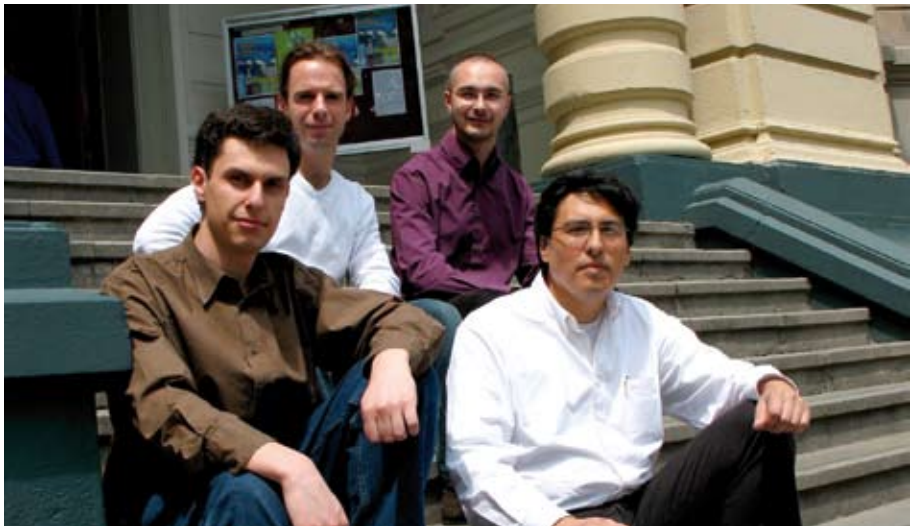
Software + Lenguajes

Alex Bergel:

CONSTRUIR CON CALIDAD

Departamento de Ciencias de la Computación, Universidad de Chile.



Alex Bergel, Romain Robbes, Felipe Bañados y Patricio Plaza.

Soy miembro del grupo PLEIAD del Departamento de Ciencias de la Computación de la Universidad de Chile.

Mi investigación se enfoca en ingeniería de software, particularmente en calidad de software. Las herramientas tradicionalmente utilizadas como lenguajes de programación y ambientes de desarrollo son tremendamente poderosas para construir software, pero son claramente insuficientes para todas las actividades de mantención. Es bien conocido que las empresas gastan un 75% de sus recursos en mantener software, en vez de producir otros nuevos. Mi actividad de investigación ofrece nuevas técnicas y metodologías para facilitar la mantención y el control de calidad de software.

Mis hipótesis de trabajo se basan en la utilización de herramientas de visualización y de metamodelización para ayudar a los desarrolladores a identificar deficiencias y anomalías en su propio código. Mis últimos resultados son Mondrian y Spy. Mondrian es una herramienta ágil para crear "mapas de software". Spy es un framework para construir perfiles de ejecución de código. Mondrian y Spy son parte de la plataforma de análisis de software Moose. Mondrian es un elemento central de Moose sobre el cual se utilizan la mayoría de las herramientas construidas con Moose.

Moose es co-desarrollado con INRIA Lille Nord-Europe (France), Universidad de Berna (Suiza), Universidad de Lugano (Suiza), Vrije Universiteit Brussel (Bélgica) y Universidad de Chile. En cada uno de estos sitios, ingenieros, doctorados e investigadores participan en un esfuerzo colectivo con una gran interacción (por ejemplo, co-escritura de artículos, feedback sobre herramientas, pair-programming, co-organización de eventos).

En la Universidad de Chile trabajo esencialmente con los profesores Cecilia Bastarica, Johan Fabry, Sergio Ochoa, Romain Robbes y Eric Tanter, y con los alumnos Felipe Bañados, Julio Hurtado, Christian Palomares y Vanessa Peña. Julio trabaja en la modernización de procesos de software. Su herramienta se llama Avispa y es desarrollada con Moose. Felipe trabaja en la diferenciación de perfiles de ejecución y es el autor de Hip, una extensión de Spy. Christian trabaja en el ambiente Seaside. Vanessa analiza la cobertura de los unit tests.

Publico los resultados de mi trabajo de investigación en las conferencias y revistas más competitivas. ECOOP, OOPSLA y TOOLS, son las conferencias que privilegio para difundir mis resultados académicos, ya que son reconocidas como las más prestigiosas en el área de programación con objetos. Mis revistas preferidas son Transaction on Software Engineering (TSE) y Elsevier Computer Languages, Systems and Structures. Mis herramientas de investigación son regularmente presentadas a SPIN Chile (red de empresas que tienen un enfoque en la calidad de software), European Smalltalk User Group (ESUG) y Smalltalks.

