Springer

Book Announcement (January 2010)

A PROCESS ALGEBRAIC APPROACH TO SOFTWARE ARCHITECTURE DESIGN

by

Alessandro Aldini
Univ. of Urbino – Italy

Marco Bernardo
Univ. of Urbino – Italy

Flavio Corradini
Univ. of Camerino – Italy

ABOUT THE BOOK

Concurrency theory, software architecture, system modeling and verification, and dependability and performance evaluation may seem unrelated disciplines, but in reality they are deeply intertwined. Each of them should be part of an integrated view in order to manage successfully the increasing complexity of nowadays software systems.

The book introduces a process algebraic approach to software architecture design. Process algebra, originally conceived for reasoning about the semantics of concurrent programs, provides a foundational basis for the modeling and verification of functional and nonfunctional aspects of communicating concurrent systems. This can be exploited at the software architecture level of design in order to improve the formality of design documents and to enable the analysis of system properties in the early design stages.

The book, which is intended for graduate students and software professionals, does not focus only on theoretical aspects, but also addresses methodological issues and exhibits application examples.

The first part of the book reports on concepts and results of process algebra theory in a quick and comparative way. It contains background material on the syntax and semantics for process calculi as well as on the bisimulation, testing, and trace approaches to the definition of behavioral equivalences for nondeterministic, deterministically timed, and stochastically timed processes.

The second part of the book provides a number of guidelines for a principled transformation of process algebra into an architectural description language. Then, it addresses the detection of architecture-level mismatches by means of a topological reduction process based on behavioral equivalences, as well as the performance-driven selection among alternative designs by associating queueing network models with process algebraic architectural descriptions. Finally, it shows how to trade dependability features and performance indices in the architectural design phase, by resorting to equivalence-checking-based noninterference analysis and standard numerical techniques.

KEY FEATURES OF THE BOOK

- Emphasizes the benefits of using process algebra in the architectural design phase in terms of formality and analyzability of system descriptions.

- Illustrates a friendly component-oriented way of modeling systems that increases the degree of usability of process algebra.

- Covers the component-oriented analysis of both functional and nonfunctional properties of system models by means of process algebraic techniques.

- Explores methodologies dealing with functional verification, performance evaluation, and the architecture-level integration of dependability and performance.

- Provides background material on process calculi for nondeterministic processes, deterministically timed processes, and stochastically timed processes.

- Compares the bisimulation, testing, and trace approaches to the definition of behavioral equivalences for the various kinds of process.

REFERENCES

ISBN: 978-1-84800-222-7 e-ISBN: 978-1-84800-223-4 DOI: 10.1007/978-1-84800-223-4

TABLE OF CONTENTS

Part I: Process Calculi and Behavioral Equivalences

1. PROCESS ALGEBRA
- 1.1 Concurrency, Communication, and Nondeterminism
- 1.2 Running Example: Producer–Consumer System
- 1.3 PC: Process Calculus for Nondeterministic Processes
  - 1.3.1 Syntax: Actions and Behavioral Operators
  - 1.3.2 Semantics: Structural Operational Rules
- 1.4 Bisimulation Equivalence
  - 1.4.1 Equivalence Relations and Preorders
  - 1.4.2 Definition of the Behavioral Equivalence
  - 1.4.3 Conditions and Characterizations
  - 1.4.4 Congruence Property
  - 1.4.5 Sound and Complete Axiomatization
  - 1.4.6 Modal Logic Characterization
  - 1.4.7 Verification Algorithm
  - 1.4.8 Abstracting from Invisible Actions
- 1.5 Testing Equivalence
  - 1.5.1 Definition of the Behavioral Equivalence
  - 1.5.2 Conditions and Characterizations
  - 1.5.3 Congruence Property
  - 1.5.4 Sound and Complete Axiomatization
  - 1.5.5 Modal Logic Characterization
  - 1.5.6 Verification Algorithm
- 1.6 Trace Equivalence
  - 1.6.1 Definition of the Behavioral Equivalence
  - 1.6.2 Congruence Property
  - 1.6.3 Sound and Complete Axiomatization
  - 1.6.4 Modal Logic Characterization
  - 1.6.5 Verification Algorithm
- 1.7 The Linear-Time/Branching-Time Spectrum

