# Especificación y Verificación Formal de Sistemas Críticos en el Instituto de Computación de la Universidad de la República (Uruguay)

Gustavo Betarte Carlos Luna Luis Sierra
Instituto de Computación, Facultad de Ingeniería
Universidad de la República

J. Herrera y Reissig 565, 5to. Piso, CP 11300
Montevideo, Uruguay
(+598) 2 7114244

[gustun, cluna, sierra]@fing.edu.uy

## RESUMEN
Este artículo presenta al Instituto de Computación (InCo) de la Facultad de Ingeniería (FING) de la Universidad de la República (Uruguay) y en particular a su grupo de Métodos Formales. Se describen las actividades principales de los miembros de dicho grupo, tanto de formación de recursos humanos como de investigación, relacionadas con la especificación y verificación formal de sistemas computacionales críticos.

## Categorias y Descriptores
D.2.4 [**Software**]: Software/Program Verification – *formal methods, correctness proofs, model checking.*

## Términos Generales
Theory, Verification, Security

## Palabras Claves
Métodos Formales, Verificación de Sistemas Embebidos, Especificación y prueba asistida de propiedades.

## 1. INTRODUCCIÓN
Este artículo presenta al Instituto de Computación (InCo) de la Facultad de Ingeniería de la Universidad de la República (Uruguay). Se describen las actividades del Instituto, su rol en la formación de profesionales e investigadores, sus vínculos con la industria y las actividades de investigación que se realizan en torno a los grupos que lo constituyen. En particular se detallan las actividades desarrolladas por el Grupo de Métodos Formales (GMF) en relación a la especificación y verificación de sistemas críticos.

En primer lugar, y en relación a la formación de recursos humanos, tres asignaturas opcionales de grado y posgrado a cargo del GMF son descriptas. Las mismas abarcan las siguientes temáticas: especificación y verificación formal de programas, modelos y verificación de sistemas de tiempo real y fundamentos de la seguridad informática. Luego se presentan las actividades más recientes de investigación en las que están involucrados los investigadores del equipo y los proyectos nacionales e internacionales en curso. En particular se describen las líneas actuales de trabajo en torno a las cuales existe interés en establecer cooperaciones con otros equipos, tanto del sector académico como del industrial.

La organización del resto del documento es como sigue. La sección 2 presenta al InCo. La sección 3 al GMF, su actividad en enseñanza, en investigación y su relación con la industria. Finalmente, la sección 4 describe el proyecto de investigación en curso que nuclea a los autores de este artículo y se detallan las líneas de trabajo en dicho contexto.

## 2. PRESENTACIÓN DEL INCO
El Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República (Uruguay) es responsable de la formación académica, la investigación y la extensión en el área de la Computación.

En este papel es el principal responsable de la carrera de Ingeniería en Computación cuyo objetivo es la formación de profesionales de alto nivel capaces de desarrollar su actividad en el medio. Es importante resaltar que la carrera de Ingeniería en Computación es la más numerosa de las impartidas por la Facultad de Ingeniería, teniendo un ingreso promedio de 450 estudiantes al año.

Asimismo, el InCo desarrolla una intensa actividad de posgrado, es responsable de la Maestría y Doctorado en Informática (programa desarrollado en conjunto con el PEDECIBA[1]-Informática), y también del programa de posgrado profesional y formación permanente que incluye un Diploma de especialización, una Maestría en Ingeniería en Computación y cursos de Actualización Profesional (organizados en torno al Centro de Posgrados y Actualización Profesional del InCo).

El Instituto cuenta con un plantel de investigadores de alrededor de cuarenta docentes con formación de posgrado (más de una decena de ellos con título de doctorado, y los restantes con maestría). En total, son aproximadamente 120 docentes.

[1] Programa de Desarrollo de las Ciencias Básicas, Uruguay.

El InCo mantiene contactos con la industria a través de proyectos conjuntos con empresas privadas y estatales. A nivel de investigación mantiene contactos con equipos europeos, latinoamericanos y canadienses.

Las actividades de investigación científica desarrolladas en el instituto abarcan desde los fundamentos teóricos de la Ciencia de la Computación hasta aplicaciones tecnológicas de la misma. Análisis de Algoritmos, Métodos Formales, Bases de Datos, Ingeniería de Software, Investigación Operativa, Tratamiento de Lenguaje Natural, Sistemas de Información Geográfica, Reconocimiento de Imágenes, Modelos Numéricos, Informática Educativa y Seguridad Informática son algunas de las temáticas en las que trabajan los diferentes grupos de investigación.

## 2.1 Estructura Organizativa

El InCo se estructura en Grupos, los cuales desarrollan actividades de enseñanza, investigación y asesoramiento en áreas temáticas. La junta de los jefes de grupo (entendiéndose por grupo en este caso, aquellos con actividades probadas de investigación) con el Director del Instituto conforman el órgano llamado Dirección Ampliada, en el cual se discuten y evalúan las estrategias docentes del Instituto que luego se encauzarán por la Comisión de Instituto. Además existe una instancia de discusión y planteo de estrategias políticas y posicionamiento del Instituto en todos los niveles de actividad del mismo. Esta instancia es la conformada por representantes estudiantiles, el gerente de RRHH, el encargado de enseñanza y el Director del Instituto.

