Preface

Software Science is a discipline that studies the theoretical framework of software as instructive and behavioral information, which can be embodied and executed by generic computers in order to create expected system behaviors and machine intelligence. Intelligence science is a discipline that studies the mechanisms and theories of abstract intelligence and its paradigms such as natural, artificial, machinable, and computational intelligence. The convergence of software and intelligent sciences forms the transdisciplinary field of computational intelligence, which provides a coherent set of fundamental theories, contemporary denotational mathematics, and engineering applications.

This book, entitled *Software and Intelligent Sciences: New Transdisciplinary Findings*, is the first volume in the IGI Series of Advances in Software Science and Computational Intelligence. The book encompasses 29 chapters of expert contributions selected from the *International Journal of Software Science and Computational Intelligence* during 2009. The book is organized in five sections on: (i) Computational intelligence; (ii) Cognitive computing; (iii) Denotational mathematics; (iv) Computational intelligence; and (v) Applications in computational intelligence and cognitive computing.

Section 1. Computational Intelligence

Intelligence science studies theories and models of the brain at all levels, and the relationship between the concrete physiological brain and the abstract soft mind. Intelligence science is a new frontier with the fertilization of biology, psychology, neuroscience, cognitive science, cognitive informatics, philosophy, information science, computer science, anthropology, and linguistics. A fundamental view developed in software and intelligence sciences is known as abstract intelligence, which provides a unified foundation for the studies of all forms and paradigms of intelligence such as natural, artificial, machinable, and computational intelligence. Abstract intelligence (αI) is an enquiry of both natural and artificial intelligence at the neural, cognitive, functional, and logical levels from the bottom up. In the narrow sense, αI is a human or a system ability that transforms information into behaviors. However, in the broad sense, αI is any human or system ability that autonomously transfers the forms of abstract information between data, information, knowledge, and behaviors in the brain or intelligent systems.

Computational intelligence (CoI) is an embodying form of abstract intelligence (αI) that implements intelligent mechanisms and behaviors by computational methodologies and software systems, such as expert systems, fuzzy systems, cognitive computers, cognitive robots, software agent systems, genetic/evolutionary systems, and autonomous learning systems. The theoretical foundations of computational intelligence root in cognitive informatics, software science, and denotational mathematics.
Chapter 1, *Convergence of Software Science and Computational Intelligence: A New Transdisciplinary Research Field*, by Yingxu Wang, presents two emerging fields of software science and computational intelligence as well as their relationship. Software Science is a discipline that studies the theoretical framework of software as instructive and behavioral information, which can be embodied and executed by generic computers in order to create expected system behaviors and machine intelligence. Intelligence science is a discipline that studies the mechanisms and theories of abstract intelligence and its paradigms such as natural, artificial, machinable, and computational intelligence. The convergence of software and intelligent sciences forms the transdisciplinary field of computational intelligence, which provides a coherent set of fundamental theories, contemporary denotational mathematics, and engineering applications. This editorial addresses the objectives of the *International Journal of Software Science and Computational Intelligence* (IJSSCI), and explores the domain of the emerging discipline. The historical evolvement of software and intelligence sciences and their theoretical foundations are elucidated. The coverage of this inaugural issue and recent advances in software and intelligence sciences are reviewed. This chapter demonstrates that the investigation into software and intelligence sciences will result in fundamental findings toward the development of future generation computing theories, methodologies, and technologies, as well as novel mathematical structures.

Chapter 2, *On Abstract Intelligence: Toward a Unifying Theory of Natural, Artificial, Machinable, and Computational Intelligence*, by Yingxu Wang, presents a novel theory known as *abstract intelligence* that is an enquiry of both natural and artificial intelligence at the reductive embodying levels of neural, cognitive, functional, and logical from the bottom up. This chapter describes the taxonomy and nature of intelligence. It analyzes roles of information in the evolution of human intelligence, and the needs for logical abstraction in modeling the brain and natural intelligence. A formal model of intelligence is developed known as the *Generic Abstract Intelligence Mode* (GAIM), which provides a foundation to explain the mechanisms of advanced natural intelligence such as thinking, learning, and inferences. A measurement framework of intelligent capability of humans and systems is comparatively studied in the forms of intelligent quotient, intelligent equivalence, and intelligent metrics. On the basis of the GAIM model and the abstract intelligence theories, the compatibility of natural and machine intelligence is revealed in order to investigate into a wide range of paradigms of abstract intelligence such as natural, artificial, machinable intelligence, and their engineering applications.