2. DETERMINISTICALLY TIMED PROCESS ALGEBRA
- 2.1 Concurrency, Communication, and Deterministic Time
- 2.2 Deterministically Timed Process Calculi
  - 2.2.1 TPC: Timed Process Calculus with Durationless Actions
  - 2.2.2 DPC: Timed Process Calculus with Durational Actions
- 2.3 Deterministically Timed Behavioral Equivalences
  - 2.3.1 Definition of Timed Bisimulation Equivalence
  - 2.3.2 Congruence Property
  - 2.3.3 Sound and Complete Axiomatization
  - 2.3.4 Modal Logic Characterization
  - 2.3.5 Verification Algorithm
  - 2.3.6 Durational Bisimulation Equivalence and its Properties
- 2.4 Semantics-Preserving Mapping for Eagerness
  - 2.4.1 Differences Between TPC and DPC
  - 2.4.2 From DPC to TPC Under Eagerness
- 2.5 Semantics-Preserving Mapping for Laziness
  - 2.5.1 Lazy TPC
  - 2.5.2 Lazy DPC
  - 2.5.3 From DPC to TPC Under Laziness
- 2.6 Semantics-Preserving Mapping for Maximal Progress
  - 2.6.1 Maximal Progress TPC
  - 2.6.2 Maximal Progress DPC
  - 2.6.3 From DPC to TPC Under Maximal Progress
- 2.7 Expressiveness of Eagerness, Laziness, Maximal Progress
  - 2.7.1 Synchronization Issues
  - 2.7.2 Choosing at Different Times
  - 2.7.3 Performing Infinitely Many Actions at the Same Time
  - 2.7.4 Performing Finitely Many Actions at the Same Time
  - 2.7.5 Coincidence Result for Sequential Processes

3. STOCHASTICALLY TIMED PROCESS ALGEBRA
- 3.1 Concurrency, Communication, and Stochastic Time
- 3.2 MPC: Markovian Process Calculus with Durational Actions
  - 3.2.1 Markov Chains
  - 3.2.2 Syntax and Semantics
- 3.3 Markovian Bisimulation Equivalence
  - 3.3.1 Exit Rates and Exit Probabilities
  - 3.3.2 Definition of the Behavioral Equivalence
  - 3.3.3 Conditions and Characterizations
  - 3.3.4 Congruence Property
  - 3.3.5 Sound and Complete Axiomatization
  - 3.3.6 Modal Logic Characterization
  - 3.3.7 Verification Algorithm
  - 3.3.8 Abstracting from Invisible Actions with Zero Duration
- 3.4 Markovian Testing Equivalence
  - 3.4.1 Probability and Duration of Computations
  - 3.4.2 Definition of the Behavioral Equivalence
  - 3.4.3 Conditions and Characterizations
  - 3.4.4 Congruence Property
  - 3.4.5 Sound and Complete Axiomatization
  - 3.4.6 Modal Logic Characterization
  - 3.4.7 Verification Algorithm
- 3.5 Markovian Trace Equivalence
  - 3.5.1 Definition of the Behavioral Equivalence
  - 3.5.2 Conditions and Characterizations
  - 3.5.3 Congruence Property
  - 3.5.4 Sound and Complete Axiomatization
  - 3.5.5 Modal Logic Characterization
  - 3.5.6 Verification Algorithm
- 3.6 Exactness of Markovian Behavioral Equivalences
- 3.7 The Markovian Linear-Time/Branching-Time Spectrum