Durante 2010, tuve ocho artículos aceptados en eventos internacionales (cuatro conferencias y cuatro workshops).

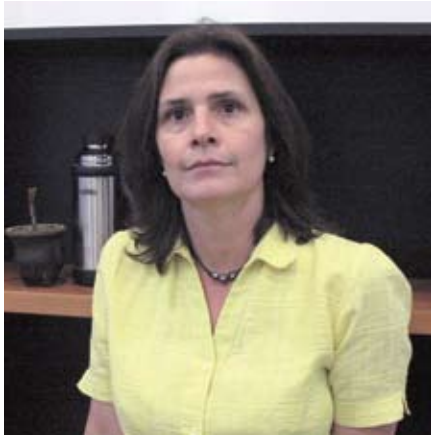
Cecilia Bastarrica:

DESARROLLO DIRIGIDO POR MODELOS: UN NUEVO ENFOQUE EN INGENIERÍA DE SOFTWARE

Departamento de Ciencias de la Computación, Universidad de Chile.

"La ingeniería de software es la aplicación de las Ciencias de la Computación para la resolución de problemas en presencia de recursos limitados."

El Departamento de Ciencias de la Computación (DCC) de la Universidad de Chile, está dentro de la Escuela de Ingeniería, y por lo tanto resulta natural



Cecilia Bastarrica.

reconocer que es importante darle un perfil ingenieril a la especialidad de Ingeniería Civil en Computación. Esto es aún más importante cuando gran parte de los alumnos titulados se desempeñan como ingenieros de software en el ámbito laboral. Sin embargo, el DCC había tenido tradicionalmente desde sus inicios y hasta alrededor de 1998, una orientación más científica que ingenieril.

Desde entonces se ha venido haciendo un esfuerzo sistemático por desarrollar la Ingeniería de Software dentro del Departamento. Es así como se contrataron varios profesores especialistas en ingeniería de software y se le ha dado, como consecuencia, una relevancia mayor a los cursos del área. También esto hace que tengamos el potencial de convertirnos en un polo poderoso en esta área tanto en Chile como en Latinoamérica.

Conceptualmente, la ingeniería de software tiene dos facetas: una más técnica y otra más relativa a la gestión de proyectos de software. Ambas han sido abordadas de manera conjunta en la docencia, pero han tomado rumbos independientes en lo relativo a la investigación y la transferencia tecnológica.

En 1998 se creó el Diploma en Gestión Informática, que luego evolucionó a lo que hoy imparte el DCC como Diploma de Postítulo en Gestión Informática. Fue el primer Postítulo de esa naturaleza en Chile, y aún hoy constituye uno de los programas más prestigiosos en su ámbito. Por su parte, la transferencia de los aspectos técnicos de la ingeniería de software se desarrolla en el Postítulo en Ingeniería y Calidad del Software, creado en el año 2002, y que no tiene competencia en Chile hasta la fecha. Ambos postítulos constituyen la base de los cursos del Magíster en Tecnologías de la Información, que se imparte en el DCC y cuyos alumnos son esencialmente profesionales que buscan actualizar sus conocimientos luego de un tiempo de haberse titulado.

La investigación en Ingeniería de Software dentro del DCC ha tenido un desarrollo más lento para obtener sus mayores logros. Las primeras publicaciones en conferencias internacionales de alguna relevancia fueron en el área de ingeniería Web (LA-WEB, ICWE) o en la revista Journal of Web Engineering, entre los años 2002 y 2004. Sin embargo, esta orientación no ha prosperado, al menos en nuestro Departamento.

Más recientemente mayores logros se han obtenido en el área de diseño de software por parte del grupo MaTE creado en 2007. Claramente, contar con una masa crítica de investigadores le ha dado un gran impulso al área y le ha permitido publicar en las conferencias más relevantes del mundo. Esto se ha visto reflejado en que artículos generados en este grupo hayan sido elegidos dentro de la mejor investigación nacional en computación que se presentó durante las Jornadas Chilenas de Computación 2009, y también entre los mejores de 2010.

El grupo MaTE se ha especializado en desarrollo de software dirigido por modelos, y también ha publicado sus resultados en conferencias de primera línea internacional tales como ASE, WICSA, SPLC, ICSR, ICSP, SHARK e ICMT. Sus resultados también han sido publicados en revistas tales como International Journal of Software Engineering and Knowledge Engineering, Journal of Software and Systems Modeling y Advances in Engineering Software.

