Artificial Intelligence and Software Engineering: Status and Future Trends

Jörg Rech, Klaus-Dieter Althoff

The disciplines of Artificial Intelligence and Software Engineering have many commonalities. Both deal with modeling real world objects from the real world like business processes, expert knowledge, or process models. This article gives a short overview about these disciplines and describes some current research topics against the background of common points of contact.

1 Introduction

During the last decades the disciplines of Artificial Intelligence (AI) and Software Engineering (SE) have developed separately without much exchange of research results. In AI we researched techniques for the computations that made it possible to perceive, reason, and act. Research in SE was concerned with supporting human beings to develop better software faster.

Today, several research directions of both disciplines come closer together and are beginning to build new research areas. *Software Agents* play an important role as research objects in Distributed AI (DAI) as well as in agent-oriented software engineering (AOSE). *Knowledge-Based Systems* (KBS) are being investigated for learning software organizations (LSO) as well as knowledge engineering. *Ambient Intelligence (AmI)* is a new research area for distributed, non-intrusive, and intelligent software systems both from the direction of how to build these systems as well as how to design the collaboration between ambient systems. Last but not least, *Computational Intelligence (CI)* plays an important role in research about software analysis or project management as well as knowledge discovery in databases or machine learning.

Furthermore, in the last five to ten years several books, journals, and conferences have focused on the intersection between AI and SE. The international conference and associated journal *Automated Software Engineering* (ASE) presents research about formal and autonomic approaches to support SE [2]. Similar topics with a stronger focus on KBS and knowledge management are published in the international conference and associated journal of *Software Engineering and Knowledge Engineering* (IJSEKE) [1].

In this paper, we give a short overview about the status and future trends in the intersection between AI and SE. We focus on the topics software agents, KBS, AmI, and CI as the areas covered by the contributions of this special issue. In Section 2 we describe the disciplines AI and SE. The focused topics are described in more detail in Section 3. Finally, in Section 4 we give an outlook for the next years and present new challenges for both disciplines.

2 Artificial Intelligence and Software Engineering

This section will shed some light on the disciplines AI and SE for those not familiar with the other discipline.

Aspects of Artificial Intelligence

There is a general agreement in the AI community that the discipline of AI was born at the Dartmouth conference in 1956. According to Winston [81] “AI is the study of the computations that make it possible to perceive, reason, and act”. Wachsmuth [78] assumes this definition and points out that, “AI differs from most of psychology because of its greater emphasis on computation, and it differs from most of computer science because of its greater emphasis on perception, reasoning, and action”. As a field of academic study, many AI researchers reach to understand intelligence by becoming able to produce effects of intelligence: intelligent behavior. One element in AI’s methodology is that progress is sought by building systems that perform: synthesis before analysis [78]. “Systems are good science”, as Hendler said [34]. Or more drastically by Wachsmuth [78]: “it is not the aim of AI to build intelligent machines having understood natural intelligence, *but* to understand natural intelligence by building intelligent machines”. Even more strikingly Aaron Sloman puts it this way (by citing his colleague Russel Beale): "AI can be defined as the attempt to get real machines to behave like the ones in the movies”. In addition, he points out that AI has two main strands, a scientific strand and an engineering strand, which overlap considerably in their concepts, methods, and tools, though their objectives are very different.

This view is supported by Wahlster [79] who clarifies that AI has two different types of goals, one motivated by cognitive science, the other by the engineering sciences (cf. Figure 1).

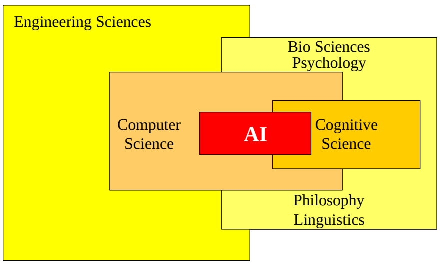

Figure 1 AI and Related Research Areas (adapted from [79])

A further sub-division (adapted from Richter [59] and Abecker [4]) of AI into sub-fields, methods, and techniques is shown in Figure 2.

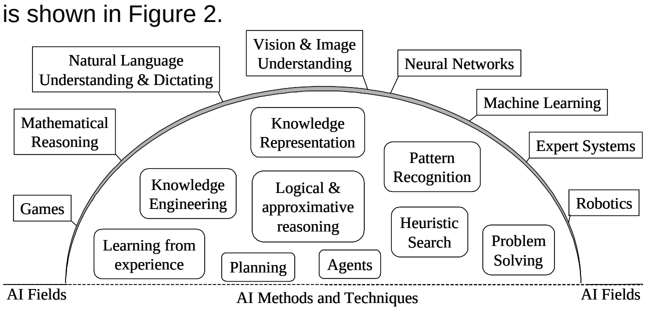

Figure 2 AI Fields, Methods, and Techniques

For SE the scientific strand orientating towards cognitive science and humanities in general could be a helpful guidance for interdisciplinary research. Of course, there is a strong overlap between SE and the engineering strand of AI. An important part of the latter are KBS.