En forma ortogonal a los grupos se organizan las tareas de Enseñanza de Grado y de Gestión de RRHH. Estas son coordinadas por docentes dependientes directamente de la Dirección del Instituto.

# 3. EL GRUPO DE MÉTODOS FORMALES

Los autores de este artículo pertenecen al grupo de ***Métodos Formales*** (GMF) del InCo. Este grupo forma parte del ***Laboratorio de Ciencia de la Computación*** (LCC).

El LCC nuclea a un número importante de investigadores y docentes del InCo, muchos de ellos retornados al Instituto tras haber finalizado sus respectivos estudios doctorales en el exterior. Las principales áreas de trabajo del LCC son:

- Análisis de Algoritmos, Combinatoria y Criptografía
- Lenguajes de Programación y Compiladores
- Computación Gráfica
- Teoría de Tipos; Editores de Prueba
- Lógicas de la Programación
- Cálculo y Transformación de Programas
- Programación Genérica
- Programación Funcional
- Verificación de Sistemas de Tiempo Real
- Orientación a Objetos y Arquitectura de Software
- Sistemas Embebidos; Tarjetas Inteligentes (Java Cards)
- Especificación Formal; Semántica de Lenguajes de Programación

Las áreas de interés del GMF son la teoría y aplicación de métodos formales con especial énfasis en el estudio de formalismos y herramientas para la especificación, desarrollo y verificación de sistemas computacionales. Las principales áreas de investigación del grupo son: calidad de software (tecnologías para el desarrollo de programas de corrección certificada), asistentes de prueba basados en Teoría de Tipos, métodos algebraicos y co-algebraicos para la construcción y transformación de programas, programación genérica y, la verificación formal de sistemas reactivos y de tiempo real.

## 3.1 Actividad de Enseñanza

Los miembros del GMF desarrollan una intensa actividad de enseñanza siendo responsables del dictado de cursos de grado de la carrera Ingeniería en Computación de FING y de cursos de posgrado del programa de Maestría y Doctorado del PEDECIBA Informática.

Gustavo Betarte y Carlos Luna son los docentes responsables de la asignatura de grado opcional y de posgrado *Construcción Formal de Programas en Teoría de Tipos* (CFPTT) [8]. La misma es dictada desde el año 2000 en versiones ligeramente diferentes en el InCo y en tres oportunidades en la Universidad Nacional de Rosario (UNR, Argentina). Asimismo, se presentaron versiones cortas en tres escuelas de ciencias informáticas en Argentina (Rio'200', ECI'2001 y Rio'2004). Actualmente la asignatura se está dictando en paralelo en el InCo y en la UNR, en modalidad semi-presencial.

Los objetivos de CFPTT son: (1) Presentar a la Teoría de Tipos como lógica de programación y familiarizar al estudiante con ambientes de desarrollo de programas basados en este formalismo. (2) Iniciar al estudiante en el uso de métodos formales para la especificación, producción, derivación y verificación de software correcto por construcción. (3) Mostrar la utilidad de editores de pruebas basados en teoría de tipos, en especial Coq [13], para la especificación y verificación de aplicaciones industriales y académicas.

El temario incluye: (1) Asistentes de pruebas para lógicos y matemáticos: Una presentación formal de la lógica proposicional y de primer orden. (2) Asistentes de Pruebas para programadores: El calculo lambda como lenguaje de programación funcional. (3) Identificación de pruebas y programas: Isomorfismo de Curry Howard. (4) Recursión: Definiciones Inductivas, Principios de Inducción, Esquemas de Recursión. (5) Extracción de programas a partir de pruebas. Construcción de Pruebas a partir de programas. (6) Construcción de programas certificados usando Coq.

En torno a las temáticas de CFPTT se han desarrollado, y se están desarrollando actualmente, varios proyectos de investigación y colaboraciones con instituciones regionales e internacionales así como trabajos finales de carreras de grado y de posgrado, tanto en Uruguay como en Argentina.

Ver http://www.fing.edu.uy/inco/grupos/mf/TPPSF/ y [8] por mayor información.

