Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

WHAT CAN BE DONE TO AUTOMATE CONCEPTUAL SIMULATION MODELING?

Ming Zhou, Qun Zhang

Center for Systems Modeling and Simulation
Indiana State University
Terre Haute, Indiana, U.S.A.

Zhimin Chen

College of Management
Shenzhen University
Shenzhen, P.R. China

ABSTRACT

Conceptual modeling is a critical step that directly affects the quality and efficiency of simulation projects. However current technology can hardly support the process and most practice demonstrated an *ad hoc* and inefficient approach. Automation can help improve the efficiency and effectiveness of conceptual simulation modeling. However, there are a number of issues must be addressed, including the formalization of model concepts, representation of modeling knowledge, and interaction between user and computer system. This paper presents a discussion of these issues based on the research by the authors, and propose suggestions for the design and development of a robust computerized modeling environment that aims to improve conceptual simulation modeling process.

1 INTRODUCTION

Conceptual modeling (CM) has been recognized as a critical step that directly affects the quality and efficiency of simulation projects. Good CM practice significantly reduce communication barriers, shorten project time, and improve the quality of simulation. Although the development of simulation technology has made significant progress in automating other steps of simulation, CM remains a task that is almost completely manually performed. There are several reasons for this lack of development. First CM is the least understood and poorly formalized step and has been given little attention in practice and by previous research (Zhou et al. 2004, Robinson 2004). Comparing with other steps in simulation (e.g., encoding and output analysis), we are not very clear about the tasks (and their nature) involved in CM. This lack of understanding and facilitation tools has aggravated the tendency among practitioners that try to use implementation tools (i.e., a specific simulation language/package) to perform conceptual modeling, which often resulted in models that were congenitally deficient and difficult for modification or adaptation. This has become worse as modern systems evolve larger and more complex. We believe that just like other steps in simulation, CM can be improved by automation, and a more structured or formalized approach is needed to achieve this improvement. First we need to identify and define the tasks involved in conceptual simulation modeling. Further we need to study the nature of these tasks (activities) to identify their cognitive characteristics, and use these characteristics to formulate "best" strategies to incorporate automation into CM process, or develop innovative automation scheme that may even revolutionize the process. Once we have better idea regarding what should or can be automated, we then develop requirement specifications for developing automated systems. These specifications are the "guidelines" that, perceived by the authors, need to be considered to improve conceptual simulation modeling through a more "automated" approach. This paper briefly describes the approach that is used in our research.

2 COGNITIVE TASKS INVOLVED IN CONCEPTUAL SIMULATION MODELING

In this section we review Norman's Two Gulf model (Norman 1986) to identify the cognitive characteristics of the tasks involved in CM processes. Norman's model provides a more abstract framework in which modeling tasks can be classified into cognitive processes that help us identify most promising opportunities for automation. Figure 1 top part showed original Norman's model that identifies main cognitive processes involved in a task-performing process. There are two gulfs: an "execution gulf", containing three processes, Intent, Plan, and Execute; and an "evaluation gulf" containing Perceive, Interpret, and Evaluate. In the lower part of Figure 1, we showed a mapping of a modeling task "Selection" (e.g., selecting a system type) to the cognitive processes in Norman's model. Note that "System" represents a computerized environment, while "Modeler" is a human actor (e.g., a domain expert) that interacts with the system to construct a conceptual model. "Setup" (i.e., anticipate the type of system to be selected) corresponds to "Intent"; "Prepare" (prepare inquires for the selection) corresponds to "Plan", and so on.