Richter [59] defines three different levels as essential for describing KBS: the cognitive layer (human-oriented, rational, informal), the representation layer (formal, logical), and the implementation layer (machine-oriented, data structures and programs). These levels are shown in Figure 3. Between the knowledge utterance and its machine utilization several transformations have to be performed (thick arrows). They point to the direction of increased structuring within the layers and proceed from the cognitive form to a more formal and more efficiently processed form. The letter A is a reminder for Acquisition (which is human-oriented) while C is a shorthand for Compilation (machine-oriented). Each syntactic result in the range of a transformation between layers has to be associated with the meaning in the domain of the transformation. The most interesting and difficult arrow is the inverse transformation back to the cognitive layer; it is usually called explanation.

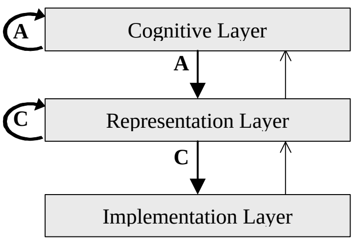

Figure 3 The Three Levels of Knowledge-Based Systems

Why is AI interesting for researchers from SE? It can provide the initial technology and first (successful) applications as well as a testing environment for ideas. The inclusion of research supports the enabling of human-enacted processes and increases user acceptance. AI technology can help to base the overall SE method on a concrete technology, providing sufficient detail for the initial method description, and through the available reference technology clarifying the semantics of the respective method. In addition, other AI techniques naturally substituting/extending the chosen technology can be used for improved versions of the SE method.

Aspects of Software Engineering

The discipline of SE was born 1968 at the NATO conference in Garmisch-Partenkirchen, Germany [52, 71] where the term “SE crisis” was coined. Its main concern is the efficient and effective development of high-qualitative and mostly very large software systems. The goal is to support software engineers and managers in order to develop better software faster with (intelligent) tools and methods.

Since its beginning several research directions developed and matured in this broad field. Figure 4 shows the software development reference model integrating important phases in a software lifecycle. *Project Engineering* is concerned with the acquisition, definition, management, monitoring, and controlling of software development projects as well as the management of risks emerging during project execution. Methods from *Requirements Engineering* are developed to support the formal and unambiguous elicitation of software requirements from the customers, to improve the usability of the systems, and to establish a binding and unambiguous definition of the resulting system during and after software project definition. The research for *Software Design & Architecture* advances techniques for the development, management, and analysis of (formal) descriptions of abstract representations of the software system as well as required tools and notations (e.g., UML). Techniques to support the professional *Programming* of software are advanced to develop highly maintainable, efficient, and effective source code. *Verification & Validation* is concerned with the planning, development, and execution of (automated) tests and inspections (formal and informal) in order to discover defects or estimate the quality of parts of the software. Research for Implementation & Distribution is responsible for the development of methods for the introduction at the customer’s site, support during operation, and integration in existing IT infrastructure.

After delivery to the customer software systems typically switch into a *Software Evolution* phase. Here the focus of research lies on methods in order to add new and perfect existing functions of the system. Similarly, in the parallel phase *Software Maintenance* techniques are developed for the adaptation to environmental changes, prevention of foreseeable problems, and correction of noticed defects. If the environment changes dramatically or further enhancements are impossible the system either dies or enters a *Reengineering* phase. Here techniques for software understanding and reverse engineering of software design are used to port or migrate a system to a new technology (e.g., from Ada to Java or from a monolithic to a client/server architecture) and obtain a maintainable system.

Since the eighties the systematic reuse and management of experiences, knowledge, products, and processes was developed and named Experience Factory (EF) [16]. This field, also known as *Learning Software Organization* (LSO), researches methods and techniques for the management, elicitation, and adaptation of reusable artifacts from SE projects.

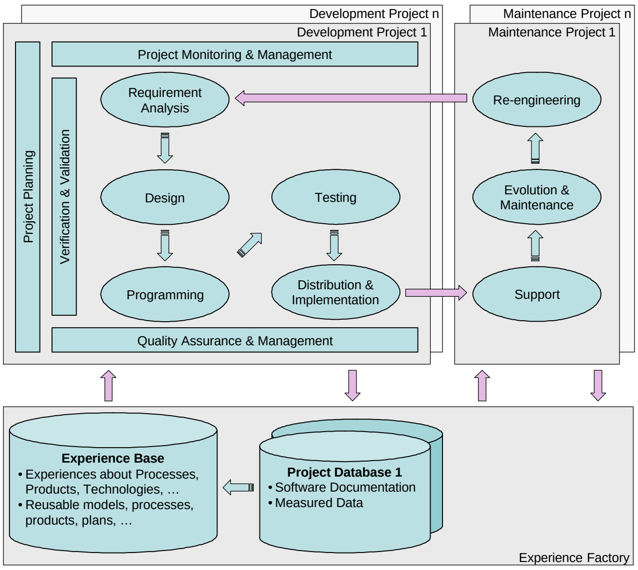

Figure 4 Software Development Reference Model

Why is SE for AI researchers interesting? It supports systematic AI application development, the operating of AI applications in real-life environments, as well as evaluating (e.g., [6]), maintaining (e.g., [54]), continuously improving, and systematically comparing them with alternative approaches (e.g., another modeling method). SE also supports the systematic definition of the respective application domains, e.g., through scoping methods [67].