Las tareas de enseñanza de Luis Sierra comprenden un curso básico de Lógica, así como cursos avanzados que involucran modelado de sistemas y uso de herramientas automáticas. Los

cursos avanzados de los que ha sido responsable desde 2002 son Verificación y Lógica (VL) y Modelos y Verificación de Sistemas de Tiempo Real (MVSTR). Estos cursos presentan al estudiante el uso de la lógica y los métodos formales como herramientas de verificación de sistemas informáticos. Son sus objetivos la formación en el modelado de sistemas y propiedades, el método de verificación de modelos, y el uso de herramientas automáticas y semiautomáticas. A lo largo de los cursos referidos los estudiantes han experimentado el uso de herramientas como Kronos, UppAal, Murphi, Spin, entre otras (ver http://www.fing.edu.uy/~sierra/TiempoReal).
Luis Sierra ha participado, además, como docente en el curso opcional de grado Taller de Firmware dictando clases sobre modelado y verificación de sistemas embebidos. Por información adicional sobre los cursos desarrollados por Luis Sierra consultar http://www.fing.edu.uy/~sierra/.

Gustavo Betarte es el responsable de la asignatura opcional de grado y de posgrado *Fundamentos de la Seguridad Informática* (FSI). Ésta fue dictada por primera vez en el primer semestre (Marzo-Junio) de 2007.

Los objetivos de FSI son capacitar al estudiante para: (1) Asimilar la seguridad informática como un conjunto de metodologías. (2) Analizar la seguridad de una red o sistema informático, identificando los puntos débiles de la misma para su protección. (3) Conocer los principales ataques de los que puede ser objeto un sistema informático, así como los posibles métodos de protección, detección y políticas de seguridad que permitan evitar el daño al sistema o minimizar su repercusión. (4) Entender el funcionamiento de diferentes protocolos criptográficos que se utilizan en la actualidad. (5) Conocer los sistemas de autenticación más importantes identificando sus características principales.

Referimos a la información publicada en http://www.fing.edu.uy/inco/cursos/fsi/ por mayor detalle.

## 3.2 Actividad de Investigación

El trabajo de investigación más reciente del GMF en relación a la actividad de verificación de sistemas computacionales concierne:

- La especificación y verificación formal de arquitecturas de seguridad de sistemas embebidos, en particular aquellos basados en la tecnología Java, como Java Card y la plataforma J2ME
- Formalizaciones en Teoría de Tipos de grafos temporales para el modelado de sistemas reactivos y de tiempo real,
- Formalizaciones de las lógicas CTL y TCTL y su utilización para razonar sobre este tipo de sistemas críticos,
- Desarrollo de metodologías que combinan el uso de model-checkers (como Kronos) y asistentes de pruebas (como Coq) para el análisis formal de sistemas reactivos y de tiempo real

Actualmente, el equipo se encuentra abocado al desarrollo de actividad de investigación enmarcada dentro de los objetivos del proyecto de I+D *STEVE* (Seguridad a Través de Evidencia Verificable). Este proyecto cuenta con el apoyo financiero del programa internacional de cooperación científico-tecnológica STIC Amsud (http://www.sticamsud.org) y del programa PDT (Proyecto de Desarrollo Tecnológico) del CONICYT (Consejo Nacional de Investigación en Ciencia y Tecnología) uruguayo. En la sección 4 se describe en mayor detalle este proyecto.

## 3.3 Experiencia en el Sector Industrial

La empresa uruguaya CCC se dedica al desarrollo y manufactura de sistemas médicos de tiempo real, en particular marcapasos. Los autores de este artículo han mantenido reuniones periódicas con gente de CCC con el objetivo de verificar prototipos, donde el tiempo juega un rol importante. Luis Sierra participó en 2001/2002 del proyecto *Verificación Formal Automatizada de un Marcapasos*, bajo la dirección de la Dra. Cristina Cornes, tomando de base modelos realizados para la empresa CCC. Recientemente esta empresa ha brindado su apoyo al proyecto PDT I+D *Verificación de Sistemas Críticos: de la Especificación al Código*, que están actualmente desarrollando Carlos Luna y Luis Sierra.

Desde abril de 2001 a octubre de 2004 Gustavo Betarte ocupó un cargo de *Ingénieur de Recherche et Développement* en la empresa *Trusted Logic* (http://www.trusted-logic.com), Versalles, Francia. Trusted Logic es una empresa francesa, creada en enero de 1999, especializada en el desarrollo de componentes de software seguros para sistemas embebidos basados en el lenguaje Java y en procesos de evaluación de seguridad de estos sistemas.

G. Betarte es redactor principal del compendio de Perfiles de Protección *Java Card System Protection Profile Collection*. Este trabajo fue desarrollado por Trusted Logic para la empresa Sun Microsystems Inc. Los perfiles de protección fueron certificados en septiembre de 2003 por la DCSSI (Direction Centrale de la Sécurité des Sistemes d'Information, Francia), uno de los más respetados centro de evaluación de la seguridad de sistemas IT. Los perfiles fueron certificados a nivel EAL4+ del estándar de evaluación conocido como los CC (Common Criteria for Information Technology Security Evaluation, ISO/IEC 15408).

Asimismo, G. Betarte fue miembro del equipo desarrollador de la metodología de seguridad CC EAL7 aplicada al Java Card System. Este trabajo fue desarrollado por Trusted Logic para la filial Sema (ahora Gemalto) de la empresa Schlumberger y fue el responsable técnico de la generación, escritura y mantenimiento de la documentación asociada al proyecto. Esta documentación es el referente principal para el proceso de evaluación y certificación del proyecto. La metodología fue certificada por la DCSSI en Julio de 2003.