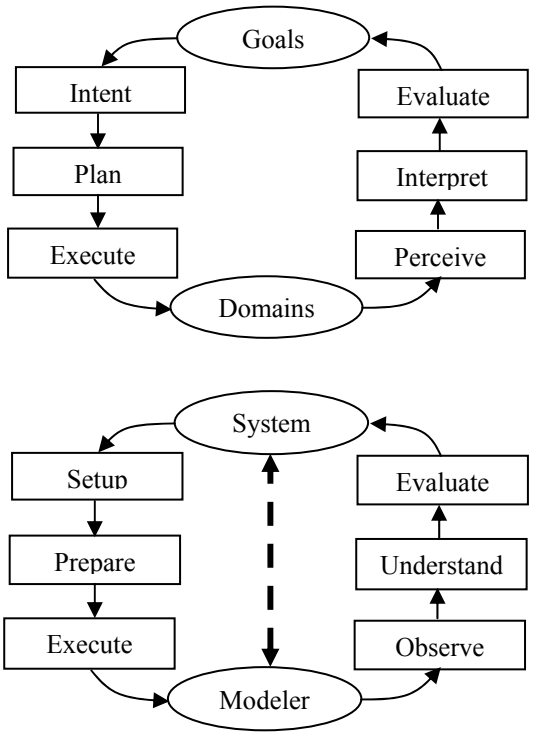

Figure 1: Mapping of Modeling Task to Norman's Cognitive Processes

The heavy dashed line in the lower part represents man-machine interaction during the course of conceptual simulation modeling. The analysis via Norman's model helped us identify the cognitive tasks that provide opportunities for automation. In the context of simulation, we can improve the efficiency of conceptual modeling by automating repetitive, repeated and tedious tasks, or assisting users in making complex decisions.

Table 1: Opportunities for Automation in "Execution Gulf"

1 EXECUTION GULF	**2 AUTOMATION FOR GOAL INSTANTIATION**
Identify and set up a goal	Accurately represent the goal: anticipate user's desire and accurately represent the desire
Translate the goal into a sequence of tasks	Use appropriately represented knowledge to plan the sequence of actions
Execute the tasks to achieve the goal	Execute the tasks by computer or with computer assistance

Table 1 showed an example that suggests possible automation corresponding to the steps in the execution gulf of Norman's model (Lewis 1998), e.g., task execution, decision-making, and evaluation (e.g., computing/collecting feedback).

3 CONSIDERATIONS ON SYSTEM REQUIREMENTS

To automate the modeling tasks analyzed through Norman's reference model and increase the degree of automation in CM, we need to develop a computer-aided system that would function as a modeling environment to facilitate the process. Following considerations identify the aspects of system requirements that need to be addressed to facilitate desired automation.

1. Knowledge acquisition and representation: we need to identify and formalize the knowledge and information needed to construct conceptual models for different types of simulation applications, and more importantly we need to develop robust and reusable structures to represent the knowledge and information for efficient and *maybe* automated modeling. Table 2 showed some schemes that can be used to represent different types of modeling knowledge.

Table 2: Knowledge Representation

KNOWLEDGE TYPE	**REPRESENTATION**
Factual knowledge	Formalized sets, object/classes
Procedural knowledge	Rules, algorithms
Interactions	Structured inquires

2. Knowledge and information processing: we need two types of mechanisms in this regard. First since a knowledge-based approach is emphasized, we need an efficient mechanism that automatically interacts with a knowledgebase (KB) to process user input and pre-stored knowledge to derive new information or make a decision. This mechanism is usually an inference engine integrated with a KB in an expert system. In addition, we need robust and efficient mechanisms that help us formalize and implement the cognitive tasks involved to possibly automate these tasks. Examples of such mechanisms include user-defined subroutines (algorithms/procedures) that perform automatic search or retrieving, generate entity

flow trees (logic models) and/or specify model elements (Zhou et al. 2004).

3. Principles and standards of software engineering: to computerize (i.e., to automate) CM process, we need to develop software systems including task-specific application programs and "platforms" that accommodate or integrate the applications. Software development is a critical effort in automating CM. We need a robust and standardized approach to guide and facilitate the developmental phases including artifacts, intermediate representations, requirement and design analysis, development and implementation processes. Unified development process and UML (Larman 1998) have been recognized to be helpful in facilitating the transition from conceptualization to implementation, and developing robust and reusable software components and systems.

4. Development of centralized or decentralized modeling environment: we need to integrate various modeling functions and develop a user-friendly environment to facilitate conceptual simulation modeling. There are two basic approaches. We can follow a *traditional* approach to develop a centralized environment, i.e., design and provide a computer model of the whole CM process so that user can directly interact with it to perform required modeling tasks. Alternatively we can create a decentralized environment that emphasizes the use of *specialized agents* and constructs conceptual models based on the collaboration of these intelligent agents. While a centralized approach (i.e., Human-Computer Interaction or HCI approach) may be simpler in understanding and controlling, Agent-based approach (Human Agent Interaction or HAI approach), on the other hand, offers several advantages, such as being specialized, *flexible*, allowing for incremental development and easy maintenance (Lewis 1998). Given the complexity of CM problem and the distributed nature of modeling knowledge, a decentralized approach is considered more effective in facilitating the automation of CM process.