3 Intersections between AI and SE

While the intersections between AI and SE are currently rare they are multiplying and growing. First points of contact emerged from the application of techniques from one discipline to the other [55].

Today, methods and techniques from both disciplines support the practice and research in the respectively other research area. Figure 5 depicts some research areas in AI and SE as well as their intersections.

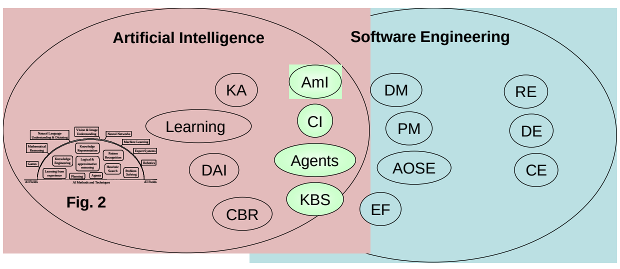

Figure 5 Research Areas in AI and SE and their Intersections

Systematic software development (including Requirements Engineering (RE), Engineering of Designs (DE), or source code (CE)) or project management (PM) methods help to build intelligent systems while using advanced data analysis techniques. Knowledge Acquisition (KA) techniques [21] help to build EF and intelligent ambient systems like Domain Modeling (DM) techniques support the construction of requirements for software systems and product lines. Case-based Reasoning (CBR) is used to support the retrieval and management of data in EF. Information Agents are used in SE to simulate development processes or to distribute and explain change requests.

Agent-Oriented Software Engineering

Software Agents are typically small intelligent systems that cooperate to reach a common goal. These agents are a relatively new area where research from KI and SE intersects. From the AI side the focus in this field lies on even more intelligent and autonomous systems to solve more complex problems using communication languages between agents. In SE agents are seen as systems that need more or less specialized formal methods for their development, verification, validation, and maintenance.

Agent-Oriented Software Engineering (AOSE) (a.k.a. Agent Based Software Engineering (ABSE)) as related to object-oriented SE (OOSE) is centered around systems where objects in a model of a software system are intelligent, autonomous, and proactive. Currently the systematic development and representation of software agents is researched and languages for their representation during development, like the Agent UML [19], were created. For example, several methods like MASSIVE by Lind [45], GAIA by Wooldridge et al. [84], MESSAGE by Caire et al. [25], TROPOS by Castro et al. [26], or MAS-CommonKADS by Iglesias et al. [38] were developed.

Agents and AOSE are applied in many areas like intelligent and agent-based user interfaces to improve system usability, trading agents in eCommerce to maximize profits, or assisting agents in everyday work to automate common tasks (e.g., booking hotel rooms) [39]. Furthermore software agents are increasingly used to simulate real world domains (e.g., traffic control) or work processes in SE. But agent technology is not a shiny new paradigm without problems – some pitfalls for AOSE are described by Wooldridge in [85].

A state of the art survey about agent-oriented SE by Tveit summarized previous publications and methods [76]. Formal methods for the specification and verification of systems were developed in order to describe, proof, and check the goals, beliefs, and interaction of agents and agent systems [82, 83].

Today, more and more workshops are being organized covering the common ground of intelligent agents and SE. Examples for communities in this field are the Workshop Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2004) [46], the International Workshop Series on Agent-Oriented Software Engineering (AOSE 2004) [33], the International Workshop on Agent-Oriented Methodologies (AO 2003), International Workshop on Radical Agent Concepts (WRAC 2003) [75], or the International Joint Conference on Autonomous Agents & Multi-Agent Systems

(AAMAS 2004) [64]. Workshops on more formal aspects of AOSE are FAABS or KIMAS [36, 37].

As summarized in the Agent Technology Roadmap by the AgentLink network future research is required to enable agent systems to adapt autonomously to communication languages, behavioral patterns, or to support the reuse of knowledge bases [47].

Knowledge-Based Software Engineering

SE is a highly dynamic field in terms of research and knowledge, and it depends heavily upon the experience of experts for the development and advancement of its methods, tools, and techniques. For example, the tendency to define and describe "best practices" or "lessons learned" is quite distinctive in the literature [13]. As a consequence, it was the SE field where an organization, the EF, was introduced that was explicitly responsible to systematically deal with experience. An EF is a logical and/or physical infrastructure for continuous learning from experience and includes an experience base (EB) for the storage and reuse of knowledge. The EF approach was invented in the mid-eighties [15]. As practice shows, it is substantial for the support of organizational learning that the project organization and the learning organization are separated [16] (see also Figure 4).

The initial example for an operating EF was the NASA SE Laboratory [62]. In the meantime EF applications were developed in the USA and also in Europe [8, 73]. The great amount of successful EF applications gave the ignition to study LSOs more intensively regarding the methodology for building up and running an EF [63]. This also includes the definition of related processes, roles, and responsibilities and, last but not least, the technical realization. The most detailed methodology for the build-up of an EF/EB on project knowledge also for the presentation of the according processes is given in [73], an extension concerning evaluation, maintenance, and architecture can be found in [54].