# 4. Proyecto I+D en curso: STEVE

Sistemas distribuidos, como Internet, redes bancarias, redes telefónicas y de vídeo digital se orientan a proveer servicios globales y en forma uniforme. Sin embargo, estos sistemas están compuestos de dispositivos autónomos que varían considerablemente en sus plataformas (sistemas operativos, protocolos de comunicación, bibliotecas) y recursos (memoria, alimentación de poder, conectividad).

Para garantizar un acceso global y uniforme es necesario extender estos dispositivos con componentes capaces de ejecutar los servicios requeridos. En este escenario, infraestructuras computacionales distribuidas no siempre pueden garantizar la separación entre aplicaciones no confiables y la plataforma fija y confiable sobre la que se ejecutan esas aplicaciones. Adicionalmente, en estas infraestructuras surgen problemas

relacionados con el desarrollo y mantenimiento de software implantado, como por ejemplo, poder garantizar la corrección de un componente que se ha obtenido a partir de la combinación de otros componentes existentes en la infraestructura.

El principal objetivo del proyecto STEVE es investigar la seguridad y fiabilidad en un modelo computacional, donde tanto las plataformas como las aplicaciones son dinámicas, de forma que componentes provistos por un agente externo puedan ser destinados a formar parte de la plataforma o ejecutar una aplicación de forma segura. Concretamente, lo que se quiere es elaborar y concebir mecanismos que ayuden a los desarrolladores de software a construir sistemas confiables a partir de componentes existentes así como infraestructuras que garanticen al usuario final que el software que este utiliza es seguro y confiable.

Los objetivos del proyecto STEVE se pueden dividir en dos ejes complementarios de investigación:

1. Especificación y Verificación de Sistemas de Componentes
2. Seguridad a Través de Evidencia Verificable

El problema que en última instancia se está abordando en este proyecto es el del desarrollo de mecanismos que permitan determinar con certeza si es seguro para un sistema de computación ejecutar código o aceptar un resultado suministrado por una fuente no confiable. El problema es altamente relevante en el contexto de la computación distribuida, en particular en la WWW.

La actividad desarrollada en el proyecto STEVE se enmarca a su vez en la del proyecto de colaboración internacional STIC Amsud "ReSeCo: Reliability and Security of Distributed Software Components".

El proyecto ReSeCo tiene como objetivo fundamental incentivar la colaboración entre la comunidad científica, e industrial, de Francia y de países Sudamericanos (Argentina, Chile y Uruguay). Participan los grupos Everest y Oasis del INRIA Sophia-Antipolis (Francia), el grupo de Ciencia de la Computación del FAMAF de la Universidad Nacional de Córdoba (Argentina), la Escuela de Informática de la Universidad Diego Portales y el Departamento de Ciencia de la Computación de la Universidad de Chile (Chile), y el grupo de Métodos Formales del InCo.

A continuación se describe en mayor detalle el marco conceptual y el enfoque abordado en cada uno de los ejes de investigación mencionados arriba así como las actividades que se están desarrollando.

## 4.1 Especificación y Verificación de Sistemas de Componentes

En los últimos años el uso de dispositivos portátiles tales como teléfonos celulares y asistentes de datos se ha popularizado a nivel mundial. Este tipo de dispositivos tienen fines y características que son en esencia diferentes de los que tienen, por ejemplo, las computadoras portátiles o de escritorio. En el desarrollo de sistemas y aplicaciones para dispositivos móviles se hace necesario, por lo tanto, el uso de una tecnología especializada que tenga en consideración aspectos tan disímiles como las limitaciones de los recursos computacionales, la heterogeneidad de las plataformas de hardware, las comunicaciones móviles y, en especial, la preservación de la confidencialidad e integridad de los datos personales.

Considerando los objetivos definidos para este eje de investigación, se definió como línea de trabajo principal el estudio y la especificación formal y verificación de arquitecturas y políticas de seguridad para la carga y ejecución de código móvil en un sistema de cómputo de recursos limitados. En particular se ha tomado a J2ME (Java 2 Micro Edition) como plataforma objetivo del estudio. El *framework* de especificación formal a utilizar es el provisto por el Cálculo de Construcciones Inductivas (CIC) ([3],[11],[14]) y se utiliza a Coq como asistente mecánico para el desarrollo de las especificaciones y prueba de propiedades de las mismas.

### *4.1.1 La Plataforma J2ME - MIDP*

En la actualidad la gran mayoría de las plataformas tecnológicas para los dispositivos portátiles incorporan la tecnología Java para sistemas embebidos – *embedded systems* – denominada Java 2 Micro Edition (J2ME). La característica fundamental de esta edición, al igual que toda tecnología Java, es el uso de una máquina virtual, especialmente diseñada y adaptada para aprovechar al máximo los escasos recursos con los que cuentan los dispositivos portátiles.

