Inteligencia artificial e ingeniería de software: estado y tendencias futuras.

Artículo · Enero de 2004

Fuente: DBLP

2 autores:

Jörg Rech

Persona de libre dedicación

52 PUBLICACIONES 608 CITAS

VER PERFIL

Klaus-Dieter Althoff

Deutsches Forschungszentrum für Künstliche Intelligenz

280 PUBLICACIONES 2.859 CITAS

VER PERFIL

Inteligencia artificial y software Ingeniería: estado y tendencias futuras

Jörg Rech, Klaus-Dieter Althoff

Las disciplinas de Inteligencia Artificial e Ingeniería de Software tienen muchos puntos en común. Ambos se ocupan de modelar objetos del mundo real a partir del mundo real, como procesos de negocio, conocimiento experto o modelos de procesos. Este artículo ofrece una breve descripción general de estas disciplinas y describe algunos temas de investigación actuales en el contexto de puntos de contacto comunes.

1 Introducción

Durante las últimas décadas, las disciplinas de Inteligencia Artificial (IA) e Ingeniería de Software (SE) se han desarrollado por separado sin mucho intercambio de resultados de investigación. En IA investigamos técnicas para los cálculos que permitieron percibir, razonar y actuar. La investigación en SE se preocupaba por ayudar a los seres humanos a desarrollar mejor software más rápidamente.

Hoy en día, varias áreas de investigación de ambas disciplinas se acercan y comienzan a construir nuevas áreas de investigación. Los agentes de software desempeñan un papel importante como objetos de investigación en la IA distribuida (DAI), así como en la ingeniería de software orientada a agentes (AOSE). Los sistemas basados en el conocimiento (KBS) se están investigando para las organizaciones de software de aprendizaje (LSO), así como para la ingeniería del conocimiento.
La inteligencia ambiental (AmI) es una nueva área de investigación para sistemas de software inteligentes, no intrusivos y distribuidos, tanto desde el punto de vista de cómo construir estos sistemas como de cómo diseñar la colaboración entre sistemas ambientales.
Por último, pero no menos importante, la Inteligencia Computacional (CI) juega un papel importante en la investigación sobre análisis de software o gestión de proyectos, así como en el descubrimiento de conocimiento en bases de datos o aprendizaje automático.

Además, en los últimos cinco a diez años varios libros, revistas y conferencias se han centrado en la intersección entre IA y SE. La conferencia internacional y las asociaciones
La revista especializada Automated Software Engineering (ASE) presenta investigaciones sobre enfoques formales y autónomos para respaldar la SE [2]. Temas similares con un mayor enfoque en KBS y la gestión del conocimiento se publican en la conferencia internacional y la revista asociada de Software Engi-neering and Knowledge Engineering (IJSEKE) [1].

En este artículo, ofrecemos una breve descripción general sobre el estado y las tendencias futuras en la intersección entre IA y SE. Nos centramos en los temas agentes de software, KBS, AmI y CI como áreas cubiertas por las contribuciones de este número especial.
En la Sección 2 describimos las disciplinas AI y SE. Los temas enfocados se describen con más detalle en la Sección 3.
Finalmente, en la Sección 4 damos una perspectiva para los próximos años y presentamos nuevos desafíos para ambas disciplinas.

2 Inteligencia artificial e ingeniería de software

Esta sección arrojará algo de luz sobre las disciplinas de IA.
y SE para quienes no estén familiarizados con la otra disciplina.

Aspectos de la Inteligencia Artificial Existe un
acuerdo general en la comunidad de IA en que la disciplina de la IA nació en la conferencia de Dartmouth en 1956. Según Winston [81] “la IA es el estudio de los cálculos que hacen posible percibir, razonar y actuar”.

Wachsmuth [78] asume esta definición y señala que “la IA difiere de la mayor parte de la psicología debido a su mayor énfasis en la computación, y difiere de la mayor parte de las ciencias de la computación debido a su mayor énfasis en la percepción, el razonamiento y la comprensión”. acción". Como campo de estudio académico, muchos investigadores de IA llegan a comprender la inteligencia al ser capaces de producir efectos de la inteligencia: el comportamiento inteligente. Un elemento de la metodología de la IA es que se busca el progreso mediante la construcción de sistemas que realicen: síntesis antes que análisis [78]. “Los sistemas son buena ciencia”, como decía Hendler [34]. O más drásticamente por Wachsmuth [78]: “el objetivo de la IA no es construir máquinas inteligentes habiendo entendido la inteligencia natural, sino comprender la inteligencia natural construyendo máquinas inteligentes”. Aún más sorprendentemente, Aaron Sloman lo expresa de esta manera (citando a su colega Russel Beale): "La IA se puede definir como el intento de conseguir que máquinas reales se comporten como las de las películas". Además, señala que la IA tiene dos vertientes principales, una vertiente científica y una vertiente de ingeniería, que se superponen considerablemente en sus conceptos, métodos y herramientas, aunque sus objetivos son muy diferentes.

Esta visión es apoyada por Wahlster [79], quien aclara que la IA tiene dos tipos diferentes de objetivos, uno motivado por la ciencia cognitiva y el otro por las ciencias de la ingeniería (cf.
Figura 1).

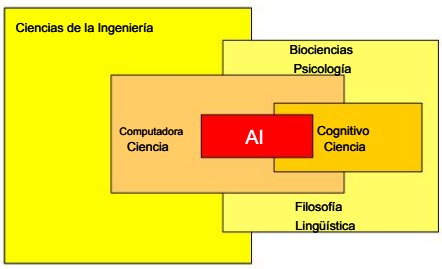

Figura 1 IA y áreas de investigación relacionadas (adaptado de [79])

En la Figura 2 se muestra otra subdivisión (adaptada de Richter [59] y Abecker [4]) de la IA en subcampos, métodos y técnicas.

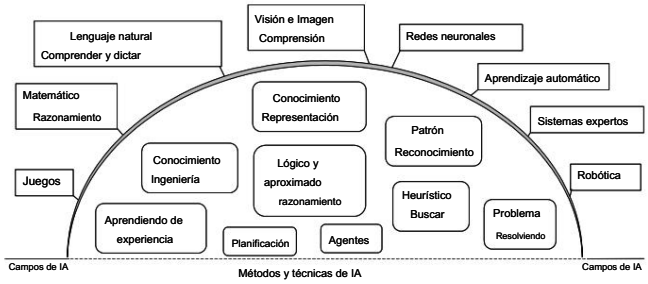

Figura 2 Campos, métodos y técnicas de IA

Para SE, la línea científica orientada hacia las ciencias cognitivas y las humanidades en general podría ser una guía útil para la investigación interdisciplinaria. Por supuesto, existe una fuerte superposición entre la SE y la rama de ingeniería de la IA. Una parte importante de estos últimos son KBS.