EF is increasingly emerging towards a generic approach for reuse of knowledge and especially experience. This includes also applications independent of the SE domain, e.g., supporting the continuous improvement process in hospitals [9], the field of help-desk and service support [72], and the management of "non-software"-projects [23]. Future trends in the scope of EF include the detailing of all necessary policies, validation, and empirical evaluation [17, 73], gaining experience with the technical realization of huge EFs [58], the integration with the according business processes [10], and the running of EFs [53].

In the areas of Cognitive Science and AI, CBR emerged in the late seventies and early eighties as a model for human problem solving and learning [65, 66]. In AI, this led to a focus of KBS on experience (case-specific knowledge) in the late eighties and beginning nineties, mostly in the form of problem-solution cases [14]. Since several years there has been a strong tendency in the CBR community [7] to develop methods for dealing with more complex applications. One example is the use of CBR for KM [5], another one is its use for SE [69]. A very important issue here is the integration of CBR with EF: Since the mid-nineties CBR is used both on the organizational EF process level as well as the technical EB implementation level [12, 35, 74]. Meanwhile this approach establishes itself more and more [7, 20, 40]. Usually product- and process-oriented approaches are used independently from each other, or as alternatives. As a first step a deep integration of the approaches of EF and CBR has been achieved [7, 11, 54, 73].

An overview on relevant approaches for knowledge-based SE is given in [11, 13, 27, 28, 43, 54, 73]. Relevant events are part of the LSO, SEKE, ASE, and CBR (www.iccbr.org) event series (as well as the corresponding journals including the International Journal on Knowledge and Information Systems (KAIS)). Conferences and workshops (again as well as journals) on knowledge management are of interest if they have a concrete relationship to software-related issues. Further relevant events are the Joint Conference on Knowledge-Based SE (JCKBSE) 2002 [80] and 2004, and the workshops on Knowledge Oriented Maintenance (KOM 2004), or Knowledge-Based Object-Oriented SE (KBOOSE 2002).

Computational Intelligence & KDD

The research area CI has recently observed an increasing interest from researchers of both disciplines. Techniques like neural networks, evolutionary algorithms, or fuzzy systems are increasingly applied and adapted for specific SE problems. They are used to estimate the progress of projects to support software project management [24], for the discovery of defect modules to ensure software quality, or to plan software testing and verification activities to minimize the effort for quality assurance [41, 44, 56].

In a state-of-the-art survey about KDD in SE, Mendonca and Sunderhaft summarized previous publications as well as several mining techniques and tools [48]. One of the first publications that explicitly collected contributions to KDD for SE data was the IJSEKE Special Issue in 1999 [50]. More and more workshops are being organized on CI and SE. Examples for workshops are WITSE (Intelligent Technologies for SE), MSR (Mining Software Repositories), DMSK (Data Mining for SE and Knowledge Engineering), and SCASE (Soft Computing applied to SE).

Today, many application areas for KDD in SE have been established in fields like quality management, project management, risk management, software reuse, or software maintenance. For example, Khoshgoftaar et al. applied and adapted classification techniques to software quality data [42]. Dick researched determinism of software failures with time series analysis and clustering techniques [31]. Cook and Wolf used the Markov approach to mine process and workflow models from activity data [29]. Pozewauning examined the discovery and classification of component behavior from code and test data to support the reuse of software [57]. Michail used association rules to detect reuse patterns (i.e., typical usage of classes from libraries) [49]. As an application of KDD in software maintenance, Shirabad developed an instrument for the extraction of relationships in software systems by inductive methods based on data from various software repositories (e.g., update records, versioning systems) to improve impact analysis in maintenance activities [70]. Zimmermann and colleagues pursue the same goal using a technique similar to CBR. In order to support software maintainers with related experiences in form of association rules about changes in software systems they mined association rules from a versioning system by collecting transactions including a specific change [86].

Morasca and Ruhe built a hybrid approach for the prediction of defect modules in software systems with rough sets and logistic regression based on several metrics (e.g., LOC) [51].

Future research in this field is required to analyze formal project plans for risk discovery, to acquire project information for project management, or directly mine software representations (e.g., UML, sourcecode) to detect defects and flaws early in development.

Ambient Intelligence

The idea behind AmI are sensitive, adaptive, and reactive systems that are informed about the user's needs, habits, and emotions in order to support them in their daily work [30]. Therefore, techniques for autonomous, intelligent, robust, and self-learning systems are needed to enable communication between systems (machine-machine interfaces and ontologies) or users and systems (human-machine interfaces).

AmI is based on several research areas, like ubiquitous and pervasive computing, intelligent systems, and context awareness [68]. Research for AmI tries to build an environment similar to the previously mentioned research areas like intelligent software agents, KBS, as well as knowledge discovery (e.g. to detect and analyze foreign systems and software).