Chapter 3, *Hierarchies of Architectures of Collaborative Computational Intelligence*, by Witold Pedrycz presents computational intelligence as a wealth of methodologies and a plethora of algorithmic developments essential to the construction of intelligent systems. Being faced with inherently distributed data, which becomes evident, the paradigm of C1 calls for further enhancements along the line of designing systems that are hierarchical and collaborative in nature. This emerging direction could be referred to as collaborative Computational Intelligence (or C²I, for short). The pervasive phenomenon encountered in architectures of C²I is that collaboration is synonymous with knowledge sharing, knowledge reuse, and knowledge reconciliation. Knowledge itself comes in different ways: as some structural findings in data and usually formalized in the framework of information granules, locally available models, some action plans, classification schemes, and the like. In such distributed systems sharing data is not feasible given existing technical constraints, which are quite often exacerbated by non-technical requirements of privacy or security. In this study, the author elaborates on the design of information granules, which comes hand in hand with various clustering techniques and fuzzy clustering, in particular. Having stressed the role of information granules and Granular Computing, in general, the chapter demonstrates that such processing leads to the representatives of information granules and granular models in the
form of metastructures and metamodels. The author elaborates on pertinent optimization strategies and emphasizes a combinatorial character of the problems, which underlines a need for advanced techniques of evolutionary optimization. Collaboration leads to consensus and the quality of consensus achieved in this manner is quantified in terms of information granules of higher order (say type-2 fuzzy sets). The concept of information granulation emerging as a result of forming constructs of justifiable granularity becomes instrumental in the quantification of the achieved quality of collaboration.

Chapter 4, Challenges in the Design of Adoptive, Intelligent and Cognitive Systems, by Witold Kinsner presents cognitive machines that could act not only autonomously, but also in an increasingly intelligent and cognitive manner. Such cognitive machines ought to be aware of their environments which include not only other machines, but also human beings. Such machines ought to understand the meaning of information in more human-like ways by grounding knowledge in the physical world and in the machines’ own goals. The motivation for developing such machines range from self-evidenced practical reasons such as the expense of computer maintenance, to wearable computing (e.g., Mann, 2001) in healthcare, and gaining a better understanding of the cognitive capabilities of the human brain. To achieve such an ambitious goal requires solutions to many problems, ranging from human perception, attention, concept creation, cognition, consciousness, executive processes guided by emotions and value, and symbiotic conversational human-machine interactions. This chapter discusses some of the challenges emerging from this new design paradigm, including systemic problems, design issues, teaching the subjects to undergraduate students in electrical and computer engineering programs, research related to design.

Chapter 5, On Visual Semantic Algebra (VSA): A Denotational Mathematical Structure for Modeling and Manipulating Visual Objects and Patterns, by Yingxu Wang presents a new form of denotational mathematics known as Visual Semantic Algebra (VSA) for abstract visual object and architecture manipulations. A set of cognitive theories for pattern recognition is explored such as cognitive principles of visual perception and basic mechanisms of object and pattern recognition. The cognitive process of pattern recognition is rigorously modeled using VSA and Real-Time Process Algebra (RTPA), which reveals the fundamental mechanisms of natural pattern recognition by the brain. Case studies on VSA in pattern recognition are presented to demonstrate VSA’s expressive power for algebraic manipulations of visual objects. VSA can be applied not only in machinable visual and spatial reasoning, but also in computational intelligence as a powerful man-machine language for representing and manipulating visual objects and patterns. On the basis of VSA, computational intelligent systems such as robots and cognitive computers may process and inference visual and image objects rigorously and efficiently.

