Desarrollo de Software Adaptativo para Robótica

Juan F. Inglés-Romero y Cristina Vicente-Chicote
División de Sistemas e Ingeniería Electrónica (DSIE)
Universidad Politécnica de Cartagena
Antiguo Cuartel de Antigones, 30202 Cartagena (Murcia)
juanfran.ingles@upct.es, cristina.vicente@upct.es

Resumen. *La complejidad de los sistemas robóticos actuales aumenta a medida que lo hace la de sus tareas y entornos de trabajo. Así, cada vez más, resulta necesario dotar a los robots de mecanismos de adaptación que les permitan hacer frente, de manera autónoma y robusta, a los cambios que se produzcan en su entorno y a los posibles fallos que puedan ocurrir tanto en su software como en su hardware. Para ello, parece adecuado adoptar un enfoque basado en modelos de alto nivel, que el robot pueda interpretar y adaptar en tiempo de ejecución. En este artículo se describen los resultados de aplicar el enfoque de Desarrollo de Software Dirigido por Modelos (DSDM), propuesto en el Proyecto Europeo DiVA, para diseñar software adaptativo para robótica.*

1 Introducción

La creciente flexibilidad y facilidad de uso de los robots actuales está permitiendo su incorporación gradual en cada vez más ámbitos de nuestra vida cotidiana. Así, hoy en día, es posible encontrar robots no sólo en la industria, sino también en muchos negocios, lugares públicos e incluso en nuestros hogares. Para conseguir que estos robots puedan desenvolverse adecuadamente en entornos no estructurados y cambiantes, resulta esencial dotarles de mecanismos que les permitan hacer frente, de manera autónoma y robusta, a la enorme variedad de situaciones con las que se pueden encontrar (cambios en su entorno, cambios en los recursos energéticos o computaciones de los que disponen en cada momento, cambios en lo que les demandan los distintos usuarios, fallos en el hardware o en el software, etc.) [1]. Para ello, resulta necesario dotar a los robots de (auto-)conciencia y de capacidad de (auto-)adaptación [2], esto es, de mecanismos que les permitan construir y adaptar dinámicamente modelos tanto de sí mismos como de sus entornos.

La Agenda Estratégica de Investigación (SRA, Strategic Research Agenda) [1], presentada por la Plataforma Tecnológica Europea para la Robótica, define *adaptación* como un *cambio en el proceso o en el método de ejecución que lleva a cabo un sistema, por lo general, en tiempo de ejecución*. La SRA pone de manifiesto la importancia de hacer uso de modelos a la hora de automatizar los procesos de adaptación. En este sentido, la adopción de un enfoque de Desarrollo de Software Dirigido por Modelos (DSDM) [3], ofrece una aproximación prometedora desde la que abordar, entre otros muchos retos, el de la (auto-)adaptación del software para robótica. Por otro lado, aunque no directamente vinculado al dominio de la robótica, el proyecto Europeo DiVA [4] propone el uso de modelos en tiempo de diseño, simulación y ejecución (*models@runtime*) para dotar al software de la capacidad de adaptarse dinámicamente.

En este artículo describimos, mediante un caso de estudio, nuestra experiencia utilizando el marco de trabajo (en adelante *framework*), desarrollado como parte de DiVA, para diseñar el comportamiento (auto-)adaptativo de dos robots móviles. Esta experiencia nos ha permitido identificar los beneficios y limitaciones de DiVA en el ámbito de la robótica, así como algunos de los retos que aún se deben abordar para poder afrontar el desarrollo de software adaptativo para robótica desde el DSDM.

2 Diseño de Software Adaptativo para Robótica

El caso de estudio, desarrollado como parte de este trabajo, considera un escenario con dos robots (tipo e-puck, http://www.epuck.com/) ubicados inicialmente en posiciones aleatorias dentro de una sala, en la que se habrán dispuesto algunos obstáculos. Uno de los robots asumirá el rol de *Victima*, mientras que el otro actuará como *Rescatador*.