Richter [59] define tres niveles diferentes como esenciales para describir KBS: la capa cognitiva (orientada a humanos, racional, informal), la capa de representación (formal, lógica) y la capa de implementación (orientada a máquinas, estructura de datos). turas y programas). Estos niveles se muestran en la Figura 3.
Entre la expresión del conocimiento y su utilización por la máquina deben realizarse varias transformaciones (flechas gruesas). Señalan la dirección de una mayor estructuración dentro de las capas y pasan de la forma cognitiva a una forma más formal y procesada más eficientemente. La letra A es un recordatorio de Adquisición (que está orientada a humanos), mientras que C es una abreviatura de Compilación (orientada a máquinas).

Cada resultado sintáctico en el rango de una transformación entre capas tiene que estar asociado con el significado en el dominio de la transformación. La flecha más interesante y difícil es la transformación inversa de regreso a la capa cognitiva; normalmente se le llama explicación.

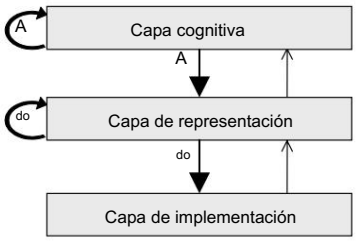

Figura 3 Los tres niveles de los sistemas basados en el conocimiento

¿Por qué la IA es interesante para los investigadores de SE? Puede proporcionar la tecnología inicial y las primeras aplicaciones (exitosas), así como un entorno de prueba para ideas. La inclusión de la investigación apoya la habilitación de procesos realizados por humanos y aumenta la aceptación del usuario. La tecnología de IA puede ayudar a basar el método SE general en una tecnología concreta, proporcionando detalles suficientes para la descripción inicial del método y, a través de la tecnología de referencia disponible, aclarando la semántica del método respectivo. Además, otras técnicas de IA que naturalmente sustituyen o amplían la tecnología elegida se puede utilizar para versiones mejoradas del método SE.

Aspectos de la ingeniería de software

La disciplina de SE nació en 1968 en la conferencia de la OTAN en Garmisch-Partenkirchen, Alemania [52, 71] donde se acuñó el término “crisis de SE”. Su principal preocupación es el desarrollo eficiente y efectivo de sistemas de software de alta calidad y, en su mayoría, de gran tamaño. El objetivo es ayudar a los ingenieros y administradores de software a desarrollar mejor software más rápido con herramientas y métodos (inteligentes).

Desde sus inicios se desarrollaron y maduraron varias direcciones de investigación en este amplio campo. La Figura 4 muestra el modelo de referencia de desarrollo de software que integra fases importantes en el ciclo de vida del software. La Ingeniería de Proyectos se ocupa de la adquisición, definición, gestión, seguimiento y control de proyectos de desarrollo de software, así como de la gestión de los riesgos que surgen durante la ejecución del proyecto. Los métodos de Ingeniería de Requisitos se desarrollan para apoyar la obtención formal e inequívoca de los requisitos de software de los clientes, mejorar la usabilidad de los sistemas y establecer una definición vinculante e inequívoca del sistema resultante durante y después de la definición del proyecto de software. La investigación sobre Diseño y Arquitectura de Software avanza en técnicas para el desarrollo, gestión y análisis de descripciones (formales) de representaciones abstractas del sistema de software, así como herramientas y notaciones requeridas (por ejemplo, UML). Se avanzan técnicas para apoyar la programación profesional de software para desarrollar código fuente altamente mantenible, eficiente y efectivo. Verificación y Validación se ocupa de la planificación, desarrollo y ejecución de pruebas e inspecciones (automatizadas) (formales e informales) para descubrir defectos o estimar la calidad de partes del software.

Investigación para Implementación y Distribución es responsable del desarrollo de métodos para la introducción en el sitio del cliente, soporte durante la operación y la integración en la infraestructura de TI existente.

Después de la entrega al cliente, los sistemas de software suelen pasar a una fase de evolución del software . Aquí el foco de la investigación radica en los métodos para agregar nuevas y perfeccionar funciones existentes del sistema. Asimismo, en la fase paralela se desarrollan técnicas de Mantenimiento de Software para la adaptación a los cambios ambientales, prevención de problemas previsibles y corrección de defectos detectados. Si el entorno cambia drásticamente o es imposible realizar más mejoras, el sistema muere o entra en una fase de reingeniería . Aquí se utilizan técnicas para la comprensión del software y la ingeniería inversa del diseño de software para portar o migrar un sistema a una nueva tecnología (por ejemplo, de Ada a Java o de una arquitectura monolítica a una cliente/servidor) y obtener un sistema mantenible.

Desde los años ochenta se desarrolló la reutilización y gestión sistemática de experiencias, conocimientos, productos y procesos y se denominó Experience Factory (EF) [16].
Este campo, también conocido como Organización de software de aprendizaje. (LSO), investiga métodos y técnicas para la gestión, obtención y adaptación de artefactos reutilizables de proyectos de SE.

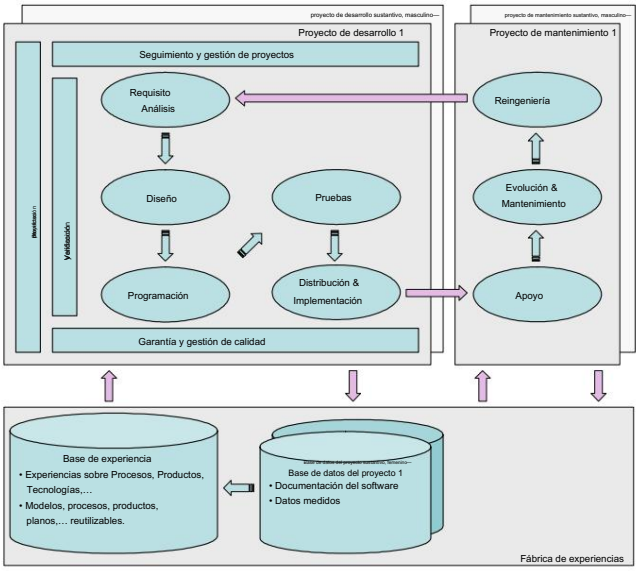

Figura 4 Modelo de referencia de desarrollo de software

¿Por qué es interesante la SE para los investigadores de IA? Apoya el desarrollo sistemático de aplicaciones de IA, el funcionamiento de aplicaciones de IA en entornos de la vida real, así como la evaluación (por ejemplo, [6]), el mantenimiento (por ejemplo, [54]), la mejora continua y la comparación sistemática con aplicaciones alternativas. enfoques (por ejemplo, otro método de modelado). SE también apoya la definición sistemática de los respectivos dominios de aplicación, por ejemplo, a través de métodos de alcance [67].