Section 2. Cognitive Computing

Computing systems and technologies can be classified into the categories of imperative, autonomic, and cognitive computing from the bottom up. The imperative computers are a passive system based on stored-program controlled mechanisms for data processing. The autonomic computers are goal-driven and self-decision-driven machines that do not rely on instructive and procedural information. Cognitive computers are more intelligent computers beyond the imperative and autonomic computers, which embody major natural intelligence behaviors of the brain such as thinking, inference, and learning. The increasing demand for non von Neumann computers for knowledge and intelligence processing in the high-tech industry and everyday lives require novel cognitive computers for providing autonomous computing power for various cognitive systems mimicking the natural intelligence of the brain.
Cognitive Computing (CC) is a novel paradigm of intelligent computing methodologies and systems based on Cognitive Informatics (CI), which implements computational intelligence by autonomous inferences and perceptions mimicking the mechanisms of the brain. CC is emerged and developed based on the transdisciplinary research in cognitive informatics, abstract intelligence, and Denotational Mathematics (DM). The latest advances in CI, CC, and DM enable a systematic solution for the future generation of intelligent computers known as Cognitive Computers (CogCs) that think, perceive, learn, and reason. A CogC is an intelligent computer for knowledge processing as that of a conventional von Neumann computer for data processing. CogCs are designed to embody machinable intelligence such as computational inferences, causal analyses, knowledge manipulations, machine learning, and autonomous problem solving.

Chapter 6, On Cognitive Computing, by Yingxu Wang presents cognitive computing as an emerging paradigm of intelligent computing methodologies and systems, which implements computational intelligence by autonomous inferences and perceptions mimicking the mechanisms of the brain. This chapter presents a survey on the theoretical framework and architectural techniques of cognitive computing beyond conventional imperative and autonomic computing technologies. Theoretical foundations of cognitive computing are elaborated from the aspects of cognitive informatics, neural informatics, and denotational mathematics. Conceptual models of cognitive computing are explored on the basis of the latest advances in abstract intelligence and computational intelligence. Applications of cognitive computing are described from the aspects of autonomous agent systems and cognitive search engines, which demonstrate how machine and computational intelligence may be generated and implemented by cognitive computing theories and technologies toward autonomous knowledge processing.

Chapter 7, On the System Algebra Foundations for Granular Computing, by Yingxu Wang, Lotfi A. Zadeh, and Yiyu Yao, presents a new mathematical means, granular algebra, for granular computing to computing system modeling and information processing. Although a rich set of work has advanced the understanding of granular computing in dealing with the “to be” and “to have” problems of systems, the “to do” aspect of system modeling and behavioral implementation has been relatively overlooked. On the basis of a recent development in denotational mathematics known as system algebra, this chapter presents a system metaphor of granules and explores the theoretical and mathematical foundations of granular computing. An abstract system model of granules is proposed in this chapter. Rigorous manipulations of granular systems in computing are modeled by system algebra. The properties of granular systems are analyzed, which helps to explain the magnitudes and complexities of granular systems. Formal representation of granular systems for computing is demonstrated by real-world case studies, where concrete granules and their algebraic operations are explained. A wide range of applications of the system algebra theory for granular computing may be found in cognitive informatics, computing, software engineering, system engineering, and computational intelligence.

Chapter 8, Semantic Matching, Propagation and Transformation for Composition in Component-Based Systems, by Eric Bouillet, Mark Feblowitz, Zhen Liu, Anand Ranganathan, and Anton Riabov, presents the composition of software applications from component parts in response to high-level goals. The authors target the problem of composition in flow-based information processing systems and demonstrate how application composition and component development can be facilitated by the use of semantically described application metadata. The semantic metadata describe both the data flowing through each application and the processing performed in the associated application code. In this chapter, the authors explore some of the key features of the semantic model, including the matching of outputs to input requirements, and the transformation and the propagation of semantic properties by components.
Chapter 9, *Adaptive Computation Paradigm in Knowledge Representation: Traditional and Emerging Applications*, by Marina L. Gavrilova, presents the constant demand for complex applications, the ever increasing complexity and size of software systems, and the inherently complicated nature of the information, which drive the needs for developing radically new approaches for information representation and processing. This drive is leading to creation of new and exciting interdisciplinary fields that investigate convergence of software science and intelligence science, as well as computational sciences and their applications. This survey article presents the new paradigm of the algorithmic models of intelligence, based on the adaptive hierarchical model of computation, and presents the algorithms and applications utilizing this paradigm in data-intensive, collaborative environment. Examples from the various areas include references to adaptive paradigm in biometric technologies, evolutionary computing, swarm intelligence, robotics, networks, e-learning, knowledge representation, and Information System design. Special topics related to adaptive models design and geometric computing are also included in the survey.

