

Pruebas Funcionales Evolutivas en Industria

Tanja E.J. Vos

Centro ProS
Universidad Politécnica de Valencia
tvos@pros.upv.es

Arthur I. Baars

Centro ProS
Universidad Politécnica de Valencia
abaars@pros.upv.es

Beatriz Marín

Centro ProS
Universidad Politécnica de Valencia
bmarin@pros.upv.es

Resumen

Las pruebas evolutivas han sido investigadas por la academia, obteniendo resultados prometedores. Sin embargo, estas pruebas casi no han sido aplicadas a sistemas complejos reales, por lo que se tiene escaso conocimiento acerca de la escalabilidad, aplicabilidad, y aceptabilidad de las pruebas evolutivas en entornos industriales. Este artículo contribuye a mejorar esta situación presentando los resultados obtenidos en el proyecto europeo EvoTest (IST-33472).

1. Introducción

Hoy en día, la técnica de aseguramiento de calidad más importante y más usada en la industria corresponde a las pruebas de software, que pueden tomar más del 50% del tiempo y coste de desarrollo [12]. Aunque actualmente existen herramientas que automatizan las pruebas, ayudando a planificarlas, controlarlas, y ejecutar y monitorizar los casos de prueba; todas ellas comparten una filosofía pasiva respecto al diseño de casos de prueba, la selección de los datos de prueba, y la evaluación de las pruebas. Esta filosofía es pasiva, ya que la persona que realiza las pruebas debe realizar estas actividades cruciales de forma manual. Esto se debe a que estas actividades son difíciles de automatizar de manera efectiva y escalable con las técnicas usadas por la industria, ya que la cantidad de posibles casos de prueba es demasiado grande para ser explorado exhaustivamente, incluso para programas triviales.

Teniendo en cuenta que los casos de prueba determinan el tipo y el alcance de una prueba, el diseño de éstos es *la actividad* que determina la calidad y efectividad de todo un proceso de pruebas. La falta de automatización de esta actividad primordial ha provocado que la industria gaste grandes cantidades de esfuerzo y dinero en las pruebas del software, y que la calidad de las

pruebas resultantes es baja y no encuentran errores importantes del sistema.

Para generar automáticamente casos de prueba efectivos, técnicas como la optimización estocástica y las búsquedas inteligentes (ej. algoritmos evolutivos) han sido aplicadas exitosamente en la academia. Respecto a las pruebas de caja negra, una parte considerable de la literatura ha demostrado el éxito de las pruebas evolutivas frente a métodos tradicionales para las pruebas temporales [14][15][16], donde la prueba es interpretada como una optimización del problema y se emplea computación evolutiva para encontrar datos de prueba con tiempos de ejecución cortos y largos. Además, para el caso de las pruebas funcionales de caja negra, en [17][18][19] se muestran resultados prometedores para las pruebas evolutivas. Sin embargo, las pruebas evolutivas han sido escasamente aplicadas a sistemas complejos reales, por lo que se tiene poco conocimiento acerca de su escalabilidad y aceptabilidad en entornos industriales.

Este artículo describe dos casos de estudio realizados en el contexto del proyecto EvoTest [11] que reportan la escalabilidad, aplicabilidad, y aceptabilidad de las pruebas funcionales evolutivas de caja negra en entornos industriales. El artículo está organizado de la siguiente manera: la Sección 2 presenta los fundamentos utilizados en los estudios, la Sección 3 presenta el diseño de los casos de estudio, y la Sección 4 presenta la ejecución de los estudios. La Sección 5 presenta la evaluación de los resultados, y la Sección 6 presenta las conclusiones y trabajos futuros.