El objetivo de la Victima es conseguir que el Rescatador la localice cuanto antes. Para ello, puede indicar su posición usando una señal sonora o luminosa. La Victima comunica, a través de su Bluetooth, tanto el tipo de señalización que está emitiendo, como su estado: i) OK, ii) herido, o iii) KO. Los sensores de proximidad permiten a la Victima detectar objetos cercanos, que podrá esquivar adoptando una de éstas dos estrategias: i) bordeando el obstáculo, o ii) cambiando de dirección. Por último, la Victima puede adoptar diferentes tácticas para mejorar su visibilidad y facilitar su localización: i) movimientos rápidos aleatorios, ii) movimientos lentos aleatorios, o iii) permanecer parado (en orden descendente según el grado en que mejoran la visibilidad de la Víctima y según su impacto en el consumo energético).

Por otro lado, el objetivo del Rescatador es encontrar a la Victima lo antes posible. Para ello, dispone de tres tipos de sensores, i) cámara, ii) micrófonos estereoscópicos (permiten identificar la dirección de

un sonido), y iii) varios sensores de proximidad (en orden descendente de precisión y consumo de energía). Si el Rescatador contacta con la Victima vía Bluetooth, sabrá qué tipo de señalización está emitiendo y, por lo tanto, podrá seleccionar el sensor y la estrategia de búsqueda más adecuada. Mientras esto no ocurra, se moverá aleatoriamente por la sala. El rescatador implementa las mismas dos estrategias de evitación de obstáculos que la Víctima. Además, también tienen en común sensores para medir el nivel de luz y de ruido ambiental, así como la carga de sus baterías. Con todo ello, los robots deberán adaptar su comportamiento dinámicamente ante los cambios que se produzcan en su entorno, en su nivel de recursos internos y su percepción del otro, seleccionando las estrategias y sensores que les permitan alcanzar sus objetivos de la forma más eficiente posible.

De forma general, el mecanismo de adaptación suele implicar tres procesos: (1) *monitorización del contexto*, (2) *toma de decisiones,* y (3) *reconfiguración*. El framework desarrollado como parte del proyecto DiVA permite modelar la información que necesitan estos tres procesos mediante la descripción de cuatro aspectos esenciales: Contexto, Variabilidad, Razonamiento y Arquitectura.

El *modelo de Contexto* describe las variables que se deben monitorizar. En la Figura 1a se muestra el modelo de Contexto definido para el Rescatador, en el que se observa que se han incluido cinco variables. Tres de ellas son booleanas y reflejan el estado de la batería del robot (*Low Battery)* y las condiciones ambientales (*Ambient Light* y *Ambient Noise)*. Las otras dos, *Signal Notification* y *Victim State*, son variables de tipo enumerado que registran los cambios en la señalización y el estado de la Victima.

El *modelo de Variabilidad* describe los puntos de variación del sistema (dimensions) y las alternativas disponibles para cada uno de ellos (variants), así como sus dependencias y restricciones. La Figura 1d muestra el modelo de Variabilidad del Rescatador en el que se puede apreciar que la dimensión *Search Strategy* tiene tres posibles variantes, cada una de ellas asociada a una de las estrategias de búsqueda del Rescatador. El hecho de que la cardinalidad mínima (Lower) y máxima (Upper) de la dimensión sea 1, indica que el Rescatador debe tener siempre una y sólo una estrategia de búsqueda seleccionada. Como se puede ver en la figura, la variante *Detailed* (búsqueda detallada) requiere el uso tanto de la cámara como de los micrófonos (ver columna *Dependency*). Las expresiones en las columnas *Available* y *Required* indican, respectivamente, cuándo cada variante está disponible o es requerida en función del valor del contexto. Así, por ejemplo, sólo se podrá seleccionar la opción de búsqueda *Detailed* cuando la batería del robot no sea baja (*not LowBattery*).

Para dar soporte al proceso de toma de decisiones (elección de las variantes más adecuadas en función del contexto), DiVA ofrece tres modelos.