4 SPECIFICATIONS OF THE REQUIREMENTS

4.1 Knowledge Representation (KR)

In the context of simulation modeling, knowledge representations are used to facilitate: (a) an across-domain communication between modeler and user; (b) the reuse of conceptual models for similar applications; (c) intelligent information processing such as automated reasoning or processing by computers; and (d) the transformation of model requirement from conceptualization to implementation. To improve conceptual simulation modeling for target applications (manufacturing, logistics and distribution, and emergency room systems), we classified the related knowledge into three categories: model elements, flow templates, and specialization algorithms, and grouped them into a collection of reusable patterns or *modeling frameworks*. To facilitate software realization, we used an object-oriented paradigm to formalize model elements due to its inherent support for abstraction-centric, reusable, and adaptable design. Representational models for composite structures, such flow templates (reusable logical configurations that define entity flows for target application types), were also developed to address the needs of communication and transformation of CM requirements.

Model elements.
To formalize the specification of model elements and develop robust representations, we proposed a set of notation. A conceptual simulation model $\boldsymbol{CSM} = \langle \boldsymbol{S_1}, \ldots \ldots, \boldsymbol{S_m} \rangle$, i.e., ***CSM*** is a collection of disjoint sets $\boldsymbol{S_1}, \ldots \ldots, \boldsymbol{S_m}$, where each $\boldsymbol{S_i}$, $1 \leq i \leq m$, is a set of partial specification of ***CSM***. Part of the definitions ($\boldsymbol{S_1}$) are displayed as an example:

> $\boldsymbol{S_1}$ = Set of simulation objects; $\exists\ \boldsymbol{A_1} \leftrightarrow \boldsymbol{S_1}$, $\boldsymbol{A_1}$ = A set of attributes associated with $\boldsymbol{S_1}$. Symbol "↔" is used as an "association" operator here. $\boldsymbol{S_1} = \langle \boldsymbol{E}, \boldsymbol{R}, \boldsymbol{W} \rangle$, where $\boldsymbol{E}$ = Set of entities, $\boldsymbol{R}$ = Set of resources, and $\boldsymbol{W}$ = Set of workstations; and $\exists\ \boldsymbol{A_E} \leftrightarrow \boldsymbol{E}$, $\exists\ \boldsymbol{A_R} \leftrightarrow \boldsymbol{R}$, and $\exists\ \boldsymbol{A_W} \leftrightarrow \boldsymbol{W}$, where $\boldsymbol{A_E}$, $\boldsymbol{A_R}$ and $\boldsymbol{A_W}$ are sets of attributes. For instance, $\boldsymbol{A_E} = \{A_{E1}, \ldots, A_{En}\}$; where A_{E1} = Entity Name; A_{E2} = Entity Type; A_{E3} = Arrival Pattern = Inter-arrival time distribution; A_{E4} = Process Plan (Routing) = An entity-type-dependent sequence of processing operations ($O_1, \ldots, O_k$).

This notation helps formalize concepts and develop robust representations for model elements, e.g., using a class of objects or frames to represent the concepts of entities, resources and workstations defined in $\boldsymbol{S_1}$. We used an object-oriented (OO) formalism, i.e., a collection of taxonomic structures to represent the model elements.

To capture and characterize the patterns of logic flows of target application types (e.g., inbound or outbound processes at a distribution center or patient flow through an emergency department), we used a composite representation scheme: flow templates. These templates can be reused to specify entity flows (logic models) for similar applications. Such templates can be defined in two views independent of implementation. In a *simulation modeling view*, a flow template is a process model that contains a sequence of logic activities performed at a number of locations using a set of dedicated or shared resources. There is a defined logic that controls the flow of entities through the

activities to achieve the requirement of simulation. The activities and the control logic together form a configuration pattern that is abstracted properly so that it is independent of application and implementation specifics while remains robust in representing the key structural and behavioral characteristics and can be reused for similar applications. From a *knowledge representation view*, a flow template is a composite class that contains a set of inter-related object classes and control classes. The properties and behavior of each class and collaborations between the classes are defined to reflect the roles or functions performed by the flow template and to meet modeling requirements (Zhou et al. 2005a). This view is particularly useful in translating the requirement from conceptual level to implementation level.