El grupo MaTE ha desarrollado un proyecto de investigación pura en colaboración con el INRIA, pero en general su labor ha estado más orientada hacia la investigación aplicada a la industria. Es en este contexto que se ha involucrado en el proyecto Tutelkán durante los últimos cinco años, que ha tenido como objetivo mejorar los estándares de desarrollo de software de la industria chilena. Este proyecto fue financiado por CORFO y se realizó en colaboración con la Universidad Técnica Federico Santa María (UTFSM), la GECHS, la ACTI y SPIN-Chile. Recientemente también se adjudicó un nuevo proyecto Fondef de investigación aplicada y de interés público, conjuntamente con el mismo grupo de la UTFSM que participó de Tutelkán, referido a la formalización y adaptación automática de modelos de proceso de desarrollo de software.

Actualmente el grupo MaTE está formado por las académicas María Cecilia Bastarrica y Nancy Hitschfeld, colaborando regularmente con otros académicos del DCC tales como Sergio Ochoa y Alexandre Bergel. También colaboramos regularmente con los profesores de la UTFSM Marcello Visconti, Hernán Astudillo, Jocelyn Simmonds y Claudio Lobos, estos dos últimos ex alumnos del DCC. Hay cuatro estudiantes de Doctorado: Andrés Vignaga, Daniel Perovich, Pedro Rossel y Julio Ariel Hurtado, y tres estudiantes de Magíster: Alejandro Lagos, Christian Peña y Eduardo Sotomayor.

Contando en la actualidad con un gran grupo humano, muy competente, dedicado a la investigación en ingeniería de software y publicando al mejor nivel internacional, el DCC es sin duda el Departamento de computación más poderoso de Chile, en esta área. También sus múltiples proyectos han contribuido a que esta área sea actualmente la que concentra la mayor parte de los estudiantes de posgrado del DCC, procedentes en general de toda Latinoamérica.

Eric Tanter:

LENGUAJES DE PROGRAMACIÓN: HERRAMIENTAS FUNDAMENTALES PARA EL DESAROLLO DE SOFTWARE

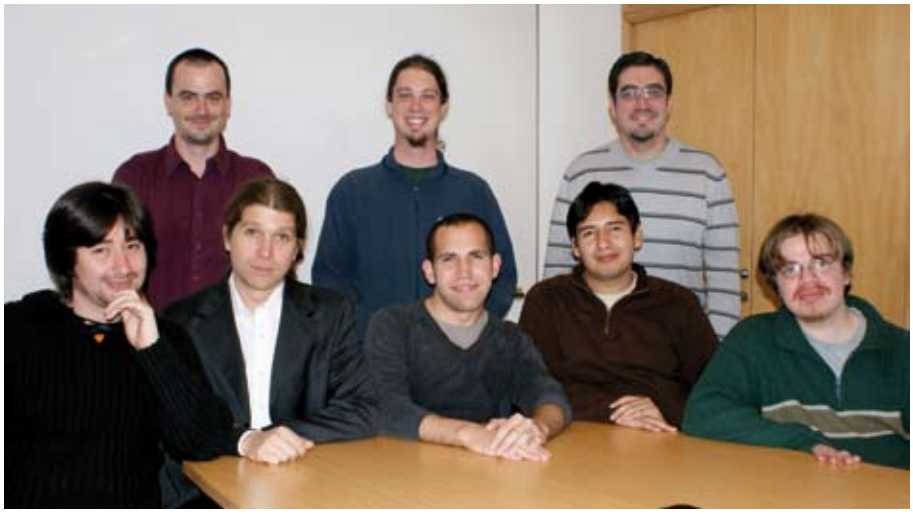
Departamento de Ciencias de la Computación, Universidad de Chile.

Si no fuese porque los podemos programar, los computadores no serían tan fascinantes y versátiles. Pero, ¿cómo programar? En primer lugar, hay que tener un lenguaje para

comunicarse con la máquina. Expresarse en el lenguaje directamente comprendido por la máquina es extremadamente de bajo nivel (a nadie le gusta hablar sólo con 0s y 1s) y no permite aprehender programas de gran tamaño. Por eso, se han creado muchos lenguajes de programación llamados de "alto nivel", y regularmente vemos nuevos lenguajes aparecer. ¿Por qué es así? ¿Por qué no estamos todos contentos y felices con el lenguaje X? (reemplace X por C, Java, C#, Python, Lisp, SQL, Ruby, Smalltalk, Scala, bash, Javascript, o cualquier otro de su gusto)

Cada lenguaje de programación refleja principios y objetivos considerados cruciales para los desarrolladores. Esos principios y objetivos pueden ser antónimos como lo son la eficiencia, la seguridad, y la flexibilidad. Además, cada vez que surge un dominio nuevo para programas (por ejemplo, la Web, los aparatos móviles, los controladores de cohetes), las necesidades son distintas, y los lenguajes existentes, que fueron concebidos con otros fines, se revelan inadecuados.