2. Pruebas Evolutivas

Para automatizar las pruebas de software utilizando pruebas evolutivas, el objetivo que persigue una prueba debe ser transformado en una tarea de optimización. Dependiendo del objetivo perseguido por la prueba, diferentes funciones de adecuación («*fitness function*» o «*objective function*») se deben especificar para la evaluación

de los datos de prueba. Si se puede definir una función de adecuación para el objetivo de la prueba, y se utiliza computación evolutiva para la búsqueda, las pruebas evolutivas se aplican así:

- 1) Se genera, usualmente de manera aleatoria, un conjunto inicial de datos de prueba.
- 2) Cada individuo dentro de la población (conjunto inicial) representa un dato de prueba con el cual el sistema sometido a pruebas («*Software Under Test*» - SUT) es ejecutado.
- 3) Por cada dato de prueba, se monitoriza la ejecución y se determina el valor de adecuación para el individuo correspondiente.
- 4) Los datos de prueba con altos valores de adecuación son seleccionados con una mayor probabilidad que aquellos que tienen bajos valores de adecuación, y se someten a procesos de combinación y mutación para generar nuevos datos de prueba (hijos).
- 5) Se genera una nueva población de datos de prueba mezclando los datos de prueba hijos y padres mediante procedimientos de supervivencia.

Este proceso de aplicación de pruebas evolutivas al software se repite hasta que el objetivo de la prueba se ha alcanzado o se ha alcanzado otra condición de detención dada.

2.1. Pruebas Funcionales Evolutivas

Cuando se realizan pruebas funcionales, el SUT es considerado como una caja negra, es decir, el probador no tiene conocimientos sobre cómo se ha implementado el software, y se limita a observar el comportamiento de sus entradas y salidas. El objetivo de las pruebas funcionales es verificar que el SUT satisface los requisitos funcionales.

Respecto a las pruebas funcionales evolutivas, se ha llevado a cabo poca investigación en comparación con la amplia investigación realizada respecto a las pruebas estructurales (ej. [1][2]). Para realizar una prueba funcional evolutiva, se utiliza un algoritmo evolutivo que busca datos de prueba para los cuales el comportamiento del SUT sea defectuoso con respecto a una propiedad funcional específica. Por este motivo, es necesario definir una función de adecuación que mida cómo de cerca está un dato de prueba de quebrar una propiedad seleccionada. Durante la búsqueda evolutiva, cada generación sucesiva de datos de prueba conseguirá estar más cerca de quebrar la propiedad funcional seleccionada. Así, cuando el óptimo es alcanzado se descubre un defecto, es decir, un dato de prueba para el cual el SUT falla

en la satisfacción de un requisito. Si no se descubren estos datos de prueba, entonces se adquiere confianza en la correcta implementación de la propiedad funcional.

2.2. Entorno para Pruebas Evolutivas

Para facilitar el desarrollo de pruebas evolutivas, el proyecto EvoTest ha implementado un entorno para ejecutar estas pruebas («*Evolutionary Testing Framework*» - ETF). El entorno ETF es una extensión de Eclipse [9] que incluye el generador evolutivo GUIDE [10] y un componente de interfaz gráfica para afinar los parámetros del algoritmo. ETF provee varias vistas para visualizar el progreso de la búsqueda. Además, ETF provee un punto de extensión para que generadores de otras técnicas meta-heurísticas de búsqueda puedan ser conectadas fácilmente.

Para personalizar el entorno para un objetivo de prueba particular, se deben proveer los siguientes componentes específicos de dominio (vea el manual de usuario de ETF [11]):

- 1) Una especificación que describe la estructura de los individuos, es decir, los datos de prueba.
- 2) Un administrador («*driver*») de pruebas, que permite conectar ETF con el SUT. El driver convierte los individuos del proceso de búsqueda en datos de prueba, ejecuta el SUT usando estos datos, monitoriza la salida del SUT, y entrega los resultados de la monitorización a ETF.
- 3) Una función de adecuación, que calcula la adecuación de los datos de prueba usando los resultados de la monitorización.

Así, para establecer el entorno ETF para pruebas funcionales evolutivas, es necesario desarrollar el driver de pruebas y la función de adecuación como plugins de Eclipse para que sean cargados y ejecutados por ETF. La especificación individual se realiza en un archivo XML. Cuando el SUT dependa de señales de entrada continuas, el entorno ETF permite usar una herramienta generadora de señales, como han propuesto Windisch y Al Moubayed [3].