3 intersecciones entre IA y SE

Si bien las intersecciones entre IA y SE son actualmente raras, se están multiplicando y creciendo. Los primeros puntos de contacto surgieron de la aplicación de técnicas de una disciplina a otra [55].

Hoy en día, los métodos y técnicas de ambas disciplinas apoyan la práctica y la investigación en las respectivas áreas de investigación. La Figura 5 muestra algunas áreas de investigación en IA y SE, así como sus intersecciones.

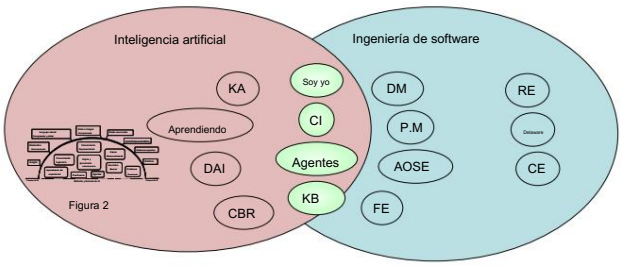

Figura 5 Áreas de investigación en IA y SE y sus intersecciones

El desarrollo sistemático de software (incluida la ingeniería de requisitos (RE), la ingeniería de diseños (DE) o el código fuente (CE)) o los métodos de gestión de proyectos (PM) ayudan a construir sistemas inteligentes mientras se utilizan datos avanzados.

técnicas de análisis. Las técnicas de Adquisición de Conocimiento (KA) [21] ayudan a construir EF y sistemas ambientales inteligentes como las técnicas de Modelado de Dominio (DM) apoyan la construcción de requisitos para sistemas de software y líneas de productos.
El razonamiento basado en casos (CBR) se utiliza para respaldar la recuperación y gestión de datos en EF. Los Agentes de Información se utilizan en SE para simular procesos de desarrollo o para distribuir y explicar solicitudes de cambio.

Ingeniería de software orientada a agentes Los agentes de software suelen ser pequeños sistemas inteligentes que cooperan para alcanzar un objetivo común. Estos agentes son un área relativamente nueva donde se cruzan las investigaciones de KI y SE. Desde el punto de vista de la IA, la atención se centra en sistemas aún más inteligentes y autónomos para resolver problemas más complejos utilizando lenguajes de comunicación entre agentes. En SE los agentes son vistos como sistemas que necesitan métodos formales más o menos especializados para su desarrollo, verificación, validación y mantenimiento.

Ingeniería de software orientada a agentes (AOSE) (también conocida como La ingeniería de software basada en agentes (ABSE)) en relación con la SE orientada a objetos (OOSE) se centra en sistemas donde los objetos en un modelo de un sistema de software son inteligentes, autónomos y proactivos. Actualmente se investiga el desarrollo sistemático y la representación de agentes software y se crearon lenguajes para su representación durante el desarrollo, como el Agent UML [19]. Por ejemplo, varios métodos como MASSIVE de Lind [45], GAIA de Wooldridge et al. [84], MENSAJE de Caire et al. [25], TROPOS de Castro et al. [26], o MAS-CommonKADS de Iglesias et al. [38] fueron desarrollados.

Los agentes y AOSE se aplican en muchas áreas, como interfaces de usuario inteligentes y basadas en agentes para mejorar la usabilidad del sistema, agentes comerciales en comercio electrónico para maximizar las ganancias o ayudar a los agentes en el trabajo diario para automatizar tareas comunes (por ejemplo, reservar habitaciones de hotel) [39]. Además, los agentes de software se utilizan cada vez más para simular dominios del mundo real (por ejemplo, control de tráfico) o procesos de trabajo en SE. Pero la tecnología de agentes no es un paradigma nuevo y brillante sin problemas: Wooldridge describe algunos de los errores de AOSE en [85].

Una encuesta sobre el estado del arte sobre SE orientada a agentes realizada por Tveit resumió publicaciones y métodos anteriores [76].
Se desarrollaron métodos formales para la especificación y verificación de sistemas con el fin de describir, probar y verificar los objetivos, creencias e interacción de agentes y sistemas de agentes [82, 83].

Hoy en día, se organizan cada vez más talleres que cubren los puntos comunes de los agentes inteligentes y la SE.
Ejemplos de comunidades en este campo son el Taller de Ingeniería de Software para Sistemas Multi-Agente a Gran Escala (SELMAS 2004) [46], la Serie de Talleres Internacionales sobre Ingeniería de Software Orientada a Agentes (AOSE 2004) [33], el Taller Internacional sobre Ingeniería de Software Orientada a Agentes (AOSE 2004) [33], el Taller Internacional sobre Ingeniería de Software Orientada a Agentes (AOSE 2004) -Metodologías orientadas (AO 2003), Taller internacional sobre conceptos de agentes radicales (WRAC 2003) [75], o la Conferencia conjunta internacional sobre agentes autónomos y sistemas multiagentes (AAMAS 2004) [64]. Los talleres sobre aspectos más formales de AOSE son FAABS o KIMAS [36, 37].

Como se resume en la hoja de ruta de tecnología de agentes de la red AgentLink, se requiere investigación futura para permitir que los sistemas de agentes se adapten de forma autónoma a la comunicación.

lenguajes, patrones de comportamiento o para apoyar la reutilización de bases de conocimiento [47].