Mi investigación se centra en explorar distintas dimensiones de los lenguajes de programación para lograr proponer medios adecuados para el desarrollo de cualquier programa, por más complejo que sea. No se trata de buscar un lenguaje perfecto (desgraciadamente no existe), pero sí de buscar mecanismos específicos que pueden ayudar en ciertos casos. Me interesa proponer nuevas abstracciones, estudiar sus propiedades formales y las garantías que proveen, ver cómo se pueden implementar eficientemente, y proponer ambientes de programación que apoyen su uso. Más específicamente, trabajo en varios temas relacionados con la programación por objetos y por aspectos (una nueva abstracción que permite definir módulos que observan y controlan a otros). Me fascina la problemática de reconciliar la flexibilidad y adaptabilidad provistas por lenguajes dinámicos, con la necesidad de imponer barreras de abstracción y encapsulación, así como proveer ciertas garantías de seguridad.



Grupo PLEIAD: Eric Tanter, Johan Fabry, Rodolfo Toledo, Paul Leger, Guillaume Pothier, Ismael Figueroa, Óscar Callau y Esteban Allende.

Soy miembro co-fundador del laboratorio PLEIAD, que explora varias temáticas relacionadas con los lenguajes de programación y sus ambientes de programación. Además de trabajar con otros profesores de PLEIAD (Alex Bergel, Johan Fabry, Romain Robbes), estoy constantemente colaborando con investigadores fuera de Chile. He dirigido y participado en proyectos de investigación con universidades y centros de investigación en Estados Unidos, Canadá, Brasil, Francia, Bélgica, Holanda, Suiza, entre otros.

A la fecha, soy profesor guía de cinco estudiantes de Doctorado en la Universidad de Chile, y co-guía de un doctorando en Bélgica. Estos estudiantes, algunos cerca de titularse y otros recién empezando, trabajan en diversos temas como debugging, aspectos y seguridad, sistemas sensibles a su contexto de ejecución, sistemas de tipos graduales, sistemas distribuidos y concurrentes, y computación pervasiva.

En los últimos cinco años he publicado más de 50 artículos de investigación, incluyendo cerca de 20 artículos en revistas internacionales (IEEE Software, Science of Computer Programming, Software Practice & Experience, etc.). Publico regularmente en conferencias de prestigio en el área como ECOOP, OOPSLA, y AOSD. He participado en más de 30 comités de programa, incluyendo los de las conferencias ECOOP y AOSD. Soy presidente del comité de programa de AOSD 2012, la conferencia de referencia en el desarrollo de software por aspectos.

Los lenguajes de programación son un área esencial de la computación, cubriendo un largo espectro desde lo teórico hasta lo muy aplicado. ¡Todo un programa!

Johan Fabry:

ASPECTOS DE DESARROLLO CON ASPECTOS

Departamento de Ciencias de la Computación, Universidad de Chile.

Un paradigma relativamente nuevo de programación, nacido de la Programación a Objetos, pero no limitado a ellos, es la Programación por Aspectos. La motivación por los aspectos, al igual que en muchas otras evoluciones en la historia de ingeniería de software, es obtener una mayor modularización del software. Donde objetos (y también otros paradigmas "clásicos") fallan es en la modularización de funcionalidad (o preocupaciones) cuya implementación está esparcida en varias partes de la estructura de la aplicación, los llamados "cross-cutting

Johan Fabry (Fotografía: Comunicaciones FCFM).

concerns" (preocupaciones transversales). Aspectos son módulos que resuelven ese problema porque tienen no sólo la implementación de su comportamiento, sino además contienen la especificación de cuándo en la ejecución de los otros módulos del programa, esa funcionalidad debe ejecutarse (lo que se llama un pointcut). Así la funcionalidad esparcida se puede centralizar en un módulo.

Mi investigación se enfoca mayoritariamente en ayudar a los programadores que escriben código con aspectos. Entre otros, escribir código con aspectos requiere especificar los pointcuts, lo que puede resultar *non-trivia*l. Mi trabajo reciente para ayudar en eso es AspectMaps: una visualización de dónde en el programa el aspecto ejecuta su funcionalidad. Eso permite ver, por un lado, si los pointcuts que uno escribe están correctos y, por otro, entender más fácilmente código ajeno (con aspectos). Otra parte de mi investigación es la combinación de lenguajes a dominio específico (DSL) y de los aspectos, resultando en lenguajes de aspecto a dominio específico: DSAL. Ese dominio de investigación busca juntar las ventajas conocidas de los DSL con la Programación por Aspectos. Destaco una ventaja específica aquí: DSLs hacen posible que gente *non experta* en programación pueda escribir programas. DSALs puede hacer posible que gente non *experta* en Programación con Aspectos pueda escribir programas con aspectos. Mi investigación en DSALs aborda infraestructura y metodologías para crear estos lenguajes y también considera posibles interacciones entre varios aspectos escritos en varios lenguajes DSAL.

