Cálculo del tiempo de respuesta de aplicaciones basadas en componentes desarrolladas con C-Forge

Francisco Sánchez-Ledesma, Diego Alonso, Juan Pastor y Francisca Rosique Contreras
División de Sistemas e Ingeniería Electrónica (DSIE), Universidad Politécnica de Cartagena, 30202 Cartagena, España.
Email: francisco.sanchez@upct.es

Resumen. *Al momento de diseñar un sistema de tiempo real estricto es importante conocer los tiempos de respuesta que va tener el sistema para cada una de sus operaciones críticas, de esta manera se puede garantizar que el sistema va a responder en un tiempo adecuado. En las aplicaciones basadas en componentes una operación crítica puede ser realizada por medio de la colaboración de múltiples componentes. Cuando se diseña un componente que va a formar parte de una aplicación de tiempo real el desarrollador establece cuál va a ser el tiempo de respuesta del mismo, pero no toma en cuenta el resto de la aplicación, puesto que no la conoce. Se hace evidente la necesidad de encontrar mecanismos para analizar el tiempo de respuesta de las operaciones que involucren múltiples componentes desarrollados por diferentes personas, que tomen en cuenta la aplicación como un todo y no solamente los componentes individuales de la misma.*

1. Introducción

Este documento es un ejemplo del formato de presentación deseado, y contiene información concerniente al diseño general del documento, familias tipográficas, y tamaños de tipografía apropiados. En el trabajo que aquí se presenta se ofrece un enfoque que usa técnicas de desarrollo basado en componentes (CBSE por sus siglas en inglés) para reducir la complejidad de las aplicaciones, sin dejar de lado los requisitos de tiempo real y centrándonos en los aspectos relacionados con el análisis de tiempo de respuesta de las operaciones críticas de una aplicación. Se plantean los mecanismos desarrollados para el análisis de tiempo de respuesta de las aplicaciones desarrolladas utilizando el lenguaje de modelado WCOMM y el framework de componentes FraCC. Se decidió seguir un enfoque de arquitectura dirigida por modelos [1] (MDA por sus siglas en inglés) para modelar este tipo de aplicaciones, en un contexto de desarrollo basado en una forma particular de modelar aplicaciones basadas en componentes independiente de la plataforma y en el uso de frameworks para proporcionar el soporte de ejecución específico de la plataforma.

2. Cadena de Herramientas C-Forge

En esta sección se presentan las instrucciones de edición para las figuras, tablas, abreviaturas y acrónimos. C-Forge es una cadena de herramientas abierta, desarrollada sobre la plataforma Eclipse, que emplea sus facilidades de diseño dirigido por modelos para soportar un proceso de desarrollo basado en componentes. C-Forge está formado por las siguientes herramientas: Un lenguaje para modelar aplicaciones basadas en componentes, denominado WCOMM. Una versión preliminar está descrita en [2]. Un framework denominado FraCC, que proporciona el soporte de ejecución necesario para las aplicaciones modeladas mediante WCOMM.

2.1. Lenguaje de componentes WCOMM

Un componente WCOMM es una entidad que encapsula su estado interno, que consta de una parte estructural y una parte de comportamiento. La parte estructural viene definida por sus puertos y por los mensajes que fluyen a través de ellos, agrupados en interfaces. Estos mensajes se envían siguiendo el esquema de comunicación asíncrono sin respuesta. El comportamiento se define mediante una máquina de estados finita, similar a la que define UML, extendidas con propiedades temporales. Es decir, el usuario modela el comportamiento del componente mediante estados, transiciones, eventos, guardas y regiones, tanto ortogonales como jerárquicas. Cada estado puede tener opcionalmente definida una actividad interna, que se asociará posteriormente en FraCC con código. En WCOMM también se modela lo que denominamos “carcasa” de la actividad, formada por los mensajes que intercambia y los eventos que genera. Estos eventos, junto con la recepción de mensajes a través de los puertos, son los responsables del cambio de estado del componente, y son por tanto, los que establecen la conexión entre estructura y comportamiento. Finalmente, una aplicación se modela como un conjunto de componentes conectados entre sí.

2.2. Framework FraCC

FraCC es un framework de componentes programado en C++ que fue desarrollado con el doble objetivo de proporcionar (1) soporte completo a las características del modelo de componentes WCOMM, (2) control completo sobre las características de concurrencia de la aplicación al usuario, que es quien decide cuántos procesos e hilos

se crean y en qué hilos se ejecutan los componentes y (3) control explícito de la distribución de componentes en nodos computacionales. Estas características permiten el uso de FraCC en aplicaciones con restricciones de tiempo real.