	Name	ID
Enum	Signal Notification	Signal
Literal	LIGHT	LIGHT
Literal	ACUSTIC	ACUSTIC
Literal	Unknown	USIG
Enum	Victim State	STATE
Literal	Unknown	UNKN
Literal	OK	OK
Literal	Wounded	Wounded
Literal	KO	KO
Boolean	Low Battery	LowBatt
Boolean	Ambient Light	LIGHT
Boolean	Ambient Noise	NOISE

(a)

	Power...	Search...
Sensors (SEN)	true	False
Camera (CAM)	High	-
Microphones (MIC)	Medium	-
Networking (NET)	true	true
Bluetooth (BT)	High	Low
Search Strategy (SST)	false	true
Detailed (DS)	-	Very High
Simple (SS)	-	Medium
Blind (BS)	-	Very Low
Obstacles Strategy (OBSST)	true	true
Surround (SUR)	Low	Medium
Change Direction (CD)	Very Low	Low

(b)

Name	context	Power...	Search...
Battery is Low	LowBatt	High	-
Unknown Signaling	Signal=USIG	Low	High
Unknown State	STATE=UNKN	Low	High
Battery is OK	not LowBatt	Low	-
Night	not (LIGHT)	-	Very High
Victim OK	STATE=OK	Medium	Medium
Victim NOK	not STATE=OK	-	Very High

(c)

	Name	ID	Lower	Upper	dependency	available
Dimension	Sensors	SEN	0	2	-	-
Variant	Camera	CAM	-	-	not BS	LIGHT or Signal=LIGHT
Variant	Microphones	MIC	-	-	not BS	not NOISE or Signal=ACUSTIC
Dimension	Networking	NET	0	1	-	-
Variant	Bluetooth	BT	-	-		
Dimension	Search Strategy	SST	1	1	-	-
Variant	Detailed	DS	-	-	CAM and MIC	not LowBatt
Variant	Simple	SS	-	-	CAM or MIC	
Variant	Blind	BS	-	-		
Dimension	Obstacles Strategy	OBSST	1	1	-	-
Variant	Surround	SUR	-	-		
Variant	Change Direction	CD	-	-		

(d)

	Name	Direction
Property	Power comsuption	0
Property	Search accuracy	1

(e)

Figura 1. Modelos DiVA desarrollados para el rol Rescatador

El primero de ellos (ver Figura 1e) establece las propiedades globales del sistema que se deben optimizar, bien maximizándolas (Direction=1) o minimizándolas (Direction=0). En el caso de estudio propuesto, el Rescatador debe maximizar la precisión de la búsqueda (*Search Accuracy*) y minimizar el consumo energético (*Power Comsuption*).

Por otro lado, DiVA también proporciona un modelo (ver Figura 1b) en el que el diseñador puede valorar, en una escala de cinco valores de 1 (Very Low) a 5 (Very High), el impacto de cada una de las variantes del modelo de Variabilidad (ver Figura 1d) en las propiedades que se deben optimizar en el sistema (ver Figura 1e). Por ejemplo, la variante Bluetooth tiene un impacto alto en el consumo de energía, pero bajo en cuando a la precisión de la búsqueda.

Finalmente, la Figura 1c muestra el modelo de reglas de prioridad en el que se indica cómo de importantes son cada una de las propiedades en función del contexto. Así, cuando la batería está agotándose (regla *Battery is Low*), el objetivo de minimizar el consumo de energía se convierte en prioritario frente al de maximizar la precisión de búsqueda. Del mismo modo, cuando el nivel de carga de la batería es adecuado (regla *Battery is OK*), ocurre justo lo contrario.

Por último, destacamos la necesidad de describir la arquitectura del sistema dado que, hasta este punto, sólo se ha descrito su variabilidad a nivel conceptual,

pero no ligada a ninguna tecnología concreta. Así, por ejemplo, si se adopta un enfoque de Desarrollo de Software Basado en Componentes (DSBC) [2], se debería especificar qué componentes hay disponibles para implementar cada una de las variantes descritas en el modelo de Variabilidad.

3 Retos Futuros

El DSDM ha logrado resultados prometedores en el ámbito del software (auto-)adaptativo, en la medida que facilita el diseño y la ejecución de mecanismos dinámicos de adaptación en sistemas de muy diversa índole. Buena muestra de ello son los resultados alcanzados en el proyecto DiVA. No obstante, en su aplicación a la robótica, aún quedan varias cuestiones por resolver. A continuación se describen algunos de los retos que hemos identificado como pendientes tras nuestra experiencia en el uso de DiVA para el desarrollo de software adaptativo para robótica.

