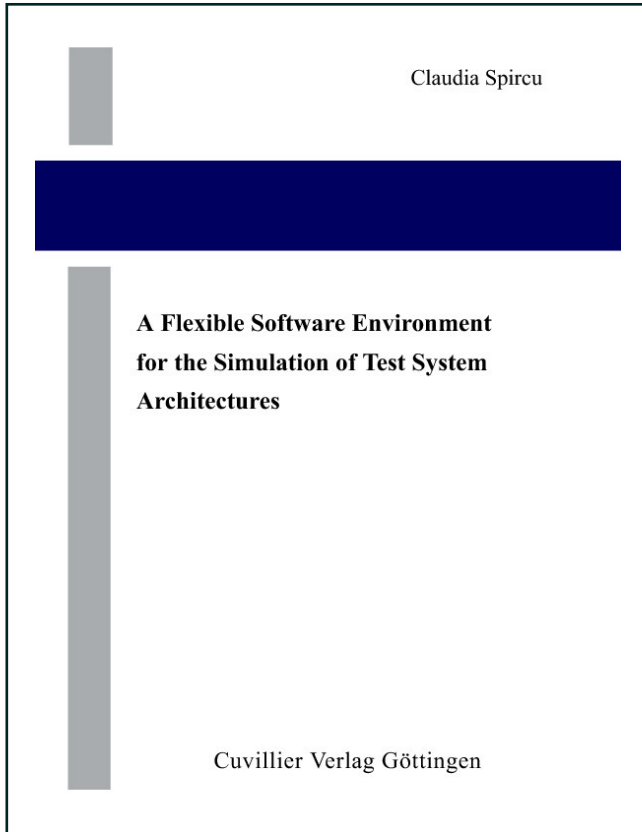




Claudia Spircu (Autor)

A Flexible Software Environment for the Simulation of Test System Architectures



<https://cuvillier.de/de/shop/publications/2202>

Copyright:

Cuvillier Verlag, Inhaberin Annette Jentsch-Cuvillier, Nonnenstieg 8, 37075 Göttingen,
Germany

Telefon: +49 (0)551 54724-0, E-Mail: info@cuvillier.de, Website: <https://cuvillier.de>

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 1 |
| 1.1 | General Motivation | 1 |
| 1.2 | ATE Modeling – Abstraction Levels and Views | 2 |
| 1.3 | The Simulation Levels Used | 3 |
| 1.4 | Used Hardware and Software Models | 5 |
| 1.5 | Motivation for the Creation of a New Software Environment - TSCE | 6 |
| 1.6 | Overview of TSCE | 7 |
| 1.7 | Structure of this Work | 8 |
| 2 | Test Program Modeling | 9 |
| 2.1 | What is a Test Program? | 9 |
| 2.1.1 | Generalities | 9 |
| 2.1.2 | Construction of a Test Step | 9 |
| 2.2 | Test Program Models | 11 |
| 2.2.1 | Simple Task Graphs | 12 |
| 2.2.2 | Hierarchical Task Graphs | 13 |
| 2.3 | Using Simple Task Graphs for Modeling Test Steps | 13 |
| 2.4 | Using Hierarchical Task Graphs for Modeling Test Sequences | 15 |
| 2.5 | Using Hierarchical Task Graphs for Modeling Test Loops | 16 |
| 2.6 | Task Graph Based Execution Time Evaluation | 17 |
| 2.6.1 | Time Evaluation for a Simple Task Graph | 18 |
| 2.6.2 | Time Evaluation for a Hierarchical Task Graph | 19 |

| | | |
|----------|---|-----------|
| 2.7 | Levels of Test Program Modeling | 20 |
| 2.8 | TSCE Implementation of the program package | 21 |
| 2.8.1 | About UML | 21 |
| 2.8.2 | The Implementation of Task Graphs in TSCE using UML | 22 |
| 2.9 | Conclusion | 24 |
| 3 | Test System Architecture Modeling on the Structural Level | 26 |
| 3.1 | What is a Test System? | 26 |
| 3.1.1 | Generalities | 26 |
| 3.1.2 | Components of a Test System | 27 |
| 3.2 | Structural Modeling of a Test System | 29 |
| 3.2.1 | Structure Oriented Models | 29 |
| 3.2.2 | Models Used in This Work | 30 |
| 3.3 | Structural Models for Test System Components | 31 |
| 3.3.1 | Abstract Data Types | 32 |
| 3.3.2 | The Abstract Data Type <i>ATE_MODEL</i> | 33 |
| 3.3.3 | The Hierarchy of Models | 43 |
| 3.4 | Structural Models for a Test System Architecture | 47 |
| 3.4.1 | The Abstract Data Type <i>ATE_ARCHITECTURE</i> | 51 |
| 3.5 | TSCE Implementation | 55 |
| 3.5.1 | Implementation of Component Models — the types package | 55 |
| 3.5.2 | Implementation of Component Model Instances and Architectures — the elements package | 55 |
| 3.5.3 | Graphical Extensions — the graphic package | 57 |
| 3.6 | Conclusion | 59 |
| 4 | Test System Component High Level Behavioral Modeling | 60 |
| 4.1 | What is High Level Behavioral Modeling? | 60 |
| 4.2 | High Level Behavioral Models | 60 |