There are several AI research areas for the development of smart algorithms for AmI applications [77], e.g., user profiling, context awareness, scene understanding [3], or planning and negotiation tasks [30]. Research from the SE side is concerned with model-driven development for mobile computing [30], the verification of mobile code [61], the specification of adaptive systems [60], or the design of embedded systems [18]. Additionally, we need intelligent human interfaces from a usability perspective that translate between users and a new configuration of the ambient system. A fusion of these two fields could be established in order to analyze and evaluate foreign software systems that try to connect with the own system to be executed on its hardware.

Currently, research for AmI is primarily funded by the EU as well as the German National Science Foundation (DFG). For example, several scenarios describing the vision of the EU were published in [32], and the integrated project WearIT@Work is funded at the University of Bremen which emphasizes AmI for work processes. In Germany, the DFG funded the „Forschungsschwerpunkt Ambient Intelligence“ at the Technical University of Kaiserslautern (http://www.eit.uni-kl.de/AmI).

Various workshops and conferences on AmI were established to foster exchange about AmI. Examples for these meetings are the European Symposium on Ambient Intelligence (EUSAI) [3], Workshop on Ambient Intelligence (WAI), Workshop on Ambient Intelligence for Scientific Discovery (AMDI), Workshop on Ambient intelligence @ Work, International Conference on Ubiquitous Computing (UbiComp) [22], or the Workshop on Agents for Ubiquitous Computing (UbiAgents).

4 Outlook

It is obvious from this overview that strong ties exist between artificial intelligence and software engineering that offer a great potential for future research. Many new applications and research fields of interest to both disciplines will develop covering knowledge-based systems for learning software organizations, the development of computational intelligence and knowledge discovery techniques for software artifacts, agent-oriented SE, or professional development of ambient intelligence systems.

As a conclusion, we have identified several research fields interesting for both disciplines. The articles in the remainder of this special issue elaborate on specific research problems in these intersections.

References

[1] *SEKE 2002: the 14th International Conference on Software Engineering and Knowledge Engineering*. Ischia, Italy: New York: ACM, 2002.

[2] *Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE)*. Montreal, Quebec, Canada: IEEE Computer Society, ISBN: 0-769-52035-9, 2003.

[3] Aarts E. H. L., *Ambient intelligence: First European symposium (EUSAI 2003)*, vol. 2875. Veldhoven, The Netherlands: Springer-Verlag, 3540204180, 2003.

[4] Abecker A., "Wissensbasierte Systeme," Berufsakademie Mosbach, Lecture Notes 2002.

[5] Aha D. W., Becerra-Fernandez I., Maurer F., and Muñoz-Avila H., *Exploring synergies of knowledge: management & case-based reasoning*. Menlo Park, Calif.: AAAI Press, ISBN: 1-577-35094-4, 1999.

[6] Althoff K.-D., *Evaluating Case-Based Reasoning Systems: The Inreca Case Study*. Postdoctoral Thesis (Habilitationsschrift). Dept. of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany, 1997.

[7] Althoff K.-D., "Case-based reasoning," in *Handbook of software engineering & knowledge engineering*, vol. 1, S.-K. Chang, Ed. Singapore: World Scientific, 2001, pp. 549-587.

[8] Althoff K. D., Becker K. U., Decker B., Klotz A., Leopold E., Rech J., and Voss A., "The indiGo project: enhancement of experience management and process learning with moderated discourses." Berlin, Germany: Springer Verlag, 2002, pp. 53-79.

[9] Althoff K.-D., Bomarius F., Müller W., and Nick M., "Using a Case-Based Reasoning for Supporting Continuous Improvement Processes," presented at German Workshop on Machine Learning, Leipzig, 1999.

[10] Althoff K.-D., Decker B., Hartkopf S., Jedlitschka A., Nick M., and Rech J., "Experience Management: The Fraunhofer IESE Experience Factory," presented at Industrial Conference Data Mining, Leipzig, 2001.

[11] Althoff K.-D. and Nick M. M., "How to Support Experience Management with Evaluation - Foundations,

Evaluation Methods, and Examples for Case-Based Reasoning and Experience Factory." Berlin: Springer Verlag, 2004 (forthcoming, Accepted for LNAI series).

[12] Althoff K.-D. and Wilke W., "Potential uses of case-based reasoning in the experience-based construction of software systems," presented at 5th German Workshop in Case-Based Reasoning (GWCBR'97), University of Kaiserslautern, 1997.

[13] Aurum A., *Managing software engineering knowledge*. Berlin: Springer, ISBN: 3540003703, 2003.

[14] Bartsch-Spörl B., "Ansätze zur Behandlung von fallorientiertem Erfahrungswissen in Expertensystemen," *Künstliche Intelligenz*, pp. 32-36, 1987.

[15] Basili V. R., *Quantitative evaluation of software methodology*. College Park, Md.: University of Maryland, 1985.

[16] Basili V. R., Caldiera G., and Rombach H. D., "Experience Factory," in *Encyclopedia of Software Engineering*, vol. 1, J. J. Marciniak, Ed. New York: John Wiley & Sons, 1994, pp. 469-476.