La tecnología J2ME proporciona mecanismos integrales para garantizar propiedades de seguridad de los dispositivos; en particular, para dispositivos móviles, define un modelo de seguridad a nivel de aplicación que restringe el acceso de las aplicaciones a funciones consideradas potencialmente peligrosas.

J2ME ha sido diseñada para poder adaptarse a gran variedad de dispositivos. Esta flexibilidad se logra principalmente a través de un conjunto de configuraciones, perfiles y componentes opcionales, que pueden combinarse para construir un entorno de ejecución especializado para toda una variedad de dispositivos con recursos y funciones similares.

Una configuración J2ME define una plataforma mínima para toda una categoría de dispositivos con recursos computacionales similares. En particular, una configuración debe definir una máquina virtual y las características del subconjunto del lenguaje Java soportado, así también como una biblioteca de las funciones que los fabricantes de dispositivos y desarrolladores de aplicaciones pueden asumir como base para la categoría de dispositivos que tiene por objetivo. Una configuración provee de esta manera una interfaz independiente del hardware para el desarrollo de aplicaciones.

Un perfil se construye por sobre una configuración subyacente, atendiendo a las características comunes de una clase más restringida de dispositivos, con el fin de aumentar la compatibilidad e interoperabilidad de las aplicaciones. Un perfil usualmente define, entre otras cosas, un modelo de seguridad para aplicaciones, el ciclo de vida de las aplicaciones, los mecanismos de comunicación de red, y una biblioteca para el acceso a funciones propias de la clase de dispositivos para la que fue concebido.

La configuración Connected Limited Device Configuration (CLDC) [5] es una configuración de J2ME pensada para ser utilizada en dispositivos con procesadores poco potentes, memoria de trabajo limitada y capacidad para establecer comunicaciones de bajo ancho de banda. Esta configuración se complementa con el perfil Mobile Information Device Profile

(MIDP) para obtener un entorno de ejecución adaptado a dispositivos portátiles como teléfonos celulares, pagers y asistentes de datos personales.

En la primera versión de la especificación de MIDP (versión 1.0) [6], toda aplicación que no haya sido preinstalada en el dispositivo (e.g. por el fabricante), se ejecuta en un ambiente controlado denominado *sandbox* que prohíbe el acceso a aquellas funciones del dispositivo que se consideran potencialmente peligrosas. Una aplicación no confiable solamente podrá tener acceso a estas funciones con autorización explícita del usuario, y aún así no podría adquirir acceso ilimitado a algunas combinaciones de funciones que se consideran particularmente delicadas. Por ejemplo, cualquier aplicación que adquiera el derecho de acceso a la información personal almacenada en el dispositivo, y al mismo tiempo pueda establecer una conexión de datos, podría valerse de ambas funciones para enviar información confidencial a terceros. Este comportamiento se considera por lo general indeseable, y el modelo de seguridad protege al usuario impidiendo que inadvertidamente otorgue a una aplicación los derechos necesarios para practicarlo.

En la segunda versión de la especificación del perfil MIDP (versión 2.0) [7], se abandona el modelo *sandbox* en favor de un modelo menos restrictivo, basado en el concepto de *dominio de protección*. Un dominio de protección es una abstracción del contexto de ejecución de una aplicación, que determina el derecho de acceso a cada función protegida del dispositivo. En los dispositivos compatibles con MIDP 2.0, cada suite de aplicaciones instalada está asociada con un único dominio de protección. En cierto sentido, el conjunto de todos los dominios de protección del dispositivo define una matriz de control de acceso donde los sujetos son las aplicaciones en las suites asociadas a cada dominio, y los objetos son las funciones que deben protegerse.

La versión 3.0 de MIDP, todavía en estado draft, extiende el modelo de seguridad de la versión 2.0. En primer término se establece una política de seguridad a nivel de plataforma para proteger el acceso a interfases de usuario (APIs) y funciones sensibles, ya sea que pertenezcan al Perfil o a paquetes opcionales. En segundo lugar se establece un mecanismo de protección a nivel de aplicación para garantizar el acceso a código compartido (mediante LIBlets), a datos compartidos (mediante RMS) o a la comunicación entre aplicaciones (mediante el protocolo IMC).

A continuación se describe en detalle las líneas de trabajo que se están desarrollando.

### *4.1.2 Especificación y Verificación del Modelo de Seguridad de MIDP 2.0 y 3.0*

Dos de los autores de este artículo, G. Betarte y C. Luna han desarrollado, en colaboración con Santiago Zanella, miembro del equipo *Secure Distributed Computations and their Proofs* del INRIA-Microsoft Research Joint Laboratory, una especificación formal del modelo de seguridad de J2ME – MIDP 2.0. Este trabajo es reportado en [15], donde se describe en detalle la construcción, utilizando CIC, de un modelo abstracto del estado de un dispositivo y de los posibles eventos que inducen cambios en dicho estado. Asimismo, se demuestran formalmente ciertas propiedades que se considera deseable el modelo satisfaga, lo que permite comprobar que su definición formal permite razonar a un nivel de abstracción adecuado sobre las propiedades de seguridad que el mismo debe garantizar.

