendo al diseñador una mayor focalización. En consecuencia, se proveen mecanismos de descripción diferenciados que permiten concentrarse en la importancia particular de ambos casos.

Dado que un usuario usualmente describe al sistema según sus estados permanentes, el diseñador debe ahondar el análisis para descubrir los estados temporarios, que son los que brindarán sus principales características de robustez.

Referencias Bibliográficas

Maier M. & Rechtin E**. (**2000) *The Art of System Architecting*, 2nd edition. CRC Press, Inc. Boca Raton, FL, EEUU.

Keating, M. (2011) *The Simple Art of SoC Design*, Springer Science + Business Media, New York, EEUU.

Harel D. (1987). "Statecharts: a visual formalism for complex systems", *Journal Science of Computer Programming*, Vol. 8, Issue 3, pp. 231-274.

Higuchi H. & Matsunaga Y. (1996). “A fast state reduction algorithm for incompletely specified fihnite state machines”, Proc. Design Automation Conference, pp. 463-466.

Laplante, P. A. & Ovaska, S. J. (2012). *Real-Time Systems Design and Analysis*, 4th edition, IEEE Press, John Willey & Sons, New Jersey, EEUU.

Hopcroft, J. E. (1971). “An n log n algorithm for minimizing the states in a finite automaton” in *The Theory of Machines and Computations* (Z. Kohavi) *,* Academic Press, New York, EEUU, pp. 189–196.

Cohen, E. D., Volentini, E. & Giori, M. (2015), “Múltiples Entradas y Salidas en Sistemas Embebidos”, Memoria – Investigaciones en Ingeniería, Nº 13, Montevideo, Uruguay, pp. 49-62.

Autores

Eduardo Daniel Cohen es Profesor Titular de las asignaturas “Arquitectura de Computadoras” y “Sistemas con Microprocesadores y Microcontroladores” de la carrera Ingeniería en Computación. Dicta también las asignaturas de posgrado “Sistemas Embebidos Modernos” y “Sistemas Embebidos Avanzados” en la Especialización en Integración de Tecnologías Informáticas, en la FACET, UNT, de la cual es Director. Es Ingeniero Electrónico, FACET, UNT y M.Sc.E.E. orientación computación, Technion – Israel Institute of Technology. Dirige un programa de Investigación acreditado por CIUNT. Ha realizado numerosas publicaciones en su especialidad. Su carrera profesional incluye cargos de gestión y desarrollo en sistemas informáticos, temas en los cuales también se desempeñó como asesor independiente.

Esteban Volentini es Profesor Adjunto de la asignatura “Sistemas con Microprocesadores y Microcontroladores” de la carrera Ingeniería en Computación y participa del grupo de investigación en Metodologías de Diseño de Sistemas embebidos del Laboratorio de Microprocesadores de la FACET. Se desempeña además como responsable de desarrollo de la firma EQUISER. Es Ingeniero en Computación, FACET, UNT.

Juan Pablo Gruer se desempeñó durante 25 años como profesor-investigador en informática en la Universidad de Tecnología de Belfort-Montbéliard, habiéndose jubilado como Profesor Titular en 2015. Trabajó en la Industria de Francia como ingeniero especializado en tiempo-real durante cuatro años. Desde 2016 se desempeña como Profesor Titular contratado por la FACET, integrado al equipo del Laboratorio de Sistemas Embebidos. Es Ingeniero Electrónico, FACET, UNT, Ingeniero Informático del Instituto Nacional Politécnico de Grenoble, Francia y Ph.D. en informática de la Universidad de Alta Alsacia. Ha realizado numerosas publicaciones en su especialidad.