Part II: Process Algebra for Software Architecture

4. COMPONENT-ORIENTED MODELING
- 4.1 Software Architecture Description Languages
- 4.2 Running Example: Client–Server System
- 4.3 Architectural Upgrade of Process Algebra: Guidelines
  - 4.3.1 G1: Separating Behavior and Topology Descriptions
  - 4.3.2 G2: Reusing Component and Connector Specification
  - 4.3.3 G3: Eliciting Component and Connector Interface
  - 4.3.4 G4: Classifying Communication Synchronicity
  - 4.3.5 G5: Classifying Communication Multiplicity
  - 4.3.6 G6: Textual and Graphical Notations (PADL Syntax)
  - 4.3.7 G7: Dynamic and Static Operators
- 4.4 Translation Semantics for PADL
  - 4.4.1 Semantics of Individual Elements
  - 4.4.2 Semantics of Interacting Elements
- 4.5 Summarizing Example: Pipe–Filter System
- 4.6 G8: Supporting Architectural Styles
  - 4.6.1 Architectural Types
  - 4.6.2 Hierarchical Modeling
  - 4.6.3 Behavioral Conformity
  - 4.6.4 Exogenous Variations
  - 4.6.5 Endogenous Variations
  - 4.6.6 Multiplicity Variations
- 4.7 Comparisons
  - 4.7.1 Comparison with Process Algebra
  - 4.7.2 Comparison with Parallel Composition Operators
  - 4.7.3 Comparison with Other Software Architecture Languages

5. COMPONENT-ORIENTED FUNCTIONAL VERIFICATION
- 5.1 MismDet: Architecture-Level Mismatch Detection
- 5.2 Class of Properties and Detection Strategy
- 5.3 Architectural Compatibility of Star-Shaped Topologies
  - 5.3.1 Case Study: Compressing Proxy System
- 5.4 Architectural Interoperability of Cycle-Shaped Topologies
  - 5.4.1 Case Study: Cruise Control System
- 5.5 Generalization to Arbitrary Topologies
  - 5.5.1 Case Study: Simulator for the Cruise Control System
- 5.6 Generalization to Architectural Types
  - 5.6.1 Generalization to Internal Behavioral Variations
  - 5.6.2 Generalization to Exogenous Variations
  - 5.6.3 Generalization to Endogenous Variations
  - 5.6.4 Generalization to Multiplicity Variations
- 5.7 Comparisons

6. COMPONENT-ORIENTED PERFORMANCE EVALUATION
- 6.1 PerfSel: Performance-Driven Architectural Selection
- 6.2 Class of Measures and Selection Strategy
- 6.3 Æmilia: Extending PADL with Performance Aspects
- 6.4 Queueing Systems and Queueing Networks
- 6.5 From Æmilia Descriptions to Queueing Networks
  - 6.5.1 General Syntactical Restrictions
  - 6.5.2 Queueing Network Basic Elements
  - 6.5.3 Documental Functions
  - 6.5.4 Characterizing Functions
- 6.6 Case Study: Selecting Compiler Architectures
  - 6.6.1 Sequential Architecture
  - 6.6.2 Pipeline Architecture
  - 6.6.3 Concurrent Architecture
  - 6.6.4 Scenario-Based Performance Selection
- 6.7 Comparisons

7. TRADING DEPENDABILITY AND PERFORMANCE
- 7.1 DepPerf: Mixed View of Dependability and Performance
- 7.2 Running Example: Multilevel Security Routing System
- 7.3 First Phase of DepPerf: Noninterference Analysis
  - 7.3.1 Noninterference Theory
  - 7.3.2 Noninterference Verification
  - 7.3.3 Architectural Noninterference Check
  - 7.3.4 Interpretation and Feedback
- 7.4 Second Phase of DepPerf: Performance Evaluation
  - 7.4.1 Model Validation
  - 7.4.2 Analysis and Tuning
  - 7.4.3 Measure Specification Language
- 7.5 Case Study I: The NRL Pump
  - 7.5.1 Informal Specification
  - 7.5.2 Architectural Description
  - 7.5.3 Noninterference Analysis
  - 7.5.4 Performance Evaluation
- 7.6 Case Study II: Power-Manageable System
  - 7.6.1 Informal Specification
  - 7.6.2 Architectural Description
  - 7.6.3 Noninterference Analysis
  - 7.6.4 Performance Evaluation
- 7.7 Comparisons