[17] Basili V. R., Shull F., and Lanubile F., "Building knowledge through families of experiments," *IEEE Transactions on Software Engineering*, vol. 25, pp. 456-73, 1999.

[18] Basten T., Geilen M., and Groot H. d., "Ambient intelligence: impact on embedded system design," T. Basten, M. Geilen, and H. d. Groot, Eds.: Kluwer Academic Publishers, 2003.

[19] Bauer B., Müller J. P., and Odell J., "Agent UML," *International Journal of Software Engineering and Knowledge Engineering*, vol. 11, pp. 207-230, 2001.

[20] Bergmann R., Althoff K.-D., Breen S., Göker M., Manago M., Traphöner R., and Wess S., *Developing industrial case-based reasoning applications : the INRECA methodology*, vol. 1612, 2nd ed. Berlin: New York, ISBN: 3-540-20737-6, 2003.

[21] Birk A., Surmann D., and Althoff K. D., "Applications of knowledge acquisition in experimental software engineering," presented at 11th European Knowledge Acquisition Workshop (EKAW'): Knowledge Acquisition, Modeling, and Management, Berlin, Germany, 1999.

[22] Borriello G., *UbiComp: 4th International Conference on Ubiquitous Computing*, vol. 2498. Berlin: ACM, 2002.

[23] Brandt M., Ehrenberg D., Althoff K.-D., and Nick M. M., "Ein fallbasierter Ansatz für die computergestützte Nutzung von Erfahrungswissen bei der Projektarbeit," presented at 5th Internationale Tagung Wirtschaftsinformatik (WI'01) Information Age Economy, Heidelberg, 2001.

[24] Brandt M. and Nick M., "Computer-supported reuse of project management experience with an experience base," presented at Third International Workshop on Advances in Learning Software Organizations (LSO 2001), Berlin, Germany, 2001.

[25] Caire G., Coulier W., Garijo F., Gomez J., Pavon J., Leal F., Chainho P., Kearney P., Stark J., Evans R., and Massonet P., "Agent oriented analysis using MESSAGE/UML," presented at Second International Workshop on Agent Oriented Software Engineering (AOSE 2001), 2002.

[26] Castro J., Kolp M., and Mylopoulos J., "Towards requirements-driven information systems engineering: the Tropos project," *Information Systems*, vol. 27, pp. 367-91, 2002.

[27] Chang S. K., "Handbook of software engineering & knowledge engineering. Vol. 1, Fundamentals," World Scientific, 2001.

[28] Chang S. K., "Handbook of software engineering & knowledge engineering. Vol. 2, Emerging technologies," World Scientific, 2002.

[29] Cook J. E. and Wolf A. L., "Discovering models of software processes from event-based data," *ACM Transactions on Software Engineering and Methodology*, vol. 7, pp. 215-49, 1998.

[30] Da Costa O. and Punie Y., *Science and Technology Roadmapping: Ambient Intelligence in Everyday Life (Ami@Life)*: Unpublished IPTS working paper, 2003.

[31] Dick S. H., *Computational Intelligence in Software Quality Assurance*. PhD Thesis. Department of Computer Science and Engineering, College of Engineering, University of South Florida, 2002.

[32] Ducatel K., *Scenarios for ambient intelligence in 2010*. Luxembourg: Office for Official Publications of the European Communities, ISBN: 9289407352, 2001.

[33] Giorgini P., Müller J. P., and Odell J. J., *Proceedings of the fourth international workshop on Agent-oriented software engineering IV (AOSE 2003)*, vol. 2935. Melbourne, Australia: Springer, 3540208267, 2004.

[34] Hendler J., "Experimental AI Systems," *Journal of Experimental and Theoretical AI*, vol. 7, pp. 1-5, 1995.

[35] Henninger S., "Developing domain knowledge through the reuse of project experiences," *SIGSOFT Software Engineering Notes*, vol. Aug. 1995, pp. 186-195, 1995.

[36] Hexmoor H., *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS 2003)*. Cambridge, MA, USA: IEEE, ISBN: 0-780-37958-6, 2003.

[37] Hinchey M. G., *Formal approaches to agent-based systems: Second international workshop (FAABS 2002)*, vol. 2699. Greenbelt, MD, USA: Springer, ISBN 3-540-40665-4, 2002.

[38] Iglesias C. A., Garijo M., Gonzalez J. C., and Velasco J. R., "Analysis and design of multiagent systems using MAS-CommonKADS," presented at Agent Theories, Architectures, and Languages. 4th International Workshop, ATAL'97, Berlin, Germany, 1998.

[39] Jennings N. R., Sycara K., and Wooldridge M., "A roadmap of agent research and development," *Vivek*, vol. 12, pp. 38-66, 1999.

[40] Kalfoglou Y., Menzies T., Althoff K. D., and Motta E., "Meta-knowledge in systems design: panacea... or undelivered promise?," *Knowledge Engineering Review*, vol. 15, pp. 381-404, 2000.

[41] Khoshgoftaar T. M., *Software engineering with computational intelligence*, vol. 731. Boston: Kluwer Academic Publishers, ISBN: 1-402-07427-1, 2003.

