State-Driven Programming

Georgiy Korneev

Anatoly Shalyto

Saint Petersburg State University of Information Technologies, Mechanics and Optics

Computer Technology Department

Research Goal

- Automata in hardware engineering
  - Applied since 50th
  - A lot of theories and formal methods
- Automata in software engineering
  - Compilers
  - Protocol specifications
  - Statecharts
  - State-driven programming

Research Guidelines

- State-driven programming
- Object-oriented state-driven programming
- Code-based automata construction
- Automata models verification
- Development tools
- Genetic algorithms

State-Driven Programming
Fundamentals

- State
  - Control state
  - Computational state
- Transition
- Input actions
  - Events
  - Input variables
- Output actions

State-Driven Programming
Benefits

- Unified approach to complex behavior systems engineering
- Logical errors detection on early stages
- Design, implementation debugging and documentation in common terms
- Scalability
- Automated code generation

State-Driven Programming
Application Areas

- Programmable logic controllers
- Embedded systems
- PC-based control systems
- Web-applications

OO State-Driven Programming
Guidelines

- Object-oriented automata wrapping
  - Automated code generation
- Object-based automata decomposition
  - State-like design patterns
- Automata inheritance
  - Semantics
  - Notation

OO State-Driven Programming
OO Automata decomposition

- State-like design patterns analysis
- State Machine design pattern
- State Machine programming language
  - Expanding Java with state-driven programming concepts
  - Text-based state-driven language

OO State-Driven Programming
Automata Inheritance

- Semantics
  - Formal automata models
  - Liskov substitution principle for automata
  - Automata inheritance requirements
- Notation
  - Compact graphical automata inheritance notation

Code-based automata construction
Guidelines

- Code-based automata model construction
- State-driven algorithm animators

Code-based automata construction

Automata model construction

- Models for explicit recursive procedures
  - Single automata
  - Stack-based approach
- Models for recursive programs
  - Interactive automata systems
  - Automata instantiation approach

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Code-based automata construction
Algorithm Animators

- Bidirectional algorithms tracing
  - Recursion support
  - Control and calculation state preserving
- Reverse execution supported in automata models
  - Low memory and time requirements
  - Stack-based architecture
- State-driven algorithm animation framework
  - http://neerc.ifmo.ru/vizi
  - http://neerc.ifmo.ru/vizi/examples

ИТМО ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Automata models verification
Guidelines

- Validation
- Behavior verification
- State-driven testing and debugging

Automata models verification
Behavior Verification

- Model checking
  - Kripke models
  - Linear temporal logic
- Implementation
  - Mapping automata model to Promela model
  - Contrary instance search with Spin
  - Backward mapping of contrary instance to automata model

Automata models verification
Testing and Debugging

- State-driven testing
  - State covering
  - Transition covering
  - Automated regression testing
- Debugging
  - Automated logging
  - Execution path reconstruction

Development Tools

- Integrated development environment
  - UniMod
- Support libraries
  - STOOL
  - OState
- Text-based state-driven languages
  - State
  - State Machine

Development Tools
UniMod (1)

- UML-based automata notation
  - Nested states support
  - Diagram editor
- Syntax diagram verification
  - Error highlight
  - Code completion
- One-click diagram execution
- Local and remote state-driven debugging
- Interpreting and executable code generation

Development Tools
UniMod (2)

- Application areas
  - Client-server application for Java ME, SE and EE
  - Symbian applications (C++)
- Open source
- Homepage http://unimod.sf.net

Development Tools
Support Libraries

- STOOL (C++)
  - Automated logging
  - Exception handling
  - Multithreading support
- OState (Java)
  - OR & AND state support
  - Generalized transitions

Genetic Algorithms for Automata Construction

- Automata representation
  - Reducing of chromosomes space
  - Structured representation
- Mutation and crossover
  - Effective automata mutations
  - Effective automata crossovers

Scientific School

- Since 1998
- Head: Anatoly Shalyto
  - Founder of state-driven programming in Russia
- 3 PhD
- 10 PhD students

Thank You for Attention

Результаты опубликованы

- Журналы
  - “Программирование”
  - “Автоматика и телемеханика”
  - “Известия РАН. Теория систем управления”
  - “Искусственный интеллект”
- Конференции
  - Телеметика 2000-2004
  - Linux Summit 2003
  - KIMAS 2003 и 2005