Chapter 10, *Protoforms of Linguistic Database Summaries as a Human Consistent Tool for Using Natural Language in Data Mining*, by Janusz Kacprzyk and Slawomir Zadrozny, presents a linguistic database that summarizes a personnel database in which most employees are young and well paid and their extensions as a very general tool for a human consistent summarization of large data sets. The authors advocate the use of the concept of a protoform as a general form of a linguistic data summary. Then, they present an extension of our interactive approach to fuzzy linguistic summaries, based on fuzzy logic and fuzzy database queries with linguistic quantifiers. The authors show how fuzzy queries are related to linguistic summaries, and that one can introduce a hierarchy of protoforms, or abstract summaries in the sense of latest ideas meant mainly for increasing deduction capabilities of search engines. The chapter shows an implementation for the summarization of Web server logs.

Chapter 11, *Measuring Textual Context Based on Cognitive Principles*, by Ning Fang, Xiangfeng Luo, and Weimin Xu, presents a measurement of the complexity of textual context with subjective cognitive degree of information. Based on minimization of Boolean complexity in human concept learning, the complexity and the difficulty of textual context are defined in order to mimic human’s reading experience. Based on maximal relevance principle, the information and cognitive degree of textual context are defined in order to mimic human’s cognitive sense. Experiments verify that more contexts are added, more easily the text is understood by a machine, which is consistent with the linguistic viewpoint that context can help to understand a text; furthermore, experiments verify that the author-given sentence sequence includes the less complexity and the more information than other sentence combinations, that is to say, author-given sentence sequence is more easily understood by a machine. So the principles of simplicity and maximal relevance actually exist in text writing process, which is consistent with the cognitive science viewpoint. Therefore, the measuring methods are validated from the linguistic and cognitive perspectives, and it could provide a theoretical foundation for machine-based text understanding.

Chapter 12, *A Lexical Knowledge Representation Model for Natural Language Understanding*, by Ping Chen, Wei Ding and Chengmin Ding, presents knowledge representation as an essential technology for semantics modeling and intelligent information processing. For decades, researchers have proposed many knowledge representation techniques. However, it is a daunting problem how to capture deep semantic information effectively and support the construction of a large-scale knowledge base efficiently. This chapter describes a new knowledge representation model, SenseNet, which provides semantic support for commonsense reasoning and natural language processing. SenseNet is formalized with a Hidden Markov Model. An inference algorithm is proposed to simulate human-like natural lan-
guage understanding procedure. A new measurement, confidence, is introduced to facilitate the natural language understanding. The authors present a detailed case study of applying SenseNet to retrieving compensation information from company proxy filings.

Chapter 13, **A Dualism Based Semantics Formalization Mechanism for Model Driven Engineering**, by Yucong Duan, presents a thorough discussion of semantics formalization related issues in model driven engineering (MDE). Motivated for the purpose of software implementation, and attempts to overcome the shortcomings of incompleteness and context-sensitivity in the existing models, the chapter proposes to study formalization of semantics from a cognitive background. Issues under study cover the broad scope of overlap vs. incomplete vs. complete, closed world assumption (CWA) vs. open world assumption (OWA), Y(Yes)/N(No) vs. T(True)/F(False), subjective (SUBJ) vs. objective (OBJ), static vs. dynamic, unconsciousness vs. conscious, human vs. machine aspects, et cetera. A semantics formalization approach called EID-SCE (Existence Identification Dualism-Semantics Cosmos Explosion) is designed to meet both the theoretical investigation and implementation of the proposed formalization goals. EID-SCE supports the measure/evaluation in a {complete, no overlap} manner whether a given concept or feature is an improvement. Some elementary cases are also shown to demonstrate the feasibility of EID-SCE.

## Section 3. Software Science

Software as instructive behavioral information has been recognized as an entire range of widely and frequently used objects and phenomena in human knowledge. Software science is a theoretical inquiry of software and its constraints on the basis of empirical studies on engineering methodologies and techniques for software development and software engineering organization. In the history of science and engineering, a matured discipline always gave birth to new disciplines. For instance, theoretical physics was emerged from general and applied physics, and theoretical computing was emerged from computer engineering. So does software science that emerges from and grows in the fields of software, computer, information, knowledge, and system engineering.