[42] Khoshgoftaar T. M., Allen E. B., Jones W. D., and Hudepohl J. P., "Data mining of software development databases," *Software Quality Journal*, vol. 9, pp. 161-76, 2001.

[43] Last M., Kandel A., and Bunke H., *Artificial Intelligence Methods in Software Testing*, 2003.

[44] Lee J., *Software engineering with computational intelligence*. Berlin: New York, ISBN: 3540004726 (alk. paper) LCCN: 2003-42588, 2003.

[45] Lind J., *Iterative software engineering for multiagent systems : the MASSIVE method*, vol. 1994. Berlin: Springer, 3540421661, 2001.

[46] Lucena C. J. P. d., Garcia A. F., Romanovsky A. B., Castro J., and Alencar P. S. C., *Proceedings of the 2nd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS 2003)*, vol. 2940. Berlin: Springer, ISBN 3-540-21182-9, 2003.

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

[48] Mendonca M. and Sunderhaft N. L., "Mining Software Engineering Data: A Survey," DACS: Data & Analysis Center for Software, Technical Report 1999.

[49] Michail A., "Data mining library reuse patterns using generalized association rules," presented at Proceedings of the 2000 International Conference on Software Engineering (ICSE 2000), New York, 2000.

[50] Morasca S. and Ruhe G., "Special Issue on: Knowledge Discovery from Empirical Software Engineering Data," *International Journal of Software Engineering and Knowledge Engineering*, vol. 9, pp. 495-498, 1999.

[51] Morasca S. and Ruhe G., "A hybrid approach to analyze empirical software engineering data and its application to predict module fault-proneness in maintenance," *Journal of Systems and Software*, vol. 53, pp. 225-237, 2000.

[52] Naur P. and Randell B., "Software Engineering: Report of a conference," sponsored by the NATO Science Committee, Garmisch, Germany 7-11 Oct. 1968.

[53] Nick M., Althoff K. D., and Tautz C., "Systematic maintenance of corporate experience repositories," *Computational Intelligence*, vol. 17, pp. 364-86, 2001.

[54] Nick M. M., *Building and Running Long-Lived Experience-Based Information Systems*. PhD Thesis. Dept. of Computer Science, University of Kaiserslautern, Kaiserslautern, to be submitted in 2004.

[55] Partridge D., *Artificial intelligence and software engineering : understanding the promise of the future*. Chicago: Glenlake Pub. Co. Fitzroy Dearborn Publishers, ISBN 1-57958-062-9, 2000.

[56] Pedrycz W. and Peters J. F., *Computational intelligence in software engineering*. Singapore: River Edge N.J., ISBN: 9-810-23503-8, 1998.

[57] Pozewaunig H., *Mining Component Behavior to Support Software Retrieval*. PhD Thesis. Institut für Informatik-Systeme der Fakultät für Wirtschaftswissenschaften und Informatik, Universität Klagenfurt, Klagenfurt, 2001.

[58] Rech J., Decker B., and Althoff K.-D., "Using Knowledge Discovery Technology in Experience Management Systems," presented at Workshop "Maschinelles Lernen (FGML01)", Universität Dortmund, 2001.

[59] Richter M. M., "Artificial Intelligence," University of Calgary, Canada, Lecture Notes 2004.

[60] Roman G.-C., *Special issue on software engineering for mobility*. Dordrecht, Netherlands: Kluwer Academic, Series ISSN: 0928-8910, 2002.

[61] Roman G. C., Picco G. P., and Murphy A. L., "Software Engineering for Mobility: A Roadmap," presented at Future of Software Engineering Track of 22nd ICSE, Limerick, Ireland, 2000.

[62] Rombach H. D. and Ulery B. T., "Establishing a measurement based maintenance improvement program: lessons learned in the SEL," University of Maryland, College Park, Md. 1989.

[63] Ruhe G. and Bomarius F., "Proceedings of Learning software organizations (LSO): methodology and applications," presented at 11th International Conference on Software Engineering and Knowledge Engineering, SEKE'99, Kaiserslautern, Germany, 1999.

[64] Sandholm T. and Wooldridge M. J., *2nd International Joint Conference on Autonomous Agents and Multiagent System (AAMAS 2003)*. Melbourne, Australia: ACM, ISBN: 1-581-13685-4, 2003.

[65] Schank R. C., *Dynamic memory : a theory of reminding and learning in computers and people*. Cambridge Cambridgeshire: New York Cambridge University Press, ISBN: 0521248582, 1982.

[66] Schank R. C., Abelson R. P., and joint a., *Scripts, plans, goals, and understanding : an inquiry into human knowledge structures*. Hillsdale, N.J. L. Erlbaum Associates: New York, ISBN: 0470990333, 1977.

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

[68] Shadbolt N., "From the Editor in Chief - Ambient Intelligence," *IEEE intelligent systems & their applications*, vol. 18, pp. 2 (2 pages), 2003.

[69] Shepperd M., "Case-Based Reasoning and Software Engineering," in *Managing Software Engineering Knowledge*, A. Aurum, Ed. Berlin: Springer, 2003.