Junto a esto, también estoy interesado en otros paradigmas de modularizacion que podemos considerar como avanzados (por ejemplo, Traits) y otras herramientas de programación.

Como miembro co-fundador del laboratorio de investigación PLEIAD coopero mayoritariamente con otros profesores del laboratorio: Alex Bergel, Romain Robbes y Eric Tanter.

Con Alex desarrollamos la infraestructura de visualización utilizada por AspectMaps, y tenemos un proyecto SticAmSud con LIFIA (UNLP, Argentina) y RMoD (INRIA Lille Nord-Europe, Francia) donde mi parte del trabajo se centra en este tema.

En este proyecto, además del trabajo junto con los integrantes de RMoD, mi colaboración local es con Romain Robbes, con quien trabajamos en implementar una infraestructura de notificaciones para herramientas, de desarrollo. Tomando inspiración de conceptos de lenguajes de aspectos podemos ofrecer mayor soporte al creador de estas herramientas lo que facilita su creación.

Con Eric Tanter hemos publicado varios trabajos sobre infraestructura para el desarrollo de DSALs (trabajo llamado ReLAx), y mayor expresividad para definir la aplicabilidad de aspectos en sistemas distribuidos (trabajo de distributed scoping strategies). Con Eric Tanter somos parte del Equipe Associee INRIA llamado RAPIDS que trata seguridad en sistemas distribuidos con aspectos, donde mi parte se enfoca en la creación de DSAL para varias facetas de distribución.

Aparte de estas cooperaciones, he trabajado junto a investigadores de la Vrije Universiteit Brussel, Bélgica, en el trabajo de AspectMaps y trabajos relacionados con DSALs.

Tengo dos estudiantes de Doctorado:

Esteban Allende: Su tema encaja con el trabajo de Óscar Callau, estudiante de Doctorado de Eric Tanter. Esteban utilizará los tipos graduales dentro de la máquina virtual Squeak para realizar optimizaciones, logrando una mayor velocidad de ejecución de programas cuando tienen esa información de tipos.

Arturo Zambrano: estudiante de Doctorado del LIFIA, su profesor guía es Silvia Gordillo y yo soy profesor co-guía. El trabajo de Arturo consiste en una evaluación de las metodologías y herramientas de desarrollo con aspectos, en el ciclo completo de desarrollo de una aplicación. Nos enfocamos en la problemática de dependencias e interacciones, tomando un caso de estudio específico con el cual Arturo tiene amplia experiencia como desarrollador en la industria.

Conferencias claves para mi área de investigación son: AOSD (Aspect-Oriented Software Development), ECOOP (European Conference on Object-Oriented Programming), ACM SAC Programming Languages y Programming for Separation of Concerns. El journal ISI de preferencia es Elsevier Science of Computer Programming.

Romain Robbes:

INVESTIGACIÓN EN MINERÍA DE REPOSITORIOS DE SOFTWARE EN PLEIAD

Departamento de Ciencias de la Computación, Universidad de Chile.

Soy miembro del Grupo de Investigación PLEIAD. Mi investigación se centra en el tema de Ingeniería de Software, específicamente en el ámbito de la Minería de Repositorios de Software (Mining Software Repositories o MSR). La investigación en MSR explota la gran cantidad de datos producidos por los desarrolladores, probadores, mantenedores, etc. a fin de validar empíricamente la eficacia de varios enfoques que apoyan a estos profesionales durante el desempeño del trabajo relacionado con la evolución de los sistemas de software.

Romain Robbes.

Pero, ¿por qué los desarrolladores, probadores, mantenedores, y también los managers y arquitectos, necesitan este apoyo?

El mantenimiento y la evolución del software tiene un valor del 75% del costo total de un sistema de software; este porcentaje va en aumento debido a que los sistemas de software se utilizan durante extensos períodos de tiempo. El mantenimiento de software tiene un costo muy elevado por su complejidad. Incluso, realizar cambios simples, como por ejemplo cambiar el nombre de una función por uno más descriptivo, puede resultar un desafío si la función se utiliza miles de veces en una base de código de gran envergadura.