Como continuación de este trabajo, se está ahora desarrollando una especificación formal del modelo de seguridad que especifica la versión 3.0 de MIDP. Los aspectos novedosos de la especificación MIDP 3.0 que impactan en la formalización arriba descripta se refieren a la seguridad a nivel de aplicación basada en la *Palabra de Autorización*. La palabra de autorización es la base del mecanismo de seguridad a nivel de aplicación. En los protocolos donde es implementada se utiliza como elemento de verificación para autorizar el acceso a código, datos o comunicación entre MIDlets. El impacto de esta funcionalidad se refleja en los eventos instalación y solicitud de permisos de una MIDlet Suite.

En el evento instalación de una Suite, se debe considerar la autenticación del certificado del Autor y del certificado del Vendedor de la misma, los cuales se deben especificar por separado y no tienen porqué ser el mismo. Si alguna de estas dos validaciones no prosperan, entonces ni siquiera se considera la condición de compatibilidad entre los permisos solicitados por la Suite y los permisos establecidos por la política de seguridad, ya que la instalación es rechazada. Por otra parte, en caso de ser satisfactorias, el certificado del Vendedor es pieza clave para el mecanismo de seguridad basado en la Palabra de Autorización. Con la misma importancia debe considerarse el conjunto de certificados de Vendedores confiables que una MIDlet declara en su manifiesto. Este conjunto también es clave para el mecanismo de seguridad como se describe a continuación.

La solicitud de permisos debe reflejar la posibilidad de acceder a una conexión ofrecida por otra MIDlet, o a un evento de publicación de otra aplicación, o a la apertura de un registro compartido. Las palabras de autorización involucradas obligan a la validación del apodo (*alias*) declarado contra alguno de los conjuntos de certificados: la raíz del Dominio de Seguridad, los Vendedores confiables o los Vendedores autenticados.

La formalización del mecanismo de Palabras de Autorización, ortogonal respecto al mecanismo de seguridad basado en dominios de seguridad, permitirá demostrar lo que hasta ahora son presunciones de flaquezas en la especificación del mismo.

El objetivo de esta actividad es extender la formalización desarrollada para MIDP 2.0 con el comportamiento especificado en la versión 3.0 y derivar un prototipo correcto por construcción que implemente el procedimiento de instalación de una MIDlet suite en un dispositivo.

### *4.1.3 Especificación y Derivación de un Access Controller para J2ME -MIDP 2.0*

El lenguaje de programación Java considera como componentes básicos mecanismos que permiten garantizar que determinadas propiedades de seguridad sean satisfechas. En particular, el administrador de seguridad (*Security Manager, SM de aquí en más*) y el controlador de acceso (Access Controller, AC de aquí en más) son dos de los componentes sobre los que recae la tarea de controlar que las políticas de seguridad, ya sea a nivel de plataforma o de aplicación, sean aplicadas.

El SM es el componente que controla que todas las reglas definidas por la política de seguridad son respetadas por cualquier aplicación ejecutándose en la plataforma. Intercepta llamados a recursos sensibles y pasa los permisos del método o clase

llamadora y la acción a ser controlada al AC. Este último componente, a su vez, es el que computa las reglas de control de acceso y decide si el código llamador tiene el permiso para efectuar la acción sensible o no; si lo tiene, la acción es realizada; si no lo tiene, la acción es denegada y una excepción de seguridad es lanzada.

La política de seguridad es impuesta gracias a la colaboración entre el SM y al AC. El SM es el punto central del control de acceso mientras que el AC implementa los algoritmos de control de acceso necesarios para la implementación de la política de seguridad. Cuando una acción sensible es invocada por alguna aplicación el SM chequea la política de seguridad y determina si la acción invocada es permitida para su ejecución o no. En caso de que la acción no pueda ser ejecutada, el administrador de seguridad interrumpe la ejecución y lanza una excepción de seguridad. El chequeo de la política de seguridad para decidir el control de acceso es delegado al AC, que es responsable de permitir o denegar el acceso a las acciones sensibles.

Para la plataforma J2SE existe una especificación de alto nivel del algoritmo que ejecuta el AC para tomar una decisión de acceso. El procedimiento básico utilizado para la resolución de la decisión es lo que en la literatura se ha denominado Inspección de Pila (*Stack Inspection*) [4]. Sin embargo, no existe una especificación estándar para el algoritmo que implementa la decisión de acceso en la plataforma J2ME.

El objetivo principal de esta actividad obtener una especificación formal del algoritmo de decisión que implementa el controlador de acceso para J2ME y derivar un prototipo correcto por construcción que lo implemente.