Algorithms for developing/specializing logical models. The procedures used by experienced or expert modelers to construct conceptual simulation models (i.e., transforming a mentally existing model into a *formalized* conceptual model) have been given little attention by previous research. It is possible to significantly increase both the efficiency and effectiveness of modeling if we can capture and represent this type of *empirical knowledge* through a structured and knowledge-based approach that can be implemented through a computer, via necessary interactions with a human modeler. Descriptions about such reusable procedures, formalized as algorithms for generating entity flow trees, specializing a model configuration pattern (e.g., subclassing in a modeling framework), and the analysis of constraints involved, are presented through our previous studies. (Zhou et al. 2004, 2005b)

Classes of modeling frameworks that integrate model elements, flow templates and specialization algorithms. It would be more efficient to construct a conceptual model using a predefined framework, which is specified as a collection of collaborating classes that provide required service for modeling a given domain. We can customize a framework to a particular application by specializing the framework and composing the instances of the framework class. Therefore modeling frameworks represent object-oriented reuse of predefined knowledge and information. These frameworks can be created through object modeling and design patterns that link the classes of objects through domain-dependent collaborations (Nicola et al. 2002). Such frameworks structure the design of applications by providing a set of predefined abstractions, given by collaborating classes in the framework, to provide an architectural guidance for system design and development. Figure 2 shows the composition of a modeling framework, which contains object-oriented knowledge representations of collaborating classes that are integrated into a reusable framework to specify the model requirements for a certain type of applications.

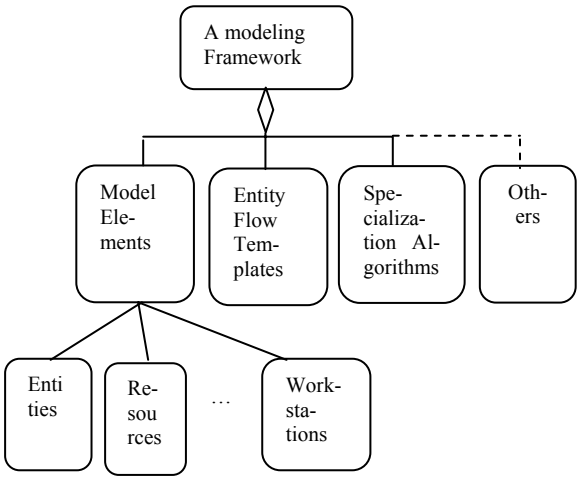

Figure 2: Composition of a Modeling Framework

4.2 Design and development of a modeling environment

As mentioned earlier, there are two basic approaches in creating a modeling environment. A centralized approach focuses on the creation of an *overall model* of CM process and assumes direct interactions between user and the model. A decentralized approach, however, emphasizes the use of collaborated task-specific agents, which act as an intermediary between the user and the computer. There is a strong motivation for pursuing a decentralized approach. Generally conceptual modeling problems in simulation are too complex for a single knowledge-based system to be efficient, effective and robust. The modeling knowledge or expertise are naturally distributed across different domains. In many cases, modeling tasks may need to be performed asynchronously for efficiency. Interoperations between multiple legacy systems (e.g., database) can also pose serious concerns when the overall system becomes large and complex. There is obviously a need to reduce system complexity (in design, development and maintenance) through modularity, flexibility and reusability at knowledge level.

In a decentralized environment, main modeling functions are delegated to knowledge-based agents (KBA) that can assist decision-making in conceptual modeling process and automate certain tasks to improve efficiency. We conceive these KBAs as static collaborative task-specific agents characterized by autonomy and cooperation. Although each agent performs a different set of functions, they have a common architecture containing four components: (1) an interface that defines the perception of the agents, i.e., its interaction with environment and other agents (what and whom to interact); (2) a knowledgebase that contains know-how to perform a set of modeling tasks; and (3) an inference engine to process the knowledge and inputs to accomplish the reasoning step and derive a action

plan; and (4) an action mechanism that execute a sequence of actions to accomplish the required tasks. Grouping problem-solving knowledge into "functional" chunks and encapsulating them through specialized agents can reduce complexity of system design/development, and improve system maintenance and support incremental development (Kim et al. 2004). These KBAs are independent proactive software brokers/modules that interact with one another and user as well as modeling frameworks to perform conceptual modeling tasks through polymorphism. The interdependency between the behavior of a KBA and modeling frameworks is defined through KBA's interface. We describe three proposed KBAs as follows.