3. Diseño del Caso de Estudio

Ninguno de los enfoques automáticos para el diseño de pruebas propuestos por la academia ha alcanzado un nivel de aplicabilidad en entornos industriales [4][5][6]. Además, existe un alto interés en estudios empíricos de técnicas de pruebas evolutivas y su escalabilidad en prácticas

reales de la industria. Así, la pregunta de investigación a ser resuelta por este trabajo es:

¿Las pruebas evolutivas de caja negra son escalables a las prácticas reales en la industria?

3.1. Selección de los casos

Para asegurar que los casos seleccionados corresponden a sistemas complejos reales, nuestros socios industriales Daimler y BMS han elegido sistemas para los cuales decidieron que la aplicación exitosa de pruebas evolutivas es más importante en sus entornos industriales.

Dado que la aplicación exitosa de pruebas evolutivas en entornos industriales es el objetivo más importante del caso de estudio, los socios industriales no fueron forzados con demasiadas restricciones para seleccionar un sistema que se ajuste mejor al estudio empírico. Aunque el acceso a previas versiones (con información de sus fallos, cobertura y técnicas usadas para encontrar los fallos) hubiera sido ideal para nuestro estudio, esto no siempre fue posible en el entorno industrial, y otras veces no se cumplía con las necesidades y deseos del socio industrial.

Daimler y BMS han elegido sistemas de control embebidos para el dominio automoción que provienen de su producción en serie. Daimler ha elegido un módulo de control de velocidad adaptable («*Adaptive Cruise Control*» - ACC) cuyo propósito mantener una velocidad dada, y además controlar la distancia hacia los vehículos que le preceden – medida usando radares – para prevenir accidentes. BMS ha seleccionado un sistema de anti-bloqueo de frenos («*Anti-lock Braking System*» - ABS) que previene que las ruedas de un vehículo se bloqueen mientras se está frenando, permitiendo así que el conductor mantenga el control de la dirección durante las frenadas fuertes, y en la mayoría de las situaciones, acortando las distancias de frenado.

3.2. Proposiciones

Hemos refinado nuestra pregunta de investigación en 5 proposiciones que pueden ser evaluadas a través de variables serán medidas durante la ejecución del estudio. Para hacer esto, hemos estudiado y discutido profundamente con los socios industriales para entender qué significa ser escalable en estas aplicaciones industriales de acuerdo a los resultados de las pruebas evolutivas.

Proposición 1: Es posible usar el ETF sin conocimientos detallados sobre computación evolutiva para buscar datos de prueba interesantes.

Proposición 2: El ETF aplicado a ejemplos de tamaño real puede generar más casos de prueba orientados al problema que las pruebas aleatorias. Esto se debe a que mientras los valores de adecuación son mejorados, el sistema está continuamente siendo ejercitado más cerca de una condición de frontera (un valor de adecuación óptimo quiebra la frontera, ej. el requisito).

Proposición 3: El ETF es más efectivo que las pruebas aleatorias en la detección de errores revelados por los casos de prueba cuando se aplica a sistemas reales para pruebas de caja negra.

Proposición 4: La afinación automática de parámetros («*Automated Parameter Tuning*») mejora los resultados de las búsquedas en términos de su efectividad y eficiencia.

Proposición 5: Después de la instalación de ETF, el tiempo y esfuerzo que toma su configuración para su aplicación a sistemas reales es conveniente para la industria en comparación con las técnicas actualmente utilizadas.

3.3. Procedimientos de Recolección de Datos

Los siguientes pasos comunes han sido definidos para llevar a cabo los estudios:

1. Instalación y configuración del ETF de acuerdo al manual de usuario [11]; mantenimiento de diarios de trabajo que deben contener las tareas (incluyendo su fecha, duración y descripción) realizadas para configurar el ETF.
2. Implementación de componentes específicos (especificación de los individuos y drivers de pruebas) para los casos de estudio. Los diarios de trabajo deberán ser mantenidos para posibilitar la estimación del esfuerzo necesario.
3. Definir, refinar, e implementar la función de adecuación y validar su conveniencia para quebrar el requisito. Los diarios de trabajo deberán ser mantenidos.
4. Ejecutar cada búsqueda 30 veces para otorgar significado estadístico.
5. Entrevistas relacionadas con la idoneidad y aceptabilidad en el entorno industrial específico. Estas entrevistas son informales y no intentan realizar pruebas estadísticas sobre los datos porque los ejemplos de Daimler y BMS no son representativos para la población de interés. Sin embargo, estas entrevistas son interesantes [20]

dado que su objetivo es conseguir las experiencias que los profesionales tuvieron cuando aplicaron las técnicas y herramientas para pruebas evolutivas. Se fomentará que los entrevistados elaboren áreas importantes para ellos y que puedan desviarse de las preguntas de la entrevista. Por este motivo, las entrevistas son altamente interactivas, es decir, los entrevistadores pueden clarificar preguntas a los entrevistados e investigar respuestas inesperadas, y también pueden ganar la confianza de un entrevistado para mejorar la calidad de sus respuestas.

3.4. Procedimientos de Análisis de Datos

Los estudios se realizarán para controlar las variables independientes y medir su efecto en las variables dependientes, que han sido elegidas para dar respuesta a las proposiciones planteadas.

Variables Independientes

- Sistemas reales con su complejidad inherente.
- El conjunto de parámetros usados para configurar el generador evolutivo ETF (detalles de los parámetros pueden ser consultados en [10]). Se tienen cuatro configuraciones de parámetros diferentes asociadas a diferentes versiones de ETF: *ETF_Random* usa búsquedas aleatorias en vez de búsquedas evolutivas; *ETF_Default* usa un conjunto de parámetros seleccionado por defecto por el generador evolutivo; *ETF_Manual* usa parámetros configurados en base a la experiencia del probador según la observación del rendimiento de diferentes conjuntos de parámetros; y *ETF_Automated* usa técnicas de afinación automática de parámetros, las cuales son ejecutadas por el generador evolutivo para elegir el “mejor” conjunto de parámetros.

Variables Dependientes

- a. Número de casos de prueba evaluados. Esto es equivalente al número de veces que el SUT es ejecutado para un requisito específico y al número de evaluaciones de la función de adecuación.
- b. Número de casos de prueba inválidos que fueron generados.
- c. Número de errores encontrados.
- d. Progreso de los valores de adecuación.

- e. Tiempo y esfuerzo necesitado para encontrar un conjunto de parámetros apropiado para el generador evolutivo - medido según el tiempo usado por los desarrolladores.
- f. Número de versiones o refinamientos de funciones de adecuación que respectivamente fueron realizados.
- g. Tiempo necesitado para definir y refinar las funciones de adecuación - medido según el tiempo usado por los desarrolladores.
- h. Tiempo y esfuerzo necesitado para instalar el ETF - medido según el tiempo usado por los desarrolladores.
- i. Tiempo y esfuerzo necesitado para personalizar el ETF para un objetivo de prueba específico, es decir, desarrollo del archivo XML para la especificación de los individuos, plugins de eclipse para el driver de pruebas y la función de adecuación - medido según el tiempo usado por los desarrolladores.
- j. Líneas de código de los componentes específicos que necesitan ser integradas en el ETF para las pruebas funcionales.
- k. Aplicabilidad y aceptabilidad en entornos industriales.

Por cada sistema industrial seleccionado, los estudios serán ejecutados con las cuatro versiones de ETF. Así, en total se realizarán 8 estudios.

3.5. Amenazas a la Validez

La *Validez de la Construcción* está relacionada a la admisibilidad de declaraciones acerca de los fundamentos tomados como base para el estudio. En este sentido, posibles amenazas son:

- El punto en el cual los errores en los sistemas están relacionados a los requisitos que intentamos quebrar. Para mitigar este riesgo, los errores encontrados fueron investigados para evaluar si efectivamente quiebran el requisito que se está probando.
- Las entrevistas no son representativas para la población de interés. Por esta razón, estas entrevistas son informales y no intentamos realizar pruebas estadísticas sobre los datos. Los datos son usados para encontrar errores en la herramienta, áreas que necesitan ser mejoradas, o extensiones necesarias para entornos industriales específicos.
- El tiempo de desarrollo medido para algunas variables depende de la experiencia y el nivel de educación de la gente involucrada. Para

mitigar esta amenaza, ambas empresas han seleccionado personas con perfiles similares para los casos de estudio. Este perfil corresponde a personas que normalmente ejecutan pruebas de manera automática.