### *4.1.4 Modelo de Permisos de MIDP: Especificación Formal, Verificación y Análisis Estático automatizado*

Los autores de este artículo, en colaboración con Santiago Zanella, están actualmente desarrollando una formalización en teoría de tipos, usando el asistente de pruebas Coq, de modelos de control de accesos para dispositivos móviles interactivos. El trabajo se focaliza en el modelo presentado en [1], donde se introduce un modelo de control de acceso para aplicaciones en las que el control del acceso a recursos involucra la interacción de usuarios para obtener los permisos necesarios. Este modelo es inspirado por, y extiende, la arquitectura de seguridad de Java MIDP utilizada en teléfonos móviles. La propuesta contempla permisos de control de acceso con multiplicidades asociadas, las que permiten modelar la utilización de un permiso un cierto número de veces.

Uno de los objetivos de esta actividad es probar formalmente que dada una solución al sistema de restricciones para un grafo de control de flujo de un programa, si en la solución no existen nodos 'error' (intentos por acceder a permisos no disponibles), entonces toda traza del programa es segura (*Safe*).

Un segundo objetivo es establecer un marco conceptual común en el cual puedan ser expresadas distintas variantes de modelos de control de accesos y analizar formalmente sus relaciones.

En [1] se presenta además un análisis estático para computar una aproximación a los pedidos necesarios en cada momento de la ejecución de un programa. Un tercer objetivo que nos planteamos es el de comparar el enfoque basado en análisis estático con la formalización desarrollada en teoría de tipos. Con ese fin hemos implementado un prototipo del analizador estático que se especifica en la referencia, así como una biblioteca de combinadores para construir nuevos análisis. El objetivo principal de esta línea de trabajo es profundizar en el entendimiento de cómo realizar distintas mejoras a la propuesta original de análisis de flujos de programas, partiendo ya sea desde un marco deductivo basado en teoría de tipos, o de la elaboración de nuevos análisis sugeridos a partir de diferentes casos de estudio.

## 4.2 Seguridad a Través de Evidencia Verificable

El abordaje elegido para este eje de investigación es el del llamado “código con prueba” (inglés: “proof carrying code”). En este caso, el proveedor del código ejecutable , debe suministrar junto con éste una prueba formal de que el código satisface una política de seguridad establecida por el sistema anfitrión. De este modo, el sistema anfitrión puede verificar, usando técnicas ya conocidas y eficientes, la corrección del código ejecutable (resultado) previamente a su ejecución (utilización). La idea original del código con prueba ha sido presentada en [9]. Desde entonces, se ha desarrollado esa línea principalmente en la dirección de garantizar la corrección de los generadores de código. Como trabajo reciente en esta dirección, se tiene por ejemplo [10].

En este proyecto interesa en particular focalizar la atención en el estudio y desarrollo de técnicas que permitan desarrollar un nuevo enfoque, que es una variante del arriba descripto, llamado “resultados con prueba” (inglés: “proof carrying result” (PCR)). Esta técnica requiere que el resultado de una computación delegada sea acompañada de un certificado que provea la evidencia suficiente de que la computación que generó el resultado es correcta. En un gran número de escenarios no es posible asumir una relación de confianza entre hosts que colaboran en forma distribuida para el procesamiento de datos. Entonces, el pedido por parte de un usuario *U* que un nodo remoto *R* ejecute una computación *c* tomando como argumentos el conjunto de valores *v* no puede asumir al recibir el resultado *r* que *R* está retornando el resultado de computar *c*, ni que *R* ha ejecutado el proceso *c* con el conjunto correcto de valores de entrada *v*, ni siquiera que *R* ha efectivamente utilizado a *c* como proceso. Para poder asegurar que el resultado *r* es correcto, la única solución válida para *U* es efectuar, bajo su responsabilidad, un proceso de validación del resultado retornado. Dado que estamos bajo las hipótesis de que *U* no puede ejecutar él mismo la computaciones en forma independiente y que tampoco cuenta con hosts confiables (por ejemplo porque los recursos para ejecutar los procesos localmente no están disponibles, ni tampoco cuenta con hosts confiables para efectuar las computaciones), *U* debe necesariamente apoyarse en el uso de un *verificador* que valide la corrección del resultado, y que este proceso pueda ser desarrollado localmente o en hosts confiables. La técnica de PCR tiene propósitos similares a los de *probabilistic result checking* [2], pero estos dos enfoques difieren en que mientras PCR puede llegar a requerir información adicional para poder asegurar la corrección del resultado, probabilistic result checking privilegia el proceso de verificación probabilística y por lo tanto no requiere ninguna información extra. En general el enfoque PCR permite que una contraparte no confiable pueda presentar información adicional que permita facilitar el proceso de verificación.

El desarrollo de PCR ha sido fuertemente motivado por la evolución en el diseño de algoritmos en disciplinas científicas y áreas de investigación que requieren un consumo computacional muy intenso, en espacio y tiempo, y por lo tanto necesariamente deben resolverse en forma distribuída (por ejemplo haciendo uso de Grids).