- Logical model construction agent (LMCA). LMCA defines the logic flows of entities in a simulation model following a process-oriented view, and specifies logical activities, in the context of simulation, involved in the flow. The flow can be specified in two different ways. When the basic configuration of a system type is *indefinite* (e.g., a general manufacturing shop), the entity flows can be created through a general tree-traverse algorithm. By "indefinite", we meant that entity flows in a system do not follow an obvious pattern and cannot be specified from predetermined flow templates. On the other hand, when the basic configuration is *definite*, we can specify entity flows by specializing predetermined flow templates. In reality, human experts synthesize logical models based on their understanding of system dynamics and simulation requirement. For LMCA agent to work, it needs to capture such empirical procedural knowledge through algorithms and rules, and represent them in a computer implementable form so that the execution can be automated and the knowledge reused to specify logical models for similar applications with indefinite or definite configuration.

- Problem description agent (PDA): its function is to classify a given problem and determine the model objectives and I/O requirements for simulation. Mapping from user descriptions to model requirements (e.g., objectives) is a difficult process. It involves sequences of inter-dependent decision making in a multi-dimensional problem space. Here a dimension is considered as one aspect that characterizes the problem (e.g., system type). Although there has been little help in practice to guide a user in describing a problem efficiently and effectively, experienced modelers know what and how to ask relevant questions to quickly and accurately characterize the problem and recommend accordingly the most appropriate set of objectives and map them into input and output (**I/O**) requirement. Our PDA needs to be empowered by such knowledge and have a mechanism that can efficiently process user input and the knowledge to plan and control the dialogue/interaction, following a modified *backward* approach (Tao and Nelson 1994, Zhou et al. 2005b). Given the special characteristics and complexity associated with different types of systems, it is possible to use a number of PDAs, each being specialized in one type of systems.

- Conceptual model simulator/analyzer (CMSA): This agent performs computer-aided verification and validation (V&V) tests of developed conceptual models, and animates models to provide user with a more intuitive understanding. This is a unique feature of this research. It incorporated such a function in the modeling environment to allow or enable the computerized test of designed conceptual simulation models. These tests are identified to verify or validate the function and *baseline performance* of conceptual models, not the *final executable simulation programs encoded with a specific tool*. Traditionally such tests are either none-exist or carried out manually through a tedious and time-consuming process. We believe that there is a great potential to improve this evaluation process by formally defining the tests, designing and implementing a knowledge-based agent subsystem to automate the tests and evaluate test results.

To facilitate system design and development, we present an architecture that integrates the subsystems identified for the development of an environment for conceptual simulation modeling (Figure 3). Note that this architectural model is for a decentralized modeling environment that emphasizes the use of knowledge based agents (KBA). These KBAs are capable of performing automatic reasoning with task-specific knowledge. Note that KBAs need to communicate with each other, and in addition, they have to interact with user and other software entities in the system, e.g., to ask or confirm with user or access a Partial Model Database (PMD) that temporarily store the model requirement specifications generated, or a Framework database to retrieve a model framework. The interactions between the agents and between the agents and other modules (e.g., User interface and PMD) can be realized through a Blackboard type of computational structure (Nii 1989). Therefore a Blackboard agent is necessary to provide a space holding state change and other problem-solving information. A prototype of this system is under development at Indiana State University, Center for Systems Modeling and Simulation.