3. Comunicaciones en FraCC

FraCC no implementa la estructura de componentes completa como se define en WCOMM. Se ha optado por implementar los elementos que afectan el comportamiento de la aplicación (actividades, estados, regiones, transiciones y mensajes) y otros elementos que sirven de soporte para la ejecución y las comunicaciones (buffers, hilos, procesos).

A partir del modelo de una aplicación es posible extraer las regiones que forman cada componente y los enlaces que existen entre cada una de ellas, obteniendo un grafo. Los enlaces entre regiones indican que dos regiones cualesquiera intercambian mensajes, independientemente de si se encuentran dentro del mismo componente o no.

En la figura 1 se muestra el diagrama de clases con los elementos que intervienen en el proceso de comunicación entre regiones. La clase RegionActivity es la encargada de gestionar el funcionamiento de las regiones (transiciones entre estados y las comunicaciones). La clase ActivityCode modela el código que va a ejecutar la aplicación, el desarrollador hereda de esta clase para implementar el código propio de la aplicación. Los buffer (clases InterprocessQueue, ActivityQueue y InternalQueue) se han implementado como pilas. Cada RegionActivity contiene un buffer (InterProcessQueue) en el que se depositan los mensajes que tienen como destino la región que maneja. La clase ActivityCode contiene un buffer (InternalQueue) por cada tipo de mensaje que recibe. Cuando una actividad (ActivityCode) quiere comunicarse con otra crea un mensaje que contiene la información que quiere enviar y lo inserta en el buffer de entrada (InterprocessQueue) de la RegionActivity que contiene la actividad receptora. La RegionActivity lee su buffer de entrada periódicamente y actualiza los buffers de entrada de las actividades dependiendo del destinatario del mensaje.

4. Análisis de tiempo de respuesta

Al momento de diseñar un sistema de tiempo real estricto es importante conocer los tiempos de respuesta que va tener el sistema para cada una de sus operaciones críticas. Las operaciones críticas de un sistema diseñado e implementado con WCOMM/FraCC pueden dividirse en múltiples componentes que pueden contener múltiples regiones. Se hace evidente la necesidad de encontrar un mecanismo que nos permita analizar el tiempo de respuesta de las operaciones que involucran múltiples regiones partiendo de la especificación de las actividades contenidas en las regiones.

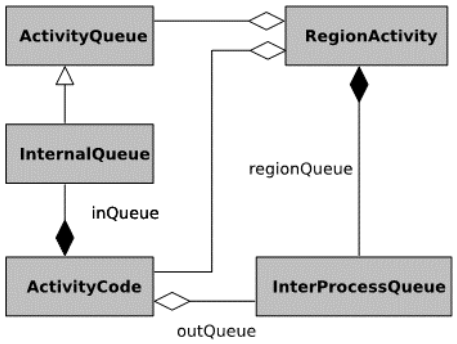

Figura 1. Diagrama de clases que muestra la estructura de las comunicaciones en FraCC.

De cada actividad se conoce su tiempo de computo WCET y el tiempo mínimo entre cada ejecución T (periodo en el caso de actividades periódicas e interarrival time en caso de esporádicas), puesto que forman parte de los requisitos del sistema. A partir de estos datos se puede calcular el periodo de ejecución de la región T y su tiempo de computo WCET en el peor de los casos como se indica en [3]. El tiempo de respuesta RT de una región ante cualquier requerimiento será igual a su peor tiempo de computo más el periodo de actualización. Estos datos junto con la información del modelo de despliegue se pueden analizar utilizando herramientas de análisis de planificabilidad como se explica en [4]. De esta manera podemos garantizar el tiempo de respuesta de cada una de las regiones de forma individual.

El grafo extraído de modelo de la aplicación indica cuáles regiones intercambian mensajes y se puede utilizar también para determinar el flujo de los datos entre dos regiones cualesquiera. Para determinar el tiempo de respuesta para una operación cualquiera se debe conocer cuál región inicia la operación y en cuál región termina. Si por ejemplo tuviéramos un robot y quisiéramos saber en qué tiempo reacciona a un comando de Stop, necesitaríamos conocer cuál región es la encargada de leer el pulsador de stop y cuál es la encargada de accionar el freno. A partir de estos datos podemos calcular la ruta crítica entre una región y otra en el grafo de conexiones entre regiones. El peso de los enlaces del grafo es igual al tiempo que tarda en entregar una región un mensaje a otra, para el caso de dos regiones que se encuentren en el mismo nodo será igual a cero (debido a que el tiempo de actualización del buffer debe ser tomado en cuenta para calcular el tiempo de computo de la actividad) y si están ubicadas en nodos distintos vendrá dado por las especificaciones de la red de comunicaciones. El peso de cada uno de los nodos (que representan las regiones) es igual al tiempo de respuesta de la región. Una vez conocida la ruta crítica se procede a sumar todos los pesos involucrados y tendremos el tiempo de respuesta. Este análisis sólo es valido si cada una de la regiones está en un estado en que puede atender la solicitud o a una transición de distancia de ese estado.