*Software Science* (SS) is a discipline of enquiries that studies the theoretical framework of software as instructive and behavioral information, which can be embodied and executed by generic computers in order to create expected system behaviors and machine intelligence. The discipline of software science studies the common objects in the abstract world such as software, information, data, concepts, knowledge, instructions, executable behaviors, and their processing by natural and artificial intelligence. From this view, software science is theoretical software engineering; while software engineering is applied software science in order to efficiently, economically, and reliably develop large-scale software systems. The phenomena that almost all the fundamental problems, which could not be solved in the last four decades in software engineering, simply stemmed from the lack of coherent theories in the form of software science. The vast cumulated empirical knowledge and industrial practice in software engineering have made this possible to enable the emergence of software science.

Chapter 14, **Exploring the Cognitive Foundations of Software Engineering**, by Yingxu Wang and Shushma Patel, presents that software is a unique abstract artifact that does not obey any known physical laws. For software engineering to become a matured engineering discipline like others, it must establish its own theoretical framework and laws, which are perceived to be mainly relied on cognitive informatics and denotational mathematics, supplementing to computing science, information science, and formal linguistics. This chapter analyzes the basic properties of software and seeks the cognitive
informatics foundations of software engineering. The nature of software is characterized by its informatics, behavioral, mathematical, and cognitive properties. The cognitive informatics foundations of software engineering are explored on the basis of the informatics laws of software and software engineering psychology. A set of fundamental cognitive constraints of software engineering, such as intangibility, complexity, indeterminacy, diversity, polymorphism, inexpressiveness, inexplicit embodiment, and un-quantifiable quality measures, is identified. The conservative productivity of software is revealed based on the constraints of human cognitive capacity.

Chapter 15, **Positive and Negative Innovations in Software Engineering**, by Capers Jones, presents the software engineering field as a fountain of innovation. Ideas and inventions from the software domain have literally changed the world as we know it. For software development, we have a few proven innovations. The way software is built remains surprisingly primitive. Even in 2008, major software applications are cancelled, overrun their budgets and schedules, and often have hazardously bad quality levels when released. There have been many attempts to improve software development, but progress has resembled a drunkard’s walk. Some attempts have been beneficial, but others have been either ineffective or harmful. This article puts forth the hypothesis that the main reason for the shortage of positive innovation in software development methods is due to a lack of understanding of the underlying problems of the software development domain. A corollary hypothesis is that lack of understanding of the problems is due to inadequate measurement of quality, productivity, costs, and the factors that affect project outcomes.

Chapter 16, **On the Cognitive Complexity of Software and its Quantification and Formal Measurement**, by Yingxu Wang, presents the quantification and measurement of functional complexity of software that were recognized as a persistent problem in software engineering. Measurement models of software complexities have been studied in two facets in computing and software engineering, where the former is machine-oriented in the small; while the latter is human-oriented in the large. The cognitive complexity of software presented in this chapter is a new measurement for cross-platform analysis of complexities, functional sizes, and cognition efforts of software code and specifications in the phases of design, implementation, and maintenance in software engineering. This chapter reveals that the cognitive complexity of software is a product of its architectural and operational complexities on the basis of deductive semantics. A set of ten Basic Control Structures (BCS’s) are elicited from software architectural and behavioral modeling and specifications. The cognitive weights of the BCS’s are derived and calibrated via a series of psychological experiments. Based on this work, the cognitive complexity of software systems can be rigorously and accurately measured and analyzed. Comparative case studies demonstrate that the cognitive complexity is highly distinguishable for software functional complexity and size measurement in software engineering.

Chapter 17, **Machine Learning and Value-Based Software Engineering**, by Du Zhang, presents a view of software engineering research and practice that are primarily conducted in a value-neutral setting where each artifact in software development such as requirement, use case, test case, and defect, is treated as equally important during a software system development process. There are a number of shortcomings of such value-neutral software engineering. Value-based software engineering is to integrate value considerations into the full range of existing and emerging software engineering principles and practices. Machine learning has been playing an increasingly important role in helping develop and maintain large and complex software systems. However, machine learning applications to software engineering have been largely confined to the value-neutral software engineering setting. In this chapter, the general message to be conveyed is to apply machine learning methods and algorithms to value-based
software engineering. The training data or the background knowledge or domain theory or heuristics or bias used by machine learning methods in generating target models or functions should be aligned with stakeholders’ value propositions. An initial research agenda is proposed for machine learning in value-based software engineering.