Ingeniería de software basada en el conocimiento

La SE es un campo altamente dinámico en términos de investigación y conocimiento, y depende en gran medida de la experiencia de expertos para el desarrollo y avance de sus métodos, herramientas y técnicas. Por ejemplo, la tendencia a definir y describir "mejores prácticas" o "lecciones aprendidas" es bastante distintiva en la literatura [13]. Como consecuencia, fue en el campo de la ES donde se introdujo una organización, la FE, que era explícitamente responsable de abordar sistemáticamente la experiencia. Una FE es una infraestructura lógica y/o física para el aprendizaje continuo a partir de la experiencia e incluye una base de experiencia (EB) para el almacenamiento y la reutilización del conocimiento. El enfoque EF se inventó a mediados de los años ochenta [15]. Como muestra la práctica, es fundamental para el apoyo del aprendizaje organizacional que la organización del proyecto y la organización que aprende estén separadas [16] (ver también la Figura 4).

El ejemplo inicial de un EF operativo fue el Laboratorio SE de la NASA [62]. Mientras tanto, se desarrollaron aplicaciones de EF en EE. UU. y también en Europa [8, 73]. La gran cantidad de aplicaciones exitosas de EF dio el impulso para estudiar los LSO de manera más intensiva con respecto a la metodología para construir y ejecutar un EF [63]. Esto también incluye la definición de procesos, roles y responsabilidades relacionados y, por último, pero no menos importante, la realización técnica. La metodología más detallada para la construcción de un EF/EB sobre el conocimiento del proyecto, también para la presentación de los procesos correspondientes, se proporciona en [73], una extensión sobre evaluación, mantenimiento y arquitectura se puede encontrar en [54].

EF está emergiendo cada vez más hacia un enfoque genérico para la reutilización del conocimiento y especialmente de la experiencia. Esto incluye también aplicaciones independientes del dominio SE, por ejemplo, el apoyo al proceso de mejora continua en hospitales [9], el campo de la mesa de ayuda y soporte de servicios [72] y la gestión de proyectos "no software" [23]. . Las tendencias futuras en el alcance de los FA incluyen el detalle de todas las políticas necesarias, la validación y la evaluación empírica [17, 73], la adquisición de experiencia con la realización técnica de enormes FA [58], la integración con los procesos de negocio correspondientes. [10], y el funcionamiento de los EF [53].

En las áreas de ciencia cognitiva e inteligencia artificial, la RBC surgió a finales de los años setenta y principios de los ochenta como modelo para la resolución de problemas y el aprendizaje humanos [65, 66]. En IA, esto llevó a que KBS se centrara en la experiencia (conocimiento de casos específicos) a finales de los ochenta y principios de los noventa, principalmente en forma de casos de solución de problemas [14]. Desde hace varios años ha habido una fuerte tendencia en la comunidad CBR [7] a desarrollar métodos para abordar aplicaciones más complejas. Un ejemplo es el uso de CBR para KM [5], otro es su uso para SE [69]. Una cuestión muy importante aquí es la integración de la RBC con la FE: desde mediados de los años noventa, la RBC se utiliza tanto en el nivel de proceso organizacional de la FE como en el nivel técnico de implementación de la BE [12, 35, 74]. Mientras tanto, este enfoque se consolida cada vez más [7, 20, 40]. Por lo general, los enfoques orientados al producto y al proceso se utilizan independientemente uno del otro o como alternativas. Como primer paso, se ha logrado una profunda integración de los enfoques de FE y RBC [7, 11, 54, 73].

En [11, 13, 27, 28, 43, 54, 73] se ofrece una descripción general de los enfoques relevantes para la SE basada en el conocimiento. Los eventos relevantes son parte de LSO, SEKE, ASE y CBR. (www.iccbr.org) (así como las revistas correspondientes, incluida la Revista Internacional sobre Conocimiento y Sistemas de Información (KAIS)). Las conferencias y talleres (también así como las revistas) sobre gestión del conocimiento son de interés si tienen una relación concreta con cuestiones relacionadas con el software. Otros eventos relevantes son la Conferencia Conjunta sobre SE basada en el conocimiento (JCKBSE) 2002 [80] y 2004, y los talleres sobre Mantenimiento orientado al conocimiento (KOM 2004) o SE orientada a objetos basada en el conocimiento (KBOOSE 2002).

Inteligencia computacional y KDD El área de investigación CI ha observado recientemente un creciente interés por parte de investigadores de ambas disciplinas. Técnicas como redes neuronales, algoritmos evolutivos o sistemas difusos se aplican y adaptan cada vez más a problemas específicos de ES. Se utilizan para estimar el progreso de proyectos para respaldar la gestión de proyectos de software [24], para el descubrimiento de módulos defectuosos para garantizar la calidad del software, o para planificar actividades de prueba y verificación de software para minimizar el esfuerzo de control de calidad [41, 44 , 56].

En un estudio de última generación sobre KDD en SE, Mendonca y Sunderhaft resumieron publicaciones anteriores, así como varias técnicas y herramientas de minería [48]. Una de las primeras publicaciones que recopiló explícitamente contribuciones a KDD para datos SE fue el número especial de IJSEKE en 1999 [50]. Cada vez se organizan más talleres sobre CI y SE.

Ejemplos de talleres son WITSE (Tecnologías inteligentes para SE), MSR (Repositorios de software de minería), DMSK (Minería de datos para SE e ingeniería del conocimiento) y SCASE (Computación suave aplicada a SE).

Hoy en día, se han establecido muchas áreas de aplicación de KDD en SE en campos como la gestión de calidad, la gestión de proyectos, la gestión de riesgos, la reutilización de software o el mantenimiento de software. Por ejemplo, Khoshgoftaar et al. Técnicas de clasificación aplicadas y adaptadas a datos de calidad del software [42]. Dick investigó el determinismo de las fallas del software con análisis de series de tiempo y técnicas de agrupamiento [31]. Cook y Wolf utilizaron el enfoque de Markov para extraer modelos de procesos y flujos de trabajo a partir de datos de actividad [29]. Pozewauning examinó el descubrimiento y clasificación del comportamiento de los componentes a partir del código y los datos de prueba para respaldar la reutilización del software [57]. Michail utilizó reglas de asociación para detectar patrones de reutilización (es decir, uso típico de clases de bibliotecas) [49]. Como una aplicación de KDD en el mantenimiento de software, Shirabad desarrolló un instrumento para la extracción de relaciones en sistemas de software mediante métodos inductivos basados en datos de varios repositorios de software (por ejemplo, registros de actualización, sistemas de versiones) para mejorar el análisis de impacto en las actividades de mantenimiento. 70]. Zimmermann y sus colegas persiguen el mismo objetivo utilizando una técnica similar a la CBR. Para ayudar a los mantenedores de software con experiencias relacionadas en forma de reglas de asociación sobre cambios en los sistemas de software, extrajeron reglas de asociación de un sistema de versiones recopilando transacciones que incluían un cambio específico [86].