| | | |
|----------|--|-----------|
| 4.2.1 | The Statecharts | 60 |
| 4.3 | Using Statecharts as Test System Component Behavioral Models | 65 |
| 4.3.1 | Events | 66 |
| 4.3.2 | Event Categories | 66 |
| 4.3.3 | States and Action Functions | 69 |
| 4.3.4 | Transitions | 71 |
| 4.4 | Binding Behavioral Models to Device Types and Device Elements | 72 |
| 4.5 | Behavioral Model Inheritance | 75 |
| 4.6 | Modeling the Data Communication | 77 |
| 4.7 | Examples of Typical Instruments and Action Functions | 81 |
| 4.7.1 | The Digitizer | 81 |
| 4.7.2 | The Generator | 81 |
| 4.7.3 | The DSP | 82 |
| 4.8 | TSCE Implementation of the model package | 83 |
| 4.8.1 | Implementation of the Behavioral Models and Model Instances | 83 |
| 4.8.2 | Implementation of the Events | 83 |
| 4.8.3 | Implementation of the States and Transitions | 83 |
| 4.8.4 | Implementation of the Action Functions | 85 |
| 4.8.5 | The XML Files for Defining Device Types, Models and Hierarchies | 85 |
| 4.9 | Conclusion | 89 |
| 5 | The Simulator | 90 |
| 5.1 | What is Discrete-Event Simulation? | 90 |
| 5.1.1 | Concepts in Discrete-Event Simulation | 90 |
| 5.1.2 | Existing Modeling Languages and Tools | 91 |
| 5.2 | TSCE Simulation Concepts | 94 |
| 5.3 | Overview of the TSCE Simulation Environment | 95 |
| 5.4 | The Event Scheduling / Time Advance Mechanism | 97 |
| 5.4.1 | The Abstract Data Type <i>EVENT_LIST</i> | 97 |
| 5.4.2 | Possible Implementations of the Abstract Data Type <i>EVENT_LIST</i> | 99 |

| | | |
|----------|--|------------|
| 5.4.3 | The Simulation Algorithm | 101 |
| 5.4.4 | Event Handling in the Models | 102 |
| 5.4.5 | Scheduling Simple nodes | 104 |
| 5.4.6 | Scheduling CALL Hierarchy Nodes | 105 |
| 5.4.7 | Scheduling BRANCH Hierarchy Nodes | 106 |
| 5.4.8 | Scheduling LOOP Hierarchy Nodes | 106 |
| 5.4.9 | The Simulation Context | 106 |
| 5.5 | A Complete Example | 107 |
| 5.6 | A Second Example - Simulating a LOOP | 120 |
| 5.7 | Complexity of the Simulation Algorithm | 122 |
| 5.8 | TSCE Implementation of the simulator package | 125 |
| 5.9 | Conclusion | 127 |
| 6 | Further Development – Test Synthesis | 128 |
| 6.1 | The Problem | 128 |
| 6.2 | Existent Algorithms | 131 |
| 6.3 | Conclusion | 134 |
| 7 | Conclusion and Future Work | 135 |
| 7.1 | Conclusion | 135 |
| 7.2 | Future Work | 138 |
| 8 | Zusammenfassung und Ausblick | 139 |
| A | Petri Nets and Petri Net Based Models | 143 |
| A.1 | The <i>Petri Net</i> Model | 143 |
| A.1.1 | The Petri Net Graph | 143 |
| A.1.2 | Execution Rules in Petri Nets | 145 |
| A.1.3 | Properties of Petri Nets | 146 |
| A.1.4 | Extended Petri Nets | 149 |
| A.2 | Petri Net Based Models | 151 |

| | | |
|----------|---|------------|
| A.2.1 | State Oriented Models | 151 |
| A.2.2 | Data Flow Oriented Models | 153 |
| A.2.3 | Heterogeneous Models | 159 |
| B | Used ATE Components Models | 160 |
| B.1 | The Root | 160 |
| B.2 | The Device | 161 |
| B.3 | The Instrument | 161 |
| B.3.1 | The Digitizer | 161 |
| B.3.2 | The Generator | 163 |
| B.3.3 | The DSP | 163 |
| B.4 | The Communication | 164 |
| C | Used Data Structures and Operations | 166 |
| C.1 | (Rooted) Trees | 166 |
| C.2 | Binary Trees | 167 |
| C.3 | Heaps | 169 |
| C.3.1 | <u>AddHeap</u> - Adding a New Item into a Heap | 169 |
| C.3.2 | <u>GetMin</u> - Getting the Item with the Minimum Key from a Heap | 170 |
| C.3.3 | <u>DeleteMin</u> - Deleting the Item with the Minimum Key from a Heap . . | 171 |
| | Bibliography | 173 |