En este contexto, la investigación en MSR parte del supuesto de que la historia de un sistema de software contiene información extremadamente valiosa. Como dice Santayana: *"Aquellos que no pueden recordar el pasado están condenados a repetirlo"*. En el caso de sistemas de software, su historia se puede registrar con gran precisión, en lo que llamamos repositorios de software. Entre algunos repositorios de software encontramos: el sistema de control de versiones (por ejemplo, CVS, Subversion, Git), que contiene todos los cambios en el sistema, sus fechas, y los autores; el sistema de seguimiento de defectos (por ejemplo, Bugzilla, JIRA, Trac), que contiene todos los informes de problemas (bugs), que afectan al sistema, su importancia y su estado; o archivos de listas de distribución de e-mails, que contienen todas las conversaciones sobre el sistema.

A continuación, dos ejemplos específicos de cómo MSR puede ayudar a los desarrolladores y managers en sus actividades diarias:

- **Predicción de los cambios:** Al observar la manera en que el sistema ha cambiado en el pasado, podemos inferir patrones de cambio. Por ejemplo: cuando un método "a()" cambia, hay una probabilidad del 90% de que el método "b()" cambie también. Si un desarrollador rompe el patrón –cambiando sólo "a()"– podemos aconsejarle que verifique si también es necesario cambiar "b()". Si el desarrollador se ha olvidado efectivamente de cambiar "b()", hemos impedido un error potencial.
- **Predicción de errores:** Cuando los recursos son limitados, un director de proyecto no puede permitirse probar todos y cada uno de los componentes del software del mismo modo. Un enfoque de predicción de errores le dirá cuáles son los componentes más propensos a tener defectos, para que así, pueda asignar más probadores para ellos.

Es posible encontrar en esta misma edición de BITS (ver página 2), una descripción más detallada de la investigación en MSR. Mi trabajo en esta área de investigación, en los últimos tres años, ha dado lugar a cuatro publicaciones en revistas (en las revistas Automated Software Engineering; Empirical Software Engineering; Science of Computer Programming; y Software Tools for Technology Transfer), catorce publicaciones en conferencias generales como ICSE y ASE, y conferencias especializadas como MSR, WCRE, ICPC, TOOLS y MoDELS.

Por supuesto, este trabajo no lo he hecho solo, sino que es fruto de colaboraciones con muchos otros investigadores.

En el pasado, he sido miembro del grupo de investigación REVEAL, en la Universidad de Lugano, en Suiza. He trabajado, y trabajo aún, con sus miembros: Prof. Michele Lanza, Dr. Mircea Lungu (ahora en el SCG in Berna), Dr. Marco D'Ambros, Dr. Richard Wettel, Lile Hattori, Fernando Olivero, y Alberto Bacchelli.

Desde mí llegada al DCC de la Universidad de Chile, en enero de 2010, he colaborado activamente con otros miembros del grupo de investigación PLEIAD (profesores Alexandre Bergel, Johan Fabry y Éric Tanter), y otros profesores del DCC (Gonzalo Navarro y Sergio Ochoa). En este momento, no estoy trabajando con ningún estudiante, pero sí, estoy en la búsqueda. Así que si este breve relato de mi trabajo de investigación te resulta interesante, ¡no dudes en ponerte en contacto conmigo! (rrobbes@dcc.uchile.cl).

SIGSE: SPECIAL INTEREST GROUP ON SOFTWARE ENGINEERING

Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile.

La relación entre personas y computadores ha cambiado dramáticamente en los últimos años y esto se debe a que hoy en día los programas de computadora (o software) juegan un rol central en casi todos los aspectos de nuestra vida diaria, en el gobierno, bancos, finanzas, educación, transporte, entretenimiento, medicina, agricultura y leyes, por citar algunos ejemplos de aplicación.

Este crecimiento notable en la dependencia del uso de productos de software se debe a que éstos brindan a las personas, herramientas que les permiten ser más eficaces al resolver sus problemas y les proveen un medio para trabajar y entretenerse que es, a menudo, más seguro, más flexible y menos limitado que otros medios. Sin embargo, la naturaleza del software supone propiedades intrínsecas esenciales -complejidad, invisibilidad, flexibilidad, evolución- que son difíciles de abordar y que sumadas a requisitos actuales como la tendencia a crear productos de escala masiva, con tiempos de respuesta que se miden en segundos, que soportan diferentes esquemas de calidad tales como la seguridad,

Jaime Navón, Rosa Alarcón, Yadran Eterovic y Andrés Neyem.

que pueden encontrarse distribuidos o requieren sofisticados mecanismos de coordinación, sitúan a los productos de software entre los sistemas más complejos hechos por el hombre.