Morasca y Ruhe construyeron un enfoque híbrido para la predicción de módulos defectuosos en sistemas de software con conjuntos aproximados y regresión logística basado en varias métricas (por ejemplo, LOC) [51].

Se requieren investigaciones futuras en este campo para analizar planes formales de proyectos para el descubrimiento de riesgos, adquirir información del proyecto para la gestión de proyectos o extraer directamente representaciones de software (por ejemplo, UML, código fuente) para detectar defectos y fallas en las primeras etapas del desarrollo.

Inteligencia ambiental La idea
detrás de AmI son sistemas sensibles, adaptativos y reactivos que están informados sobre las necesidades, hábitos y emociones del usuario para ayudarlo en su trabajo diario [30]. Por lo tanto, se necesitan técnicas para sistemas autónomos, inteligentes, robustos y de autoaprendizaje que permitan la comunicación entre sistemas (interfaces máquina-máquina y ontologías) o usuarios y sistemas (interfaces hombre-máquina).

AmI se basa en varias áreas de investigación, como la computación ubicua y omnipresente, los sistemas inteligentes y la conciencia del contexto [68]. La investigación en AmI intenta construir un entorno similar a las áreas de investigación mencionadas anteriormente, como agentes de software inteligentes, KBS, así como el descubrimiento de conocimientos (por ejemplo, para detectar y analizar sistemas y software extraños).

Existen varias áreas de investigación de IA para el desarrollo de algoritmos inteligentes para aplicaciones AmI [77], por ejemplo, elaboración de perfiles de usuario, conciencia del contexto, comprensión de la escena [3] o tareas de planificación y negociación [30]. La investigación desde el punto de vista SE se ocupa del desarrollo impulsado por modelos para la informática móvil [30], la verificación del código móvil [61], la especificación de sistemas adaptativos [60] o el diseño de sistemas integrados [18]. Además, necesitamos interfaces humanas inteligentes desde una perspectiva de usabilidad que se traduzcan entre los usuarios y una nueva configuración del sistema ambiental. Se podría establecer una fusión de estos dos campos para analizar y evaluar sistemas software ajenos que intentan conectarse con el propio sistema para ser ejecutados en su hardware.

Actualmente, la investigación sobre AmI está financiada principalmente por la UE y por la Fundación Nacional Alemana para la Ciencia (DFG). Por ejemplo, en [32] se publicaron varios escenarios que describen la visión de la UE, y el proyecto integrado WearIT@Work está financiado en la Universidad de Bremen y hace hincapié en la AmI para los procesos de trabajo. En Alemania, la DFG financió el "Forschungsschwerpunkt Ambient Intelligence" en la Universidad de Kaiserslautern (http://www.eit.uni-kl.de/AmI).

Se organizaron varios talleres y conferencias sobre la AmI para fomentar el intercambio sobre la AmI. Ejemplos de estas reuniones son el Simposio Europeo sobre Inteligencia Ambiental (EUSAI) [3], el Taller sobre Inteligencia Ambiental (WAI), el Taller sobre Inteligencia Ambiental para el Descubrimiento Científico (AMDI), el Taller sobre Inteligencia Ambiental en el Trabajo, la Conferencia Internacional. sobre Computación Ubicua (Ubi-Comp) [22], o el Taller sobre Agentes para la Computación Ubicua (UbiAgents).

4 perspectiva

De esta visión general se desprende claramente que existen fuertes vínculos entre la inteligencia artificial y la ingeniería de software que ofrecen un gran potencial para futuras investigaciones. Se desarrollarán muchas nuevas aplicaciones y campos de investigación de interés para ambas disciplinas que cubrirán sistemas basados en el conocimiento. desarrollar sistemas que cubran el conocimiento para organizaciones de software de aprendizaje, el desarrollo de inteligencia computacional y técnicas de descubrimiento de conocimiento para artefactos de software, SE orientada a agentes o el desarrollo profesional de sistemas de inteligencia ambiental.

Como conclusión, hemos identificado varios campos de investigación interesantes para ambas disciplinas. Los artículos del resto de este número especial profundizan en problemas de investigación específicos en estas intersecciones.

Referencias

[1] SEKE 2002: la 14ª Conferencia Internacional sobre Ingeniería del Software e Ingeniería del Conocimiento. Is-chia, Italia: Nueva York: ACM, 2002.

[2] Actas de la 18ª Conferencia Internacional IEEE sobre Ingeniería de Software Automatizada (ASE). Montreal, Quebec, Canadá: IEEE Computer Society, ISBN: 0-769-52035-9, 2003.

[3] Aarts EHL, Inteligencia ambiental: primer estudio europeo simposio (EUSAI 2003), vol. 2875. Veldhoven, Países Bajos: Springer-Verlag, 3540204180, 2003.

[4] Abecker A., "Wissensbasierte Systeme", Berufsaka-demie Mosbach, Lecture Notes 2002.

[5] Aha DW, Becerra-Fernandez I., Maurer F. y Mu-ñoz-Avila H., Explorando sinergias de conocimiento: gestión y razonamiento basado en casos. Menlo Park, California: AAAI Press, ISBN: 1-577-35094-4, 1999.

[6] Althoff K.-D., Evaluación de sistemas de razonamiento basados en casos: el estudio de caso de Inreca. Tesis postdoctoral (Habilitationsschrift). Departamento de Ciencias de la Computación, Universidad de Kaiserslautern, Kaiserslautern, Alemania, 1997.

[7] Althoff K.-D., "Razonamiento basado en casos", en Manual de ingeniería de software e ingeniería del conocimiento, vol. 1, S.-K. Cambió. Singapur: World Scientific, 2001, págs. 549-587.

[8] Althoff KD, Becker KU, Decker B., Klotz A., Leo-pold E., Rech J. y Voss A., "El proyecto indiGo: mejora de la gestión de experiencias y el aprendizaje de procesos con discursos moderados". Berlín, Alemania: Springer Verlag, 2002, págs. 53-79.

[9] Althoff K.-D., Bomarius F., Müller W. y Nick M., "Utilización de un razonamiento basado en casos para respaldar procesos de mejora continua", presentado en el Taller alemán sobre aprendizaje automático, Leipzig, 1999.