- El generador de señales de ETF puede no permite la generación de señales totalmente realistas. Para mitigar este riesgo, la cantidad inválida de casos de prueba es medida en la variable dependiente *b*.

La *Validez Interna* está relacionada a la interpretación de los resultados, es decir, describe la certeza con la que los cambios en las variables dependientes son atribuidos a los cambios en las variables independientes. En este sentido, posibles amenazas son:

- La experiencia de la gente usando el ETF con computación evolutiva o pruebas evolutivas. La gente con experiencia en afinamiento de algoritmos evolutivos ejecutará mejor las pruebas que personas que no tienen esta experiencia, ya que seleccionan mejor los parámetros que pueden conducir una búsqueda más eficiente. Esta amenaza es mitigada seleccionando a personas que tengan conocimientos básicos sobre computación evolutiva.
- Inestabilidades y errores del ETF en fases tempranas del desarrollo pueden dificultar la ejecución de los estudios.
- La aleatoriedad intrínseca de los algoritmos evolutivos. Esta amenaza es mitigada ejecutando cada caso de estudio varias veces.

La *Validez Externa* puede influenciar el punto en el cual las conclusiones pueden ser generalizadas. En este sentido, posibles amenazas son:

- La representatividad de los sistemas seleccionados como casos de estudio, ya que ambos corresponden al dominio automoción. Esto puede causar que los resultados sean válidos sólo en este contexto industrial. Repetir los casos de estudio en otros dominios puede entregar más información acerca de las generalizaciones de los resultados.
- Las señales de entrada generadas pueden ser estrictamente sintéticas (sin ruido) y podrían no ser generalizables para sistemas reales.
- La cantidad de repeticiones necesarias para obtener significado estadístico puede tomar demasiado tiempo en entornos industriales.

4. Recolección de Datos

Para obtener una línea base de comparación, un número aleatorio de búsquedas fue realizado sin usar ningún tipo de optimización en el generador (*ETF_Random*). Después, las búsquedas fueron realizadas usando los parámetros especificados por defecto (*ETF_Default*). Posteriormente, los resultados de los estudios fueron optimizados adaptando los parámetros de optimización al problema específico seleccionado (*ETF_Manual*). Finalmente, un conjunto de búsquedas fueron llevadas a cabo usando *ETF_Automated*, que automatiza la costosa tarea de afinar los parámetros de optimización durante el progreso hacia la optimización final. Los parámetros que deben ser especificados son aquellos que limitan el tamaño de la población y el número de las generaciones evolutivas durante la optimización.

Durante estas actividades, se recolectaron los siguientes datos para las variables dependientes a hasta *j* y los sistemas ACC (Adaptive Cruise Control) y ABS (Anti-lock Braking System):

	ACC	ABS
<i>a</i>	1,2 millones de pruebas ejecutadas	150.000 millones de pruebas ejecutadas
<i>b</i>	0	0
<i>c</i>	0	0
<i>d</i>	Vea Figura 1	Vea Figura 2
<i>e</i>	1 persona-mes	1 persona-mes
<i>f</i>	2	7
<i>g</i>	0,5 persona-mes	1 persona-mes
<i>h</i>	2 horas	2 horas
<i>i</i>	3 persona-mes	4 persona-mes
<i>j</i>	365	5195

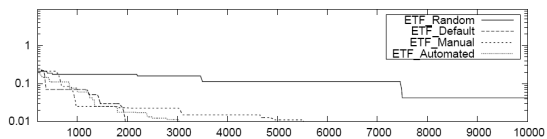


Figura 1: ACC

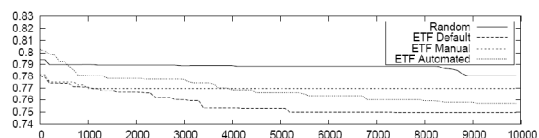


Figura 2: ABS

5. Evaluación

En esta sección se presentan respuestas a la pregunta de investigación planteada (*¿Las pruebas evolutivas de caja negra son escalables a las prácticas reales en la industria?*) utilizando el conjunto de proposiciones presentadas en la Sección 3.2 y las variables dependientes que fueron medidas.