[70] Shirabad J. S., *Supporting Software Maintenance by Mining Software Update Records*. PhD Thesis. School of Information Technology and Engineering, University of Ottawa, Ottawa, Ontario, Canada, 2003.

[71] Simons C. L., Parmee I. C., and Coward P. D., "35 years on: to what extent has software engineering design achieved its goals?," *IEE Proceedings Software*, vol. 150, pp. 337-50, 2003.

[72] Stolpmann M. and Wess S., *Optimierung der Kundenbeziehungen mit CBR-Systemen - Intelligente Systeme für E-Commerce und Support*. Bonn: Addison Wes-ley Longmann, 1998.

[73] Tautz C., *Customizing Software Engineering Experience Management Systems and Related Processes for Sharing Software Engineering Experience*. Ph.D

Thesis. Department of Computer Science, University of Kaiserslautern, Germany, Kaiserslautern, 2000.

[74] Tautz C. and Althoff K. D., "Using case-based reasoning for reusing software knowledge," *Case Based Reasoning Research and Development*, pp. Plaza, E.. - Berlin, Germany, Germany Springer-Verlag, 1997, xiii+648 156-65, 18 Refs., 1997.

[75] Truszkowski W., Rouff C., and Hinchey M., *Proceedings of the first international workshop on Radical Agent Concepts (WRAC 2002)*, vol. 2564. McLean, VA, USA: Springer, 3-540-40725-1, 2003.

[76] Tveit A., "A survey of Agent-Oriented Software Engineering," presented at Proc. of the First NTNU CSGS Conference, 2001.

[77] Verhaegh W. F. J., Aarts E. H. L., and Korst J., *Algorithms in ambient intelligence*, vol. 2. Boston: Dordrecht, ISBN: 140201757X, 2004.

[78] Wachsmuth I., "The Concept of Intelligence in AI," in *Prerational Intelli-gence - Adaptive Behavior and Intelligent Systems without Symbols and Logic*, vol. 1, H. Cruse, D. J., and RitterH., Eds. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000, pp. 43-55.

[79] Wahlster W., "Einführung in die Methoden der Künstlichen Intelligenz," University of Saarbrücken, Germany, Lecture Notes 2002.

[80] Welzer T., Yamamoto S., and Rozman I., *Proceedings of the fifth Joint Conference on Knowledge-Based Software Engineering*. Amsterdam: Washington DC, ISBN: 1586032747, 2002.

[81] Winston P. H., *Artificial intelligence*, 3rd (repr. with corrections 1993) ed. Reading, Mass.: Addison-Wesley, ISBN: 0-201-53377-4, 1993.

[82] Wooldridge M., "Agent-based software engineering," *IEE Proceedings Software Engineering*, vol. 144, pp. 26-37, 1997.

[83] Wooldridge M. and Ciancarini P., "Agent-oriented software engineering: the state of the art," presented at 1st International Workshop on Agent Oriented Software Engineering (AOSE 2000), Berlin, Germany, 2000.

[84] Wooldridge M., Jennings N. R., and Kinny D., "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi Agent Systems*, vol. 3, pp. 285-312, 2000.

[85] Wooldridge M. J. and Jennings N. R., "Software engineering with agents: pitfalls and pratfalls," *IEEE Internet Computing*, vol. 3, pp. 20-7, 1999.

[86] Zimmermann T., Weißgerber P., Diehl S., and Zeller A., "Mining Version Histories to Guide Software Changes," presented at 26th International Conference on Software Engineering (ICSE), Edinburgh, UK, 2004.

Contact

Dr. habil. Klaus-Dieter Althoff
Data and Knowledge Management
University of Hildesheim
PO Box 101363
31113 Hildesheim, Germany
althoff@dwm.uni-hildesheim.de

Jörg Rech
Fraunhofer IESE
Institute for Experimental Software Engineering
67661 Kaiserslautern, Germany
rech@iese.fraunhofer.de

Klaus-Dieter Althoff received Ph.D. and Habilitation in computer science from the University of Kaiserslautern, Germany. He was/is responsible for a number of research projects on knowledge-based systems, machine learning, case-based reasoning, knowledge management (KM), and experience factory. He was/is member of various program committees, reviewer for many scientific journals, and co-chair of a number of international events, e.g., the 3rd Conference on Professional KM (WM 2005). From 1997 to 2004 he was responsible for Experience-based Systems and Processes at the Fraunhofer Institute for Experimental Software Engineering as group leader/department head. Since April 2004, he represents the chair for Data and Knowledge Management at the University of Hildesheim.

Jörg Rech studied Computer Science at the University of Kaiserslautern. He received the B.S. (Vordiplom) as well as the M.S. (Diplom) in computer science with a minor in electrical science from the University of Kaiserslautern, Germany. He was a research assistant at the software engineering research group (AGSE) led by Prof. Dieter Rombach at the university of Kaiserslautern. Currently, he is a researcher and project manager at the Fraunhofer Institute for Experimental Software Engineering. His research mainly concerns knowledge discovery in software repositories, defect discovery, code mining, code retrieval, automated code reuse, software analysis, and knowledge management.