[10] Althoff K.-D., Decker B., Hartkopf S., Jedlitschka A., Nick M. y Rech J., "Experience Management: The Fraunhofer IESE Experience Factory", presentado en la Conferencia Industrial Data Mining, Leipzig, 2001.

[11] Althoff K.-D. y Nick MM, "Cómo apoyar a ex-Gestión de la experiencia con evaluación: fundamentos, métodos de evaluación y ejemplos para el razonamiento basado en casos y la fábrica de experiencias". Berlín: Springer Verlag, 2004 (de próxima publicación, aceptado para la serie LNAI).

[12] Althoff K.-D. y Wilke W., "Usos potenciales de case-razonamiento basado en la experiencia en la construcción de sistemas de software", presentado en el 5º Congreso Alemán

comprar en Case-Based Reasoning (GWCBR'97), Universidad de Kaiserslautern, 1997.

[13] Aurum A., Gestión del conocimiento de la ingeniería de software. Berlín: Springer, ISBN: 3540003703, 2003.

[14] Bartsch-Spörl B., "Ansätze zur Behandlung von fallori-entiertem Erfahrungswissen in Expertensystemen", Künstliche Intelligenz, págs. 32-36, 1987.

[15] Basili VR, Evaluación cuantitativa de la metodología del software. College Park, Maryland: Universidad de Maryland, 1985.

[16] Basili VR, Caldiera G. y Rombach HD, "Experi-ence Factory", en Encyclopedia of Software Engineer-ing, vol. 1, JJ Marciniak, Ed. Nueva York: John Wiley & Sons, 1994, págs. 469-476.

[17] Basili VR, Shull F. y Lanubile F., "Construcción de conocimiento a través de familias de experimentos", IEEE Transac-tions on Software Engineering, vol. 25, págs. 456-73, 1999.

[18] Basten T., Geilen M. y Groot H. d., "Inteligencia ambiental: impacto en el diseño de sistemas integrados", T. Bas-ten, M. Geilen y H. d. Groot, Eds.: Kluwer Aca-demic Publishers, 2003.

[19] Bauer B., Müller JP y Odell J., "Agente UML", Revista Internacional de Ingeniería de Software e Ingeniería del Conocimiento, vol. 11, págs. 207-230, 2001.

[20] Bergmann R., Althoff K.-D., Breen S., Göker M., Manago M., Traphöner R. y Wess S., Desarrollo de aplicaciones de razonamiento industrial basadas en casos: la metodología IN-RECA, vol. 1612, 2ª ed. Berlín: Nueva York, ISBN: 3-540-20737-6, 2003.

[21] Birk A., Surmann D. y Althoff KD, "Aplicaciones de adquisición de conocimientos en ingeniería de software experimental", presentado en el 11º Taller Europeo de Adquisición de Conocimientos (EKAW): Adquisición, modelado y gestión de conocimientos, Berlín, Alemania, 1999.

[22] Borriello G., UbiComp: 4ª Conferencia Internacional sobre Computación Ubicua, vol. 2498. Berlín: ACM, 2002.

[23] Brandt M., Ehrenberg D., Althoff K.-D. y Nick MM, "Ein fallbasierter Ansatz für die computergestützte Nutzung von Erfahrungswissen bei der Projektarbeit", presentado en la 5ª Internationale Tagung Wirtschaftsin-formatik (WI'01) Economía de la era de la información, Heidelberg, 2001.

[24] Brandt M. y Nick M., "Reutilización respaldada por computadora de la experiencia de gestión de proyectos con una base de experiencia", presentado en el Tercer Taller Internacional sobre Avances en las Organizaciones de Software de Aprendizaje (LSO 2001), Berlín, Alemania, 2001.

[25] Caire G., Coulier W., Garijo F., Gómez J., Pavón J., Leal F., Chainho P., Kearney P., Stark J., Evans R. y Massonet P., "Agent análisis orientado utilizando MESSAGE/UML", presentado en el Segundo Taller Internacional sobre Ingeniería de Software Orientada a Agentes (AOSE 2001), 2002.

[26] Castro J., Kolp M. y Mylopoulos J., "Hacia la ingeniería de sistemas de información basada en requisitos: el proyecto Tropos", Information Systems, vol. 27, págs. 367-91, 2002.

[27] Chang SK, "Manual de ingeniería de software y ingeniería del conocimiento. vol. 1, Fundamentos", World Scientific, 2001.

[28] Chang SK, "Manual de ingeniería de software e ingeniería del conocimiento. Vol. 2, Tecnologías emergentes", World Scientific, 2002.

[29] Cook JE y Wolf AL, "Descubriendo modelos de procesos de software a partir de datos basados en eventos", ACM Transactions on Software Engineering and Methodol-ogy, vol. 7, págs. 215-49, 1998.

[30] Da Costa O. y Punie Y., Hoja de ruta científica y tecnológica: inteligencia ambiental en la vida cotidiana (Ami@Life): documento de trabajo no publicado del IPTS, 2003.

[31] Dick SH, Inteligencia computacional en software Seguro de calidad. Tesis Doctoral. Departamento de Ingeniería y Ciencias de la Computación, Facultad de Ingeniería, Universidad del Sur de Florida, 2002.

[32] Ducatel K., Escenarios de inteligencia ambiental en 2010. Luxemburgo: Oficina de Publicaciones Oficiales de las Comunidades Europeas, ISBN: 9289407352, 2001.

[33] Giorgini P., Müller JP y Odell JJ, Actas del cuarto taller internacional sobre ingeniería de software orientada a agentes IV (AOSE 2003), vol. 2935. Melbourne, Australia: Springer, 3540208267, 2004.

[34] Hendler J., "Sistemas de IA experimentales", Revista de IA experimental y teórica, vol. 7, págs. 1-5, 1995.

[35] Henninger S., "Desarrollo del conocimiento del dominio mediante la reutilización de experiencias de proyectos", SIGSOFT Software Engineering Notes, vol. Agosto de 1995, págs. 186-195, 1995.

[36] Hexmoor H., Actas de la Conferencia Internacional sobre Integración de Sistemas Multiagentes Intensivos en Conocimiento (KIMAS 2003). Cambridge, MA, EE.UU.: IEEE, ISBN: 0-780-37958-6, 2003.

[37] Hinchey MG, Enfoques formales para sistemas basados en agentes: Segundo taller internacional (FAABS 2002), vol. 2699. Greenbelt, MD, EE.UU.: Springer, ISBN 3-540-40665-4, 2002.