Esta misma necesidad de computaciones distribuidas, y de requerir una prueba de validez de los resultados ejecutados remotamente, se hace presente, aunque en una escala diferente, en el escenario de dispositivos computacionales de recursos limitados, como pueden ser los teléfonos celulares, las tarjetas inteligentes, y los sistemas embebidos en general. En este proyecto, los casos de estudio en los que se trabajará para experimentar con las técnicas de PCR resultantes que se desarrollarán pertenecerán principalmente a esta clase de dispositivos.

### *4.2.1 Unmarshaling Seguro de Tipos Abstractos de Datos usando Proof Carrying Result*

Acute [12] es una extensión de ML que provee mecanismos robustos y seguros para desarrollar y ejecutar programas creados en forma separada. En particular, el lenguaje soporta computación de valores distribuidos por intermedio de procedimientos de (*un*)*marshall* que permiten la cooperación de programas que envíen y reciban valores a través de canales de comunicaciones no tipados. Esto posibilita la aparición de errores de tipos cuando un programa hace el unmarshall de un valor recibido. El chequeo de un tipo abstracto no es siempre posible en tiempo de hacer el unmarshal. Igualmente, los invariantes característicos de un tipo abstracto deben necesariamente ser satisfechas durante toda la infraestructura distribuida de ejecución para ser correctas. Cuando programas cooperantes son confiables y *well-behaved*, el receptor de los valores marshalled puede omitir el chequeo del valor recibido. En este trabajo se extenderá el lenguaje Acute con mecanismos que permitan a un programa verificar la corrección de un valor marshalled enviado por un programa, ya sea confiable o no, utilizando para ello una infraestructura de PCR.

## 5. REFERENCIAS


[1] F. Besson, G. Dufay, and T. Jensen. *A Formal Model of Access Control for Mobile Interactive Devices*. In *11th European Symposium On Research In Computer Security (ESORICS'06)*, volume 4189 of *Lecture Notes in Computer Science*, 2006. Springer.

[2] M. Blum, S. Kannan, *Software Reliability via Run-Time Result-Checking*, Journal of the ACM, vol. 44, 1997.

[3] T. Coquand and G. Huet. *The Calculus of Constructions*. In: Information and Computation, vol. 76 (Academic Press, 1988), pp. 95–120.

[4] M. Debbabi, M. Saleh, C. Talhi, S. Zhioua. *Embedded Java Security: Security for Mobile Devices*, Springer 2006.

[5] JSR 30 & JSR 139. *Connected Limited Device Configuration Specification*. Version 1.1, Sun Microsystems Inc., 2000.

[6] JSR 37 Expert Group. *Mobile Information Device Profile for Java 2 Micro Edition*. Version 1.0. Sun Microsystems, Inc., 2000.

[7] JSR 118 Expert Group. *Mobile Information Device Profile for Java 2 Micro Edition*. Version 2.0. Sun Microsystems, Inc. and Motorola, Inc, 2002.

[8] C. Luna. *Enseñando Métodos Formales con COQ*, publicado en la Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología: TE&ET, vol.1, páginas 55-64, Diciembre de 2006. Versión online disponible en http://teyet-revista.info.unlp.edu.ar.

[9] G. Necula, P. Lee; *Research in Proof Carrying Code for Untrusted-Code Security*. En Proceedings of the 1997 IEEE Symposium on Security and Privacy, Oakland, 1997.

[10] G. Necula, R. Schneck; *A Sound Framework for Untrusted Verification-Condition Generators*. En "Proceedings of IEEE Symposium on Logic in Computer Science, LICS03", July 2003.

[11] C. Paulin-Mohring. *Inductive Definitions in the system Coq - Rules and Properties*. In *First Int. Conf. on Typed Lambda Calculi and Application*s, edited by M. Bezem and J. F. Groote, LNCS, vol. 664 (Springer-Verlag, 1993), pp. 328–345.

[12] Peter Sewell, James J. Leifer, Keith Wansbrough, Francesco Zappa Nardelli, Mair Allen-Williams, Pierre Habouzit, and Viktor Vafeiadis. *Acute – highlevel programming language design for distributed computation: Design rationale and language definition*. Technical report, University of Cambridge and INRIA Rocquencourt, October 2004.

[13] The Coq Development Team. *The Coq Proof Assistant* Reference Manual, Version V8.0 (2004).

[14] B. Werner. *Une Théorie des Constructions Inductives*. Phd Thesis, Université Paris 7, France (1994).

[15] S. Zanella Béguelin, G. Betarte, C. Luna. *A Formal Specification of the MIDP 2.0 Security Model*. In *Proc. 4th International Workshop on Formal Aspects in Security and Trust, FAST 2006, Hamilton, Canada, August 26-27 2006*, Lectures Notes in Computer Science, vol. 4691, Springer, 2007.