Proposición 1: Respecto a las pruebas funcionales de caja negra, *no* es posible usar el entorno ETF para pruebas evolutivas sin conocimientos detallados sobre computación evolutiva para buscar datos de prueba interesantes, ya que es necesario diseñar funciones de adecuación sofisticadas. La función de adecuación es un factor crucial para el éxito de búsquedas evolutivas y, por lo tanto, para el proceso de generación de datos de pruebas evolutivas en general. La falta de experiencia en el diseño de funciones de adecuación convenientes puede fácilmente producir errores como la creación de óptimos locales o introducir riesgos similares que dificultan la búsqueda de los datos de prueba deseados.

Respecto a los parámetros del generador evolutivo, es necesario proveer consejos acerca de cómo realizar el afinamiento manual del transformador evolutivo utilizando computación evolutiva experta. Sin embargo, si el afinamiento manual no es necesario, entonces, se necesita poco conocimiento para buscar datos de pruebas interesantes, como se puede ver en los resultados de *ETF_Default* que no son tan malos en comparación con *ETF_Manual*.

Proposición 2: Las Figuras 1 y 2 muestran que el ETF genera un gran conjunto de casos de pruebas orientados al problema. Las pruebas aleatorias fueron incapaces de alcanzar resultados comparables con las generaciones evolutivas. Como era esperado y de acuerdo a sus capacidades usuales, las pruebas aleatorias gradualmente encontraron ligeramente mejores soluciones durante la optimización sin alcanzar áreas del espacio de búsqueda cercanos a la solución óptima, sin mencionar la solución global. En contraste, el generador evolutivo, es decir, *ETF_Default*, *ETF_Manual* y *ETF_Automated* mostró un comportamiento que se acerca a la solución. Después de alcanzar áreas interesantes, ellos tratan de acercarse al óptimo generando

soluciones ligeramente diferentes que son también casos de pruebas altamente orientados al problema, porque su proximidad a las buenas soluciones ya se ha encontrado.

Proposición 3: Los sistemas elegidos por los socios industriales para investigar la escalabilidad de las pruebas evolutivas provienen de su producción en serie. Por lo tanto, durante el estudio fue poco probable encontrar errores. Así, no es posible evaluar esta proposición en el contexto los casos de estudio seleccionados. Esperamos que repeticiones de los casos de estudio en trabajos futuros posibiliten comentar sobre este aspecto.

Proposición 4: El afinamiento automático de parámetros funciona bien con respecto a la efectividad y eficiencia dada, obteniendo, en promedio, los mismos resultados que el afinamiento manual, pero relevando al probador de la tediosa y difícil tarea de especificar manualmente los parámetros del generador evolutivo que resultó ser considerable (entre 0,5 y 1 persona-mes - vea la variable dependiente e). Así, el afinamiento automático de parámetros contribuye a la aplicabilidad en la industria, donde asignar recursos humanos al afinamiento es más caro que utilizar recursos computacionales. Sin embargo, se debe tener en cuenta que si la ejecución del SUT toma una gran cantidad de tiempo (que es frecuentemente el caso con pruebas funcionales de caja negra), puede también requerir muchos recursos computacionales.

Proposición 5: Después de la instalación del entorno ETF para pruebas evolutivas, el tiempo y el esfuerzo tomado en su configuración para aplicarlo a las prácticas reales que son realizadas para la ejecución de pruebas funcionales fue encontrado razonable dentro de los entornos industriales estudiados. Si bien la creación de la función de adecuación es una tarea que consume tiempo y requiere conocimientos específicos de computación evolutiva, los socios industriales encontraron que este esfuerzo es compensado por la gran cantidad de casos de prueba orientados al problema generados y después evaluados automáticamente.