El grafo sobre el cual se hace el análisis de tiempo de respuesta es conforme al meta-modelo que se

representa en la figura 2. Con las clases RegionGraph, Region y Edge se modela el grafo que representa todas las regiones de una aplicación y los enlaces que existen entre ellas. La clase TransactionSpec asocia la región de origen y la región en la que termina la operación crítica que se desea analizar.

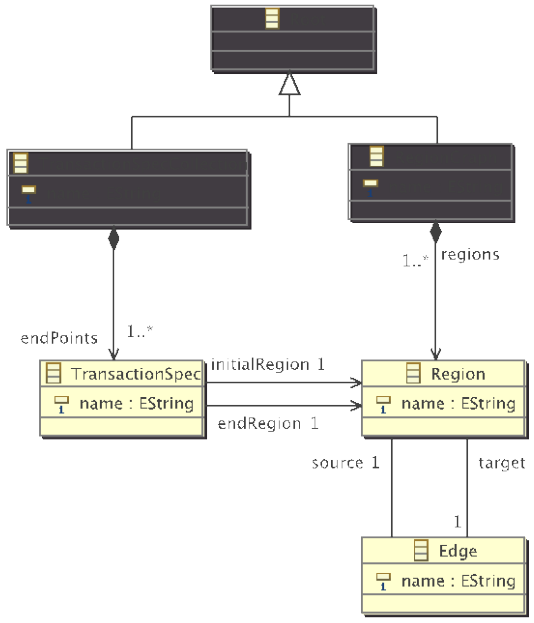

Figura 2: Meta-modelo del grafo utilizado para el análisis del tiempo de respuesta.

4. Conclusiones y trabajos futuros

En este artículo se ha descrito la evolución de un trabajo anterior, donde fue descrito un framework orientado a objetos para la implementación de aplicaciones basadas en componentes. Se han definido mecanismos para analizar el tiempo de respuesta de las operaciones en las que colaboran múltiples componentes. Este tipo de análisis hace que los desarrolladores puedan garantizar que las aplicaciones van a poder cumplir con las tareas críticas en un tiempo determinado. Esta propuesta no es la solución definitiva para hacer el análisis de tiempo de respuesta en C-Forge, sino que es una primera aproximación y en un futuro se implementarán y probarán otros esquemas. En particular intentaremos integrar herramientas de análisis existentes y adaptarnos a los estándares de la industria.

El trabajo descrito en este artículo es un trabajo en marcha. Actualmente se sigue trabajando para mejorar y perfeccionar las herramientas desarrolladas, editores y transformaciones para obtener una herramienta más robusta. Por otro lado, también estamos interesados en enriquecer el modelo de despliegue para que permita mayor granularidad. En particular queremos agregar capacidad para que el despliegue se pueda hacer en plataformas con múltiples núcleos y se pueda especificar en cual núcleo van a ejecutar cada uno de los hilos de la aplicación. Y que a la aplicación resultante sea temporalmente analizable para verificar si es panificable o no.

Agradecimientos

Este trabajo ha sido financiado parcialmente por la beca otorgada por el gobierno de España MEC FPU (ref. AP2009-5083).

Referencias

[1] S. Mellor, K. Scott, A. Uhl, and D. Weise, MDA Distilled, 1st ed., ser. Object Technology, S. Mellor, Ed. awp, Mar. 2004.

[2] D. Alonso, C. Vicente-Chicote, F. Ortiz, J. Pastor, and B. Álvarez, “V3CMM: a 3-view component meta-model for model-driven robotic software development,” Journal of Software Engineering for Robotics (JOSER), vol. 1, no. 1, p. 3–17, Jan. 2010.

[3] F. Sanchez-Ledesma, L. Pinho, D. Alonso, and J. Pastor, “Desarrollo de aplicaciones con requisitos de criticidad temporal mixta utilizando c-forge,” Simposio de Sistemas de Tiempo Real, 2013.

[4] F. Sánchez-Ledesma, J. Á. Pastor, D. Alonso, B. Álvarez, and P. Sánchez, “Propuesta de análisis temporal de aplicaciones basadas en componentes,” in XV Jornadas de Tiempo Real, Santander, 2012.