El término *ingeniería de software* se usa hoy ampliamente en los sectores productivos y de servicios, en el gobierno y en las universidades. Ingeniería de software significa la aplicación de un enfoque sistemático, disciplinado y cuantificable de desarrollo, operación y mantenimiento de software. Se trata de algo más que simplemente producir código, incluye calidad, plazos y presupuestos, y el conocimiento y la aplicación de principios y disciplina. Se la puede definir como la ingeniería que aplica sistemáticamente y en forma disciplinada los principios de la Ciencia de la Computación y las matemáticas para lograr soluciones confiables y económicas a problemas de software, así como para la operación y el mantenimiento de estas soluciones. La ingeniería de software también es diferente en carácter a otras disciplinas de la ingeniería, debido tanto a la naturaleza intangible del software como a la naturaleza discreta de la operación del software.

El Departamento de Ciencia de la Computación (DCC) de la Pontificia Universidad Católica de Chile ha estado involucrado en ingeniería de software por más de una década. Un ejemplo de esto es que hemos sido pioneros en incorporar el lenguaje de modelado UML en el programa de estudios de pregrado y también en apoyar su introducción en la industria a nivel nacional realizando asesorías y programas de capacitación en empresas grandes y medianas. Esto último se inserta también en la preocupación del DCC por la transferencia tecnológica. Otro ejemplo de este interés fue la ejecución de un proyecto Fondef que buscaba desarrollar frameworks de aplicación (patrones más componentes de software) para la banca cuando el término framework aún no era conocido por el ambiente local.

En cuanto a la formación de profesionales, a partir de 2009, los alumnos que ingresan a computación pueden elegir la especialidad de Ingeniería de Software. Esta especialidad se enfoca en la formación de ingenieros capaces de diseñar, implementar, validar, operar y mantener sistemas de software como soluciones a problemas reales, satisfaciendo las necesidades de los clientes y usuarios y las restricciones presupuestarias y de tiempo. El programa curricular, de cinco años, proporciona una sólida formación inicial común en ciencias básicas, ciencias de la ingeniería, e ingeniería industrial, y luego se concentra en la formación especializada: Ciencia de la Computación, e ingeniería de software propiamente tal. En esta última área se estudia los conceptos y se ponen en práctica las técnicas y herramientas asociados a la gestión y ejecución de proyectos, la arquitectura de sistemas y el diseño detallado de software, y el testing. El propósito es brindar al profesional conocimiento integral y experiencia práctica para el desarrollo de sistemas en ambientes complejos.

Desarrollando investigación en ingeniería de software

En el año 2008, nace SIGSE (Special Interest Group on Software Engineering), un grupo de investigación en ingeniería de software, actualmente formado por cuatro profesores que colaboran en la supervisión del trabajo de varios alumnos de doctorado, alumnos de magíster y alumnos memoristas. Una de las premisas de SIGSE es generar conocimiento de relevancia nacional e internacional. La investigación que realiza nuestro grupo abarca diversas áreas sobre problemas relacionados con procesos de desarrollo, diseño y arquitectura de software, arquitecturas orientadas a servicios y computación móvil. A modo de ejemplo, detallamos algunas de estas líneas de investigación:

(1) Computación Orientada a Servicio:

La Computación Orientada a Servicios (SOC - Service Oriented Computing) es un paradigma que ha ganado mucha atención en la industria del software debido a que representa una nueva forma de desarrollar arquitecturas de sistemas distribuidos. SOC es una evolución de la ingeniería de software basada en componentes que introduce un nuevo tipo de bloque de construcción llamado servicio, el cual es una funcionalidad que es consumida remotamente utilizando protocolos estándares. A pesar de los importantes beneficios que proporciona este paradigma, aún siguen pendientes temas relacionados con decrecer los costos de creación y mantenimiento de este tipo de aplicaciones. Por ejemplo, los desarrolladores tienen que invertir bastante esfuerzo en descubrir los servicios manualmente, proporcionar el código para invocarlos y realizar las modificaciones necesarias durante la fase de mantenimiento. Los servicios pueden además invocarse entre ellos dando

lugar a servicios compuestos, cuyas partes proveen diferentes niveles de calidad y podrían no estar disponibles en tiempo de ejecución. Una de las líneas de investigación que nuestro grupo realiza es proporcionar mejoras a este paradigma mediante la incorporación de ontologías, metadatos semánticos, y técnicas de razonamiento que permitan la composición diámica de servicios y la garantía de diferentes niveles de calidad. La investigación se enfoca en el diseño y desarrollo de plataformas para crear servicios Web semánticos así como en la incorporación de enfoques ligeros como REST y técnicas de la Web 2.0 tales como Mashups.