Chapter 18, *The Formal Design Model of a Telephone Switching System (TSS)*, by Yingxu Wang presents a typical real-time system, the Telephone Switching System (TSS), as a highly complicated system in design and implementation. This chapter presents the formal design, specification, and modeling of the TSS system using a denotational mathematics known as Real-Time Process Algebra (RTPA). The conceptual model of the TSS system is introduced as the initial requirements for the system. Then, the architectural model of the TSS system is created using the RTPA architectural modeling methodologies and refined by a set of Unified Data Models (UDMs). The static behaviors of the TSS system are specified and refined by a set of Unified Process Models (UPMs) such as call processing and support processes. The dynamic behaviors of the TSS system are specified and refined by process priority allocation, process deployment, and process dispatching models. Based on the formal design models of the TSS system, code can be automatically generated using the RTPA Code Generator (RTPA-CG), or be seamlessly transformed into programs by programmers. The formal model of TSS may not only serve as a formal design paradigm of real-time software systems, but also a test bench of the expressive power and modeling capability of exiting formal methods in software engineering.

Chapter 19, *The Formal Design Model of a Lift Dispatching System (LDS)*, by Yingxu Wang, Cyprian F. Ngolah, Hadi Ahmadi, Philip Sheu, and Shi Ying, presents a Lift Dispatching System (LDS) that is highly complicated in design and implementation. This chapter presents the formal design, specification, and modeling of the LDS system using a denotational mathematics known as Real-Time Process Algebra (RTPA). The conceptual model of the LDS system is introduced as the initial requirements for the system. The architectural model of the LDS system is created using RTPA architectural modeling methodologies and refined by a set of Unified Data Models (UDMs). The static behaviors of the LDS system are specified and refined by a set of Unified Process Models (UPMs) for the lift dispatching and serving processes. The dynamic behaviors of the LDS system are specified and refined by process priority allocation and process deployment models. Based on the formal design models of the LDS system, code can be automatically generated using the RTPA Code Generator (RTPA-CG), or be seamlessly transferred into programs by programmers. The formal models of LDS may not only serve as a formal design paradigm of real-time software systems, but also a test bench of the expressive power and modeling capability of exiting formal methods in software engineering.

Chapter 20, *A Theory of Program Comprehension: Joining Vision Science and Program Comprehension*, by Yann-Gaël Guéhéneuc, presents a program comprehension technology in vision science and software engineering. It is identified that these two domains of research have been so far rather disjoint. Indeed, several cognitive theories have been proposed to explain program comprehension. These theories explain the processes taking place in the software engineers’ minds when they understand programs. They explain how software engineers *process* available information to perform their tasks but not how software engineers *acquire* this information. Vision science provides explanations on the processes used by people to acquire visual information from their environment. Joining vision science and program comprehension provides a more comprehensive theoretical framework to explain facts on program comprehension, to predict new facts, and to frame experiments. The author joins theories in vision science and in program comprehension; the resulting theory is consistent with facts on program
comprehension and helps in predicting new facts, in devising experiments, and in putting certain program comprehension concepts in perspective.

Chapter 21, Requirements Elicitation by Defect Elimination: An Indian Logic Perspective, by G.S. Mahalakshmi and T.V. Geetha, presents an Indian-logic based approach for automatic generation of software requirements from a domain-specific ontology. The structure of domain ontology is adapted from Indian logic. Domain concepts and its member qualities, relations between concepts, relation-qualities etc. contribute to the Indian logic ontology. This is different from the western-logic based ontology, where only concepts and relations between concepts are classified into an ontological framework. The interactive approach proposed in this chapter parses the problem statement, and section or sub-section of the domain ontology, which matches the problem statement, is identified. The software generates questions to the stakeholders based on the concepts of the identified sections of the domain ontology. The answer to every question is collected and analysed for presence of flaws or inconsistencies. Subsequent questions are recursively generated to repair the flaw in the previous answer. The flaws or missing (or inconsistent) information are actually identified by mapping the answers (or values) with that of the concepts of the domain ontology. These answers are populated into requirements ontology, which contains problem specific information coupled with the interests of the stakeholder. The information gathered in this fashion is stored in a database, which is later segregated into functional and non-functional requirements. These requirements are classified, validated, and prioritized based on combined approach of AHP and stakeholders’ defined priority. Conflict between requirements is resolved by the application of cosine correlation measure.