[38] Iglesias CA, Garijo M., González JC y Velasco JR, "Análisis y diseño de sistemas multiagente utilizando MAS-CommonKADS", presentado en Teorías, arquitecturas y lenguajes de agentes. 4to Taller Internacional, ATAL'97, Berlín, Alemania, 1998.

[39] Jennings NR, Sycara K. y Wooldridge M., "A hoja de ruta de investigación y desarrollo de agentes", Vivek, vol. 12, págs. 38-66, 1999.

[40] Kalfoglou Y., Menzies T., Althoff KD y Motta E., "Metaconocimiento en el diseño de sistemas: ¿panacea... o promesa no cumplida?" Knowledge Engineering Review, vol. 15, págs. 381-404, 2000.

[41] Khoshgoftaar TM, Ingeniería de software con inteligencia computacional, vol. 731. Boston: Kluwer Aca-demic Publishers, ISBN: 1-402-07427-1, 2003.

[42] Khoshgoftaar TM, Allen EB, Jones WD y Hudepohl JP, "Extracción de datos de bases de datos de desarrollo de software", Software Quality Journal, vol. 9, págs. 161-76, 2001.

[43] Last M., Kandel A. y Bunke H., Métodos de inteligencia artificial en pruebas de software, 2003.

[44] Lee J., Ingeniería de software con inteligencia computacional. Berlín: Nueva York, ISBN: 3540004726 (papel alcalino) LCCN: 2003-42588, 2003.

[45] Lind J., Ingeniería de software iterativa para sistemas multiagente: el método MASSIVE, vol. 1994. Berlín: Springer, 3540421661, 2001.

[46] Lucena CJP d., García AF, Romanovsky AB, Castro J. y Alencar PSC, Actas del segundo taller internacional sobre ingeniería de software para sistemas multiagente a gran escala (SELMAS 2003), vol. 2940. Berlín: Springer, ISBN 3-540-21182-9, 2003.

[47] Luck M., McBurney P. y Preist C., "Agent Technology: Enabling Next Generation Computing", Informe AgentLink No. 0-854-32788-6, 2003.

[48] Mendonca M. y Sunderhaft NL, "Mining Software Engineering Data: A Survey", DACS: Centro de análisis y datos para software, Informe técnico 1999.

[49] Michail A., "Patrones de reutilización de bibliotecas de minería de datos utilizando reglas de asociación generalizadas", presentado en las Actas de la Conferencia Internacional sobre Ingeniería de Software de 2000 (ICSE 2000), Nueva York, 2000.

[50] Morasca S. y Ruhe G., "Número especial sobre: Conocimiento edge Discovery from Empirical Software Engineering Data", Revista internacional de ingeniería de software e ingeniería del conocimiento, vol. 9, págs. 495-498, 1999.

[51] Morasca S. y Ruhe G., "Un enfoque híbrido para analizar datos empíricos de ingeniería de software y su aplicación para predecir la propensión a fallas del módulo en el mantenimiento", Journal of Systems and Software, vol. 53, págs. 225-237, 2000.

[52] Naur P. y Randell B., "Software Engineering: Report of a Conference", patrocinado por el Comité Científico de la OTAN, Garmisch, Alemania, 7-11 de octubre de 1968.

[53] Nick M., Althoff KD y Tautz C., "Mantenimiento sistemático de repositorios de experiencias corporativas", Computational Intelligence, vol. 17, págs. 364-86, 2001.

[54] Nick MM, Creación y ejecución de sistemas de información basados en experiencias de larga duración. Tesis Doctoral. Departamento de Ciencias de la Computación, Universidad de Kaiserslautern, Kaiserslautern, que se presentará en 2004.

[55] Partridge D., Inteligencia artificial e ingeniería de software. Neering: comprender la promesa del futuro. Chicago: Pub Glenlake. Co. Fitzroy Dearborn Publishers, ISBN 1-57958-062-9, 2000.

[56] Pedrycz W. y Peters JF, Inteligencia computacional gencia en ingeniería de software. Singapur: River Edge Nueva Jersey, ISBN: 9-810-23503-8, 1998.

[57] Pozewaunig H., Comportamiento de los componentes mineros para respaldar la recuperación de software. Tesis Doctoral. Institut für Infor-matik-Systeme der Fakultät für Wirtschaftswissen-schaften und Informatik, Universität Klagenfurt, Kla-genfurt, 2001.

[58] Rech J., Decker B. y Althoff K.-D., "Using Knowl-edge Discovery Technology in Experience Management Systems", presentado en el taller "Maschinelles Lernen (FGML01)", Universität Dortmund, 2001.

[59] Richter MM, "Artificial Intelligence", Universidad de Calgary, Canadá, Lecture Notes 2004.

[60] Roman G.-C., Número especial sobre ingeniería de software. para la movilidad. Dordrecht, Países Bajos: Kluwer Academic, Serie ISSN: 0928-8910, 2002.

[61] Roman GC, Picco GP y Murphy AL, "Software Engineering for Mobility: A Roadmap", presentado en Future of Software Engineering Track de la 22ª ICSE, Limerick, Irlanda, 2000.

[62] Rombach HD y Ulery BT, "Establecimiento de un programa de mejora del mantenimiento basado en mediciones: lecciones aprendidas en el SEL", Universidad de Maryland, College Park, Maryland 1989.

[63] Ruhe G. y Bomarius F., "Actas de las organizaciones de software de aprendizaje (LSO): metodología y aplicaciones", presentado en la 11ª Conferencia Internacional sobre Ingeniería de Software e Ingeniería del Conocimiento, SEKE'99, Kaiserslautern, Alemania, 1999.

[64] Sandholm T. y Wooldridge MJ, Segunda Conferencia Internacional Conjunta sobre Agentes Autónomos y Sistemas Multiagentes (AAMAS 2003). Melbourne, Australia: ACM, ISBN: 1-581-13685-4, 2003.

[65] Schank RC, Memoria dinámica: una teoría del recuerdo y el aprendizaje en computadoras y personas. Cambridge Cambridgeshire: Nueva York Cambridge University Press, ISBN: 0521248582, 1982.

[66] Schank RC, Abelson RP y conjunto a., Scripts, Planes, metas y comprensión: una investigación sobre las estructuras del conocimiento humano. Hillsdale, NJL Erlbaum Associates: Nueva York, ISBN: 0470990333, 1977.

[67] Schmid K., "Systematische Wiederverwendung im Produktlinienumfeld - ein Entscheidungsproblem", Kunstliche Intelligenz, vol. 3, 2004.