- *Adaptación cooperativa*. En robótica, suele ser habitual que un robot tenga que interactuar con otros sistemas (p. ej., otros robots). En este sentido los diseñadores deberían poder modelar los mecanismos de adaptación de un robot considerando sus relaciones con otros sistemas. Sin embargo, tanto en DiVA como en la mayoría de las propuestas actuales, el diseño de estrategias de adaptación cooperativa es una cuestión no resuelta y prácticamente inexplorada.
- *Adaptación multicapa*. Relacionado con el punto anterior, es posible conformar grupos de trabajo en los que varios robots llevan a cabo una o más tareas de manera cooperativa. Las estrategias de adaptación de estas agrupaciones deberían poder diseñarse como si de una única entidad se tratase (p. ej. definiendo sus objetivos y estrategias globales a nivel de agrupación). Así, debería ser posible definir una jerarquía de adaptación, que permitiera especificar los objetivos y estrategias, por ejemplo, de una comunidad de robots, de las colonias de dicha comunidad, de los grupos de trabajo de cada colonia, o de los robots de cada grupo de trabajo. Actualmente DiVA no permite realizar esta distinción.
- *Impacto del contexto en la simulación*. Para poder simular adecuadamente el comportamiento adaptativo de los sistemas modelados con DiVA, sería interesante conocer el impacto de cada dimensión del contexto, en particular, aquellas que dependen de otros sistemas.
- *Contextos compartidos*. DiVA permite definir el contexto de un sistema como el conjunto de variables que se deben monitorizar. No obstante, no soporta la compartición de este contexto entre diferentes sistemas, algo por lo general muy necesario en robótica (p. ej., para permitir que varios robots cooperen en la elaboración de un mapa global compartiendo sus mapas locales).
- *Gestión de la incertidumbre*. Si se considera la posibilidad de contextos compartidos, será necesario controlar la consistencia y fiabilidad de la información compartida, proporcionando alguna medida que permita a los robots saber cómo de segura o precisa es la información que recibe de los otros.
- *Uso de modelos en tiempo de ejecución (models@runtime) para implementar, no sólo la arquitectura del sistema, sino también sus componentes.* Los enfoques actuales, como DiVA, describen los sistemas, como mucho, a nivel de arquitectura. En consecuencia, sólo soportan mecanismos de adaptación a este nivel (esto es, añadir o eliminar componentes de la arquitectura y enlaces entre ellos). Para soportar una adaptación de grano más fino, sería interesante poder diseñar estrategias de adaptación a nivel de componentes o incluso de subcomponentes de la arquitectura.

4 Conclusiones

En este artículo hemos descrito nuestra experiencia utilizando los resultados del proyecto DiVA para diseñar dos sistemas adaptativos robóticos que interactúan entre sí. Entre las conclusiones, cabría destacar las siguientes: (1) las propuestas basadas en DSDM y, en particular, en *models@runtime*, como es el caso de DiVA, ofrecen una primera aproximación al desarrollo de software (auto-)adaptativo; (2) los modelos deberían convertirse en el artefacto central del diseño, simulación y ejecución del software robótico; (3) para que los robots sean capaces de llevar a cabo tareas, cada vez más complejas, de manera autónoma y robusta, es necesario dotarles de capacidades de (auto-)conciencia y (auto-)adaptación; (4) los robots deben diseñarse para poder interactuar y cooperar con otros sistemas. En este sentido, es importante dotarles de mecanismos para compartir información e incluso estrategias de adaptación. Como línea de trabajo futuro, nos planteamos abordar algunos de los retos identificados en este artículo.

Agradecimientos

Este trabajo ha sido financiado por los proyectos EXPLORE (MICINN, TIN2009-08572) y MISSION (F. Séneca, 15374/PI/10). Juan F. Inglés agradece a la F. Séneca su beca FPI (Exp. 15561/FPI/10).

Referencias

[1] European Robotics Technology Platform (EUROP). Robotic visions to 2020 and beyond - The Strategic Research Agenda for Robotics in Europe. Appendix: Technology Roadmaps. http://www.robotics-platform.eu/sra, 2009.

[2] D. Brugali and A. Shakhimardanov. Component-Based Robotic Engineering. Part II: Models and Systems. IEEE Robotics and Automation Magazine, 17(1):100-112, 2010.

[3] T. Stahl, M. Voelter, and K. Czarnecki, Model-Driven Software Development: Technology, Engineering, Management, ed. Wiley, 2006, ISBN: 978-0-470-02570-3.

[4] DiVA Project. http://www.ict-diva.eu/