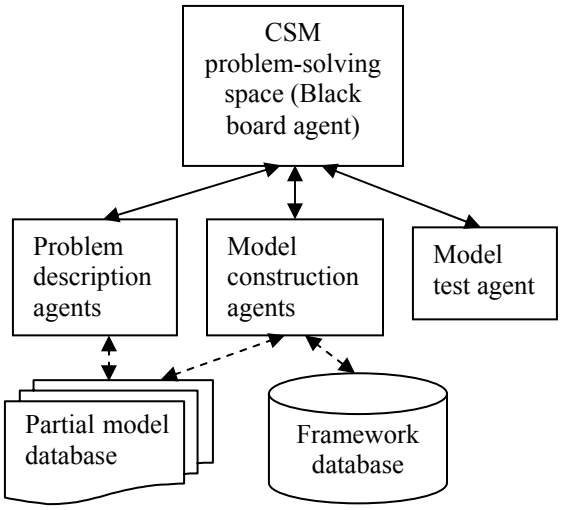

Figure 3: An Architectural Model of a Decentralized Modeling Environment

5 CONCLUSION

Automation can help improve the efficiency and effectiveness of conceptual simulation modeling. However there are a number of issues must be addressed. Model concepts need to be formalized to allow structured and accurate synthesis. Modeling knowledge involved must be represented properly so that they can be effectively and efficiently processed and reused by computer. The interaction between user and computer needs to be implemented through a computerized modeling environment; and for this environment to be robust and effective, a decentralized approach emphasizing the use of specialized collaborative agents can be considered.

REFERENCES

Kim, B. I., S. S. Heragu, R. J. Graves, and A. S. Onge. 2004. Intelligent Agent based Framework for Warehouse Control. *Proceedings of the 37th Hawaii International Conference on Systems Science*.

Larman, C. 1998. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design. Prentice Hall PTR, Upper Saddle River, NJ.

Lewis, M. 1998. Design for Human-Agent Interaction. American Association for Artificial Intelligence. Summer Issue, pp. 67 – 78.

Nicola, J., M. Mayfield, and M. Abney. 2002. Streamlined Object Modeling: Patterns, Rules, and Implementation. Prentice Hall PTR, Upper Saddle River, NJ.

Nii, H. P. 1989. Blackboard Systems. The Handbook of Artificial Intelligence. Volume IV, Chapter XVI (A. Barr, P. R. Cohen and E. A. Feigenbaum, eds.). Addison-Wesley, Massachusetts, pp. 3 – 65.

Norman, D. A. and S. W. Draper. 1986. User Centered System Design: New Perspectives on Human-Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, NJ. pp. 1 – 61.

Robinson, S. 2004. Designing the Conceptual model in Simulation Studies. Proceedings of the 2004 Operational Research Society Simulation Workshop. Operational Research Society, Birmingham, UK, 259-266.

Tao, Y. H. and B. L. Nelson. 1994. A Conceptual Framework for Simulation Experiment Design and Analysis. *Proceedings of the 1994 Winter Simulation Conference*. Washington, D.C.

Zhou, M., Z. Chen, J. Son, and J. Ma. 2004. Knowledge Representations for Conceptual Simulation Modeling. *Proceedings*, 2004 Winter Simulation Conference. Washington D.C.

Zhou, M., Z. Chen, and K. Setavoraphan. 2005. Conceptual Simulation Modeling Of Warehousing Operations. *Proceedings*, 2005 Winter Simulation Conference. Orlando, F.L.

Zhou, M., Y. Son, Z. Chen, Q. Zhang, and J. Ma. 2005. Conceptual Simulation Modeling (I): Patterns and Knowledge Representation, submitted to International Journal of Industrial Engineering, pending on review.

AUTHOR BIOGRAPHIES

MING ZHOU is a professor and program coordinator of Mechanical Engineering Technology at Indiana State University. He is also the Director and Chief Researcher at the Center for System Modeling and Simulation of the university. He received a Ph.D. in Systems and Industrial Engineering from The University of Arizona in 1995. His research interests include knowledge based simulation and intelligent decision support systems. He has been a member of IIE and the Editorial Board, International Journal of Industrial Engineering since 1997. His email address is <mzhou@isugw.indstate.edu>.

ZHIMIN CHEN is a professor and the Associate Dean of the College of Management, Shenzhen University, China. He received a Ph.D. in Engineering from Beijing University of Aeronautics & Astronautics, and had been a visiting professor at Boston University during 1999 and 2000.

QUN ZHANG is a professor emeritus of Tongji University, and currently a Senior Research Associate at the Center for Systems Modeling and Simulation. He has over thirty years of experience in college teaching and academic research. His area of interest includes modern software engineering, and distributed computer information systems.