(2) Diseño y arquitectura de software Web:

El DCC ha sido pionero también en incorporar las tecnologías de la Web a sus programas de estudio a nivel de pregrado y posgrado. Rápidamente las páginas Web se transformaron en sitios para finalmente aparecer como aplicaciones Web. El desarrollo para la Web tiene características especiales tanto en el proceso como en la arquitectura de la solución. En el proceso mismo, la participación de diseñadores y el diálogo de éstos con los ingenieros ha sido una problemática que ha motivado incluso el surgimiento de nuevas tecnologías. El patrón modelo-vista-controlador ya conocido cobra una nueva relevancia y dimensión, y aparecen los diversos frameworks que permiten facilitar una tarea que al principio era bastante dura. En la actualidad se llevan a cabo trabajos y tesis de alumnos en temas tan diversos como arquitecturas para customización en el lado del cliente que permitan mantener las ventajas de ubicuidad de la aplicación Web; arquitecturas para aplicaciones autoadaptables de acuerdo a frecuencia de navegación de los usuarios; sintetización de un servicio Web en forma automática a partir de un sitio Web, etc.

(3) Procesos de desarrollo:

El desarrollo de software ha sido, desde sus inicios históricos, hace más de 50 años, una tarea difícil. Los proyectos tienden a exceder sus plazos y presupuestos, y a ofrecer menos funcionalidad y de peor calidad que la inicialmente acordada. Para mejorar esta situación, investigadores y profesionales han propuesto modelos de procesos de desarrollo de software basados en las prácticas que aplican habitualmente las organizaciones que son exitosas desarrollando software. Sin embargo, la adopción de un proceso de desarrollo o, incluso, de algunas de sus prácticas, tampoco es una tarea simple. Esta línea de investigación aplicada busca, por una parte, determinar el proceso más adecuado a las características de un proyecto u organización particular y, por otra, formas eficaces de llevar adelante la adopción de prácticas tales como gestión de requisitos, estimaciones de esfuerzo y plazos, gestión de riesgos, modelado visual, y desarrollo iterativo e incremental. Para esto, se debe realizar unas serie de actividades, tales como convencer a algunas personas y entrenar a otras, ayudar a elegir el proyecto adecuado para iniciar la aplicación del nuevo proceso de desarrollo y supervisar la ejecución del mismo por el equipo de desarrollo, identificando fortalezas y debilidades. En resumen, nuestro objetivo es ayudar a las organizaciones de desarrollo de software a ser mejores en lo que hacen.

(4) Computación móvil:

En las últimas tres décadas, nuestra relación con la computación ha estado centrada principalmente en el PC e Internet. Hoy, los avances en computación móvil y comunicaciones inalámbricas, nos están llevando hacia una nueva relación con la tecnología caracterizada principalmente por una fuerte inclusión de la computación en las actividades diarias llevadas a cabo por las personas (por ejemplo en el ocio y entretenimiento, en operaciones financieras y en la educación, entre otros). El desarrollo de software para dispositivos móviles es una tarea que impone al desarrollador lidiar con nuevo desafíos originados por la diversidad de hardware y sensores, sistemas operativos, plataformas de desarrollo y escenarios de aplicación. Esta línea de investigación se orienta a brindar soluciones a los desarrolladores para facilitar la creación de aplicaciones móviles, a través de frameworks de desarrollo que encapsulan estrategias de diseño que abordan problemas recurrentes de un dominio específico de aplicación.

Resumiendo, el DCC en general y SIGSE, en particular, buscan contribuir al mejoramiento del marco conceptual y de la práctica de la ingeniería de software, tanto en Chile como internacionalmente. Para esto, desarrollamos actividades de investigación, de formación de capital humano de pre y posgrado, y de transferencia tecnológica. Como resultado, esperamos transformarnos en el mediano plazo en un referente latinoamericano en el área.

Journals

- IEEE Software
- Computers and Education
- Computing and Informatics
- Expert Systems with Applications
- Journal of Group Decision and Negotiation

Conferencias

- WWW (World Wide Web)
- ECTEL (European Conference on Technology Enhanced Learning)
- IASTED (International Conference on Software Engineering and Applications)
- SEDE (International Conference on Software Engineering and Data Engineering)
- IWSSA (Workshop on System/Software Architectures)
- CSCWD (Computer Supported Cooperative Work in Design)
- CRIWG (Workshop on Groupware: Design, Implementation, and Use)

Más información sobre nuestro grupo puede ser obtenida en sigse.ing.puc.cl