Cuando la función de adecuación ha sido diseñada, el proceso de generación de datos es llevado a cabo de manera completamente automática sin ningún esfuerzo humano. Mientras tanto el SUT es ejecutado con un número de datos

de prueba específicos del problema excepcionalmente grande y altamente relevantes, los cual no podría ser posible cuando se hace manualmente. Como puede ser visto en los datos recolectados, el SUT ha sido ejecutado 1.2 millones de veces o con 150.000 entradas diferentes, respectivamente.

Algunos problemas de usabilidad e ideas para mejoras adicionales obtenidos en las entrevistas informales estuvieron relacionados a la necesidad de interacción más directa del usuario con las pruebas funcionales de ETF. Por ejemplo, la página de preferencia provista para configurar manualmente los parámetros del generador evolutivo puede ser mejorada. Además, proveer guías sobre cómo desarrollar la función de adecuación serían muy apreciadas.

6. Conclusiones

En este artículo hemos mostrado que la implementación del ETF para las pruebas funcionales evolutivas es escalable en el dominio automoción. Los socios industriales se mostraron positivos acerca de la eficiencia de la herramienta y encontraron los resultados interesantes. El afinamiento automático de parámetros generalmente no requiere conocimientos avanzados sobre computación evolutiva. Sin embargo, para definir y refinar una función de adecuación apropiada, es necesario tener habilidades de computación evolutiva y debe ser factible dedicar una cantidad de tiempo significativo para realizar esta tarea. Aunque en algunos casos se compensa con los resultados, todavía es un factor importante que influye en la aceptabilidad total de las pruebas funcionales evolutivas en la industria.

Para una aceptación más amplia en entornos industriales es importante automatizar el afinamiento de parámetros para ofrecer la posibilidad de que probadores inexpertos usen pruebas evolutivas y obtengan resultados interesantes. Lindlar [7] presenta un enfoque para simplificar el proceso de diseñar espacios de búsqueda y funciones de adecuación para las pruebas funcionales evolutivas, y de esta manera incrementar su aceptabilidad.

Además, la difícil tarea de diseñar una función de adecuación conveniente puede ser apoyada usando un enfoque basado en asistentes (wizards) o preguntas/respuestas. Probadores más avanzados demandarán la combinación de diferentes

objetivos de pruebas en una búsqueda (ej. pruebas funcionales combinadas con pruebas del tiempo de respuesta), también llamadas búsquedas multi-objetivos. Distintas estrategias para sembrar datos de prueba [8] y paralelizar las pruebas son requeridas (ej. usando objetivos de pruebas múltiples y distribuidas) para diseñar una función de adecuación.

Un conjunto común de guías de desarrollo acerca de la utilización de las pruebas evolutivas como técnica de optimización podría incrementar la aceptación en la industria de estas pruebas, ya sea para usuarios medios y avanzados. Existe una necesidad real para sistemas embebidos de tener guías sobre qué técnicas de pruebas usar para diferentes objetivos de pruebas, diferentes niveles de pruebas, o diferentes fases del desarrollo de software; cuándo estas técnicas contribuyen con la fiabilidad general y la dependencia del sistema embebido; y qué tan eficiente y usable es su aplicación. Estas guías no existen para técnicas de pruebas tradicionales, e incluso se tiene menos conocimiento sobre pruebas evolutivas. Estudios empíricos son esenciales para sentar las bases para estas guías y así poder integrarlas en una estrategia general de *Pruebas y Aseguramiento de Calidad* para sistemas embebidos. Para esto, contar con un repositorio central de sistemas embebidos de ejemplo que puedan actuar como punto de referencia para la evaluación de técnicas de pruebas evolutivas para propiedades funcionales y no funcionales sería de ayuda.

Así, esperamos que el trabajo de este artículo constituya el primer paso para construir una familia de casos de estudio, como ha sido recomendado por Basili et. al. [13]. Los casos de estudio deben investigar la aplicabilidad de las pruebas evolutivas en la industria, y proveer señales claras acerca de las barreras que pueden evitar su amplia adopción en la industria. Finalmente, invitamos a otros investigadores a repetir nuestro estudio en otros entornos industriales, para posteriormente construir un repositorio con los resultados obtenidos.