Chapter 22, Measurement of Cognitive Functional Sizes of Software, by Sanjay Misra, presents cognitive functional size measure based on Wang’s theory. Since traditional measurement theory has problem in defining empirical observations on software entities in terms of their measured quantities, Morasca has tried to solve this problem by proposing Weak Measurement theory. Further, in calculating complexity of software, the emphasis is mostly given to the computational complexity, algorithm complexity, functional complexity, which basically estimates the time, efforts, computability and efficiency. On the other hand, understandability and compressibility of the software, which involves the human interaction, are neglected in existing complexity measurement approaches. Recently, Wang has tried to fill this gap and developed the cognitive complexity to calculate the architectural and operational complexity of software. In this chapter, an attempt has been made to evaluate cognitive complexity against the principle of measurement theory/weak measurement theory. The author finds that the approach for measuring cognitive complexity is more realistic and practical in comparison to the existing approaches. Cognitive complexity satisfies most of the parameters required from measurement theory perspective. The chapter also investigates the applicability of Extensive Structure for deciding on the type of scale for cognitive complexity. It is found that the cognitive complexity is on weak ratio scale.

Chapter 23, Motivational Gratification: An Integrated Work Motivation Model with Information System Design Perspective, by Sugumar Mariappanadar, presents a view that researchers in the field of Information System (IS) endorse the view that there is always a discrepancy between the expressions of client’s automation requirements and IS designers understanding of such requirement because of difference in the field of expertise. In this chapter, an attempt is taken to develop a motivational gratification model (MGM) from the cognitive informatics perspective for the automation of employee motivation measurement, instead of developing a motivation theory from a management perspective and expecting the IS designers to develop a system based on the understanding of the theory that is alien to his/her field of expertise. Motivational gratification is an integrated work motivation model that theoretically
explains how employees self-regulate their effort intensity for “production” or “reduction” of motivational force towards future high performance, and it is developed using taxonomies of system approach from psychology and management. The practical implications of MGM in management and IS analysis and design are discussed.

**Section 4. Applications of Computational Intelligence and Cognitive Computing**

A series of fundamental breakthroughs have been recognized and a wide range of applications has been developed in software science, abstract intelligence, cognitive computing, and computational intelligence in the last decade. Because software science and computational intelligence provide a common and general platform for the next generation of cognitive computing, some expected innovations in these fields will emerge such as cognitive computers, cognitive knowledge representation technologies, semantic searching engines, cognitive learning engines, cognitive Internet, cognitive robots, and autonomous inference machines for complex and long-series of inferences, problem solving, and decision making beyond traditional logic- and rule-based technologies.

Chapter 24, **Supporting CSCW and CSCL with Intelligent Social Grouping Services**, by Jeffrey J.P. Tsai, Jia Zhang, Jeff J.S. Huang, and Stephen J.H. Yang, presents an intelligent social grouping service for identifying right participants to support CSCW and CSCL. The authors construct a three-layer hierarchical social network, in which they identify two important relationship ties – a knowledge relationship tie and a social relationship tie. They use these relationship ties as metric to measure the collaboration strength between pairs of participants in a social network. The stronger the knowledge relationship tie, the more knowledgeable the participants; the stronger the social relationship tie, the more likely the participants are willing to share their knowledge. By analyzing and calculating these relationship ties among peers using our computational models, the authors present a systematic way to discover collaboration peers according to configurable and customizable requirements. Experiences of social grouping services for identifying communities of practice through peer-to-peer search are also reported.

Chapter 25, **An Enhanced Petri Net Model to Verify and Validate a Neural-Symbolic Hybrid System**, by Ricardo R. Jorge, Gerardo R. Salgado, and Vianey G.C. Sánchez, presents that as the Neural-Symbolic Hybrid Systems (NSHS) gain acceptance, it increases the necessity to guarantee the automatic validation and verification of the knowledge contained in them. In the past, such processes were made manually. In this chapter, an enhanced Petri net model is presented to the detection and elimination of structural anomalies in the knowledge base of the NSHS. In addition, a reachability model is proposed to evaluate the obtained results of the system versus the expected results by the user. The validation and verification method is divided in two stages: 1) it consists of three phases: rule normalization, rule modeling and rule verification. 2) It consists of three phases: rule modeling, dynamic modeling and evaluation of results. Such method is useful to ensure that the results of a NSHS are correct. Examples are presented to demonstrate the effectiveness of the results obtained with the method.