[68] Shadbolt N., "Del editor en jefe - Ambient Intelligence", Sistemas inteligentes IEEE y sus aplicaciones, vol. 18, págs. 2 (2 páginas), 2003.

[69] Shepperd M., "Razonamiento basado en casos e ingeniería de software", en Gestión del conocimiento de ingeniería de software, A. Aurum, Ed. Berlín: Springer, 2003.

[70] Shirabad JS, Mantenimiento de software de soporte mediante registros de actualización de software de minería. Tesis Doctoral. Escuela de Ingeniería y Tecnología de la Información, Universidad de Ottawa, Ottawa, Ontario, Canadá, 2003.

[71] Simons CL, Parmee IC y Coward PD, "35 años después: ¿en qué medida el diseño de ingeniería de software ha logrado sus objetivos? IEE Proceedings Software, vol. 150, págs. 337-50, 2003.

[72] Stolpmann M. y Wess S., Optimierung der Kunden-beziehungen mit CBR-Systemen - Sistemas inteligentes para comercio electrónico y soporte. Bonn: Addison Wesley Longmann, 1998.

[73] Tautz C., Personalización de sistemas de gestión de experiencias de ingeniería de software y procesos relacionados para compartir experiencias de ingeniería de software. Tesis doctoral. Departamento de Ciencias de la Computación, Universidad de Kaiserslautern, Alemania, Kaiserslautern, 2000.

[74] Tautz C. y Althoff KD, "Uso de razonamiento basado en casos para reutilizar el conocimiento del software", Investigación y desarrollo de razonamiento basado en casos, págs. Plaza, E. - Berlín, Alemania, Alemania Springer-Verlag, 1997, xiii+648 156-65, 18 referencias, 1997.

[75] Truszkowski W., Rouff C. y Hinchey M., Actas del primer taller internacional sobre conceptos de agentes radicales (WRAC 2002), vol. 2564. McLean, VA, EE.UU.: Springer, 3-540-40725-1, 2003.

[76] Tveit A., "Una encuesta sobre ingeniería de software orientada a agentes". neering", presentado en el Proc. de la Primera Conferencia NTNU CSGS, 2001.

[77] Verhaegh WFJ, Aarts EHL y Korst J., Algoritmos en inteligencia ambiental, vol. 2. Boston: Dordrecht, ISBN: 140201757X, 2004.

[78] Wachsmuth I., "El concepto de inteligencia en IA", en Inteligencia preracional: comportamiento adaptativo y sistemas inteligentes sin símbolos ni lógica, vol. 1, h.
Cruse, DJ y Ritter H., Eds. Dordrecht, Países Bajos: Kluwer Academic Publishers, 2000, págs. 43-55.

[79] Wahlster W., "Einführung in die Methoden der Künstli-chen Intelligenz", Universidad de Saarbrücken, Alemania, Lecture Notes 2002.

[80] Welzer T., Yamamoto S. y Rozman I., Actas de la quinta conferencia conjunta sobre ingeniería de software basada en el conocimiento. Ámsterdam: Washington DC, ISBN: 1586032747, 2002.

[81] Winston PH, Inteligencia artificial, 3.ª (repr. con correcciones 1993) ed. Reading, Massachusetts: Addison-Wesley, ISBN: 0-201-53377-4, 1993.

[82] Wooldridge M., "Ingeniería de software basada en agentes", Ingeniería de software de actas de la IEE, vol. 144, págs. 26-37, 1997.

[83] Wooldridge M. y Ciancarini P., "Agente orientado ingeniería de software: el estado del arte", presentado en el 1er Taller Internacional sobre Ingeniería de Software Orientada a Agentes (AOSE 2000), Berlín, Alemania, 2000.

[84] Wooldridge M., Jennings NR y Kinny D., "The Metodología Gaia para el análisis y diseño orientado a agentes", Agentes autónomos y sistemas multiagente, vol. 3, págs. 285-312, 2000.

[85] Wooldridge MJ y Jennings NR, "Ingeniería de software con agentes: trampas y errores", IEEE Inter-net Computing, vol. 3, págs. 20-7, 1999.

[86] Zimmermann T., Weißgerber P., Diehl S. y Zeller A., "Mining Version Histories to Guide Software Changes", presentado en la 26ª Conferencia Internacional sobre Ingeniería de Software (ICSE), Edimburgo, Reino Unido, 2004.

Contacto

Dr. habil. Klaus-Dieter Althoff Gestión
de datos y conocimientos Universidad de
Hildesheim PO Box 101363
31113 Hildesheim,
Alemania althoff@dwm.uni-
hildesheim.de

Jörg Rech
Fraunhofer IESE
Instituto de Ingeniería de Software Experimental 67661
Kaiserslautern, Alemania
rech@iese.fraunhofer.de

Klaus-Dieter Althoff recibió su doctorado. y Habilitación en informática de la Universidad de Kaiserslautern, Alemania. Fue/es responsable de una serie de proyectos de investigación sobre sistemas basados en el conocimiento, aprendizaje automático, razonamiento basado en casos, gestión del conocimiento (KM) y fábrica de experiencias. Fue/es miembro de varios comités de programas, revisor de muchas revistas científicas y copresidente de varios eventos internacionales, por ejemplo, la 3ª Conferencia sobre KM Profesional (WM 2005). De 1997 a 2004 fue responsable de Sistemas y Procesos basados en la experiencia en el Instituto Fraunhofer de Ingeniería Experimental de Software como líder de grupo/jefe de departamento. Desde abril de 2004 ocupa la cátedra de Gestión de Datos y Conocimiento en la Universidad de Hildesheim.

Jörg Rech estudió Ciencias de la Computación en la Universidad de Kaiserslautern. Recibió la licenciatura (Vordiplom) y la maestría (Dip-lom) en informática con especialización en ciencias eléctricas de la Universidad de Kaiserslautern, Alemania. Fue asistente de investigación en el grupo de investigación en ingeniería de software (AGSE) dirigido por el Prof. Dieter Rombach en la Universidad de Kaiserslautern.

Actualmente es investigador y director de proyectos en el Instituto Fraunhofer de Ingeniería de Software Experimental. Su investigación se centra principalmente en el descubrimiento de conocimientos en repositorios de software, el descubrimiento de defectos, la extracción de códigos, la recuperación de códigos, la reutilización automatizada de códigos, el análisis de software y la gestión del conocimiento.