Agradecimientos

Este trabajo ha sido soportado por el proyecto Europeo IST-33472. Queremos agradecer por su soporte y ayuda a Andreas Windisch y Felix Lindlar de Daimler; Joachim Wegener y Peter Kruse de Berner&Matter; Mark Harman, Kiran Lakhotia y Youssef Hassoun de Kings College London; Marc Schoenauer y Luis da Costa de

INRIA; Jochen Hänsel de Fraunhofer; Dimitar Dimitrov y Ivaylo Spasov de RILA; y Dimitris Togias de European Dynamics.

Referencias

- [1] Bühler O, Wegener J. Automatic testing of an autonomous parking system using evolutionary computation. Proceedings of SAE 2004 World Congress, 2004; pages 115–122.
- [2] Windisch A, Lindlar F, Topuz S, Wappler S. Evolutionary functional testing of continuous control systems. Proceedings of the 11th GECCO conference, ACM, 2009; 1943–1944.
- [3] Windisch A, Al Moubayed N. Signal generation for search-based testing of continuous systems. Proceedings of the 2nd International Conference on Software Testing, Verification, and Validation Workshops, IEEE Computer Society, 2009; 121–130.
- [4] Hyunsook ES D, Rothermel G. Infrastructure support for controlled experimentation with software testing and regression testing techniques. Proceedings of the International Symposium On Empirical Software Engineering, ISESE '04, ACM, 2004; 60–70.
- [5] Juristo, M, Vegas S. Reviewing 25 years of testing technique experiments. Journal of Empirical SW Eng 2004; 9(1):7–44.
- [6] Briand LC. A critical analysis of empirical research in software testing, ESEM 2007, 1–8.
- [7] Lindlar F. Search-based functional testing of embedded software systems. Doctoral Symposium in conj with IEEE ICST 2009.
- [8] Arcuri A, White DR, Clark J, Yao X. Multi-objective improvement of software using co-evolution and smart seeding. Proceedings of SEAL'08, LNCS, 5361, 2008; pp 61–70.
- [9] Eclipse. <http://www.eclipse.org>. Last accessed 2010-02-03.
- [10] GUIDE. <http://gforge.inria.fr/projects/guide/>. Last accessed 2010-02-03.
- [11] ETF user manual. <http://www.evotest.eu>. Last accessed 2010-02-03.
- [12] Beizer B. Software Testing Techniques. International Thomson Computer Press, 1990.
- [13] Basili VR, Shull F, Lanubile F. Building knowledge through families of experiments. IEEE TSE 1999; 25(4):456–473.
- [14] Mueller F, Wegener J. A comparison of static analysis and evolutionary testing for the verification of timing constraints. Proceedings of the Fourth IEEE RTAS, IEEE, 1998; 144.
- [15] Sthamer H, Wegener J. Using evolutionary testing to improve efficiency and quality in software testing. Proceedings of 2nd Asia-Pacific Conf on Software Testing, 2002.
- [16] Gros HG. Evaluation of dynamic, optimisation-based worst-case execution time analysis. Int. Conference on Information Technology: Prospects and Challenges in the 21st Century, vol. 1, 2003; pp. 8–14.
- [17] Wegener J, Bühler O. Evaluation of different fitness functions for the evolutionary testing of an autonomous parking system. Proceedings of GECCO, 2004; 1400–1412.
- [18] Baresel A, Pohlheim H, Sadeghipour S. Structural and functional sequence test of dynamic and state software with evolutionary algorithms. GECCO2003; 2428–2441.
- [19] Windisch A, Lindlar F, Topuz S, Wappler S. Evolutionary functional testing of continuous control systems. GECCO, 2009; 1943–1944.
- [20] Lethbridge TC, Sim SE, Singer J. Studying software engineers: Data collection techniques for software field studies. Empirical Softw. Engg. 2005; 10(3):311–341.