Chapter 26, **System Uncertainty Based Data-Driven Knowledge Acquisition**, by Jun Zhao and Guoyin Wang, presents that in the three-layered framework for knowledge discovery, it is necessary for technique layer to develop some data-driven algorithms, whose knowledge acquiring process is characterized by and hence advantageous for the unneccessity of prior domain knowledge or external information. System uncertainty is able to conduct data-driven knowledge acquiring process. It is crucial for such a knowledge acquiring framework to measure system uncertainty reasonably and precisely. Herein, in order to find a suitable measuring method, various uncertainty measures based on rough set theory are
comprehensively studied: their algebraic characteristics and quantitative relations are disclosed; their performances are compared through a series of experimental tests; consequently, the optimal measure is determined. Then, a new data-driven knowledge acquiring algorithm is developed based on the optimal uncertainty measure and the Skowron’s algorithm for mining propositional default decision rules. Results of simulation experiments illustrate that the proposed algorithm obviously outperforms some other congeneric algorithms.

Chapter 27, Hierarchical Function Approximation with a Neural Network Model, by Luis F. de Mingo, Nuria Gómez, Fernando Arroyo, and Juan Castellanos, presents a model based on neural networks that permits to build a conceptual hierarchy in order to approximate functions over a given interval. A new kind of artificial neural networks using bio-inspired axo-axonic connections. These connections are based on the idea that the signal weight between two neurons is computed by the output of other neuron. Such model can generate polynomial expressions with lineal activation functions and the degree \( n \) of the output depends on the number \( n - 2 \) of hidden layers. This network can approximate any pattern set with a polynomial equation, similar to Taylor series approximation. Results concerning function approximation using artificial neural networks based on multi-layer perceptrons with axo-axonic connections are shown. This neural system classifies an input pattern as an element belonging to a category or subcategory that the system has, until an exhaustive classification is obtained, that is, a hierarchical neural model. The proposed neural system is not a hierarchy of neural networks; this model establishes relationships among all the different neural networks in order to propagate the neural activation when an external stimulus is presented to the system. Each neural network is in charge of the input pattern recognition to any prototyped class or category, and also in charge of transmitting the activation to other neural networks to be able to continue with the approximation. Therefore, the communication of the neural activation in the system depends on the output of each one of the neural networks, so as the functional links established among the different networks to represent the underlying conceptual hierarchy.

Chapter 28, Application of Artificial Neural Computation in Topex Waveform Data: A Case Study on Water Ratio Regression, by Bo Zhang, Franklin W. Schwartz, and Daoqin Tong, presents that using the TOPEX radar altimeter for land cover studies has been of great interest due to the TOPEX near global coverage and its consistent availability of waveform data for about one and a half decades from 1992 to 2005. However, the complexity of the TOPEX Sensor Data Records (SDRs) makes the recognition of the radar echoes particularly difficult. In this chapter, artificial neural computation as one of the most powerful algorithms in pattern recognition is investigated for water ratio assessment over Lake of the Woods area using TOPEX reflected radar signals. Results demonstrate that neural networks have the capability in identifying water proportion from the TOPEX radar information, controlling the predicted errors in a reasonable range.

Chapter 29, A Generic Framework for Feature Representations in Image Categorization Tasks, by Adam Csapo, Barna Resko, Morten Lind, and Peter Baranyi, presents computerized modeling of cognitive visual information. The research field is interesting not only from a biological perspective, but also from an engineering point of view when systems are developed that aim to achieve similar goals as biological cognitive systems. This chapter introduces a general framework for the extraction and systematic storage of low-level visual features. The applicability of the framework is investigated in both unstructured and highly structured environments. In a first experiment, a linear categorization algorithm originally developed for the classification of text documents is used to classify natural images taken from the Caltech 101 database. In a second experiment, the framework is used to provide an automatically guided vehicle with obstacle detection and auto-positioning functionalities in highly structured
environments. Results demonstrate that the model is highly applicable in structured environments, and also shows promising results in certain cases when used in unstructured environments.

This book is intended to the readership of researchers, engineers, graduate students, senior-level undergraduate students, and instructors as an informative reference book in the emerging fields of software science, cognitive intelligence, and computational intelligence. The editor expects that readers of Software and Intelligent Sciences: New Transdisciplinary Findings will benefit from the 29 selected chapters of this book, which represents the latest advances in research in software science and computational intelligence and their engineering applications.

Yingxu Wang
University of Calgary, Canada