Foundation for Open Project Documentation

Anatoly Shalyto

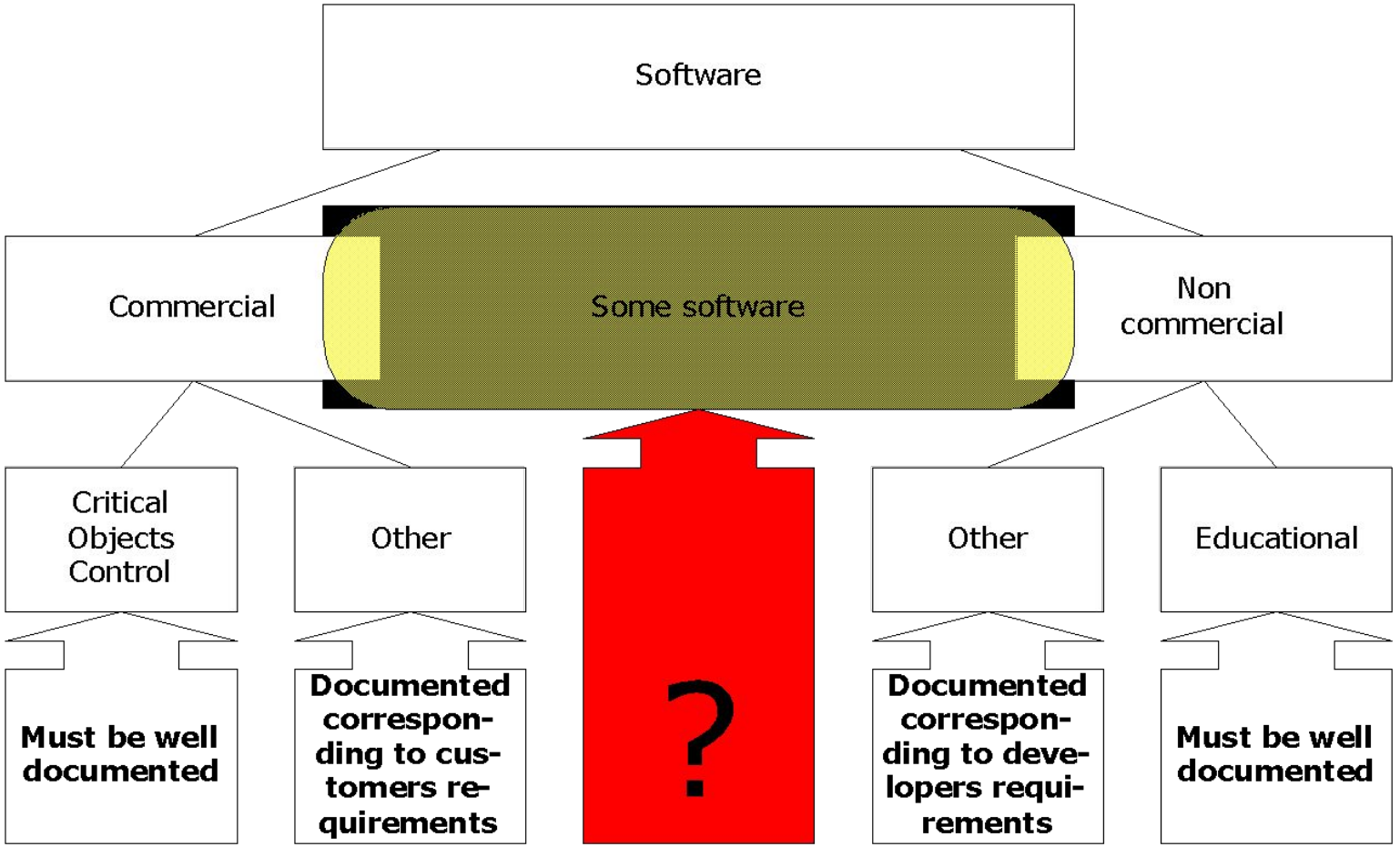
shalyto@mail.ifmo.ru

Computer Technology Department

Saint-Petersburg State University of Information Technology, Mechanics and Optics,

Russia

Project Documentation in the Real World (1)



Project Documentation in the Real World (2)

- Commercial Software
  - Critical Objects Control Software
    - Main — by standards
    - Extra — by customer needs
  - Other
    - By customers requirements
- Non commercial software
  - Educational
    - Good quality
  - Other
    - By developers wishes

Hardware Documentation vs. Software Documentation

- Designers and manufacturers are different people
- Hardware Documentation
  - Project Documentation
  - Design Basis
  - Verification Results
  - User Guide

- Designers and manufacturers are same people
- Software Documentation
  - User Manual
  - Developers Guide
  - Source Code (for open source projects)

Open Project Documentation

- Why Project Documentation?
  - Software quality improvement
  - Better verification
  - Faster and safer modification
- Why Open Project Documentation?
  - Open project documentation increases freedom
  - Better project understandability
  - Project design borrowing
  - Educational purposes
    - For students
    - For specialists

Why Only Open Project Documentation?

- **Open** = Must be available for further using and development
- *Foundation for Open project Documentation* is **Free**, but it is in different area comparing with *Free Software Foundation* or *Open Source Foundation*
  - Foundation results is applicable not only for free software, but also for commercial, secret and other kinds of software

Software Project Documentation

- In engineering practice projects must be well-documented
  - So on www.sourceforge.net there are not 76000 project, but much fewer
- The code must be based on the project documentation, not vice versa
- Project execution flow must be documented, not only final results

SWITCH-technology (Automata Programming)

- Proposed in 1991
- Based on states decomposition
- Model-driven development
- Usage scope – systems with complex behavior
- Applicable for different type of computing devices
  - Logic Controllers Programming
  - Microcontrollers
  - Microprocessors

SWITCH-technology guidelines

- Logic control
- State-based procedural programming
- State-based object-oriented programming
- Computational algorithms

SWITCH-technology Basics

- State
- Set of states

- Input variables + Events = Input Actions
- States + Input Actions = Automata With No Output
- Automata With No Output + Output Actions = Automata

- States are encoded with multiple values
- Observation of the automata states
- Correlated automata systems
- Logging
- Project documentation

Automata in Automata Programming

- Logic specification language
- Isomorphic mapping to source code
- Program works and builds logs in terms of automata

Educational Experiment (1)

- Computer Technology Department in University of Information Technology, Mechanics and Optics, Saint Petersburg, Russia
  - Chosen students from the whole Russia
  - International Olympiads in Informatics medalists
  - ACM International Collegiate Programming contest medalists

Educational Experiment (2)

- 1998-2001 Common Teaching 1
  - Lectures and Exams
- 2001-2002 Common Teaching 2
  - Lectures, Course Works and Exams
- 2002-2003 Experimental Teaching
  - Lectures and Projects
  - Project Documentation Verification
  - More than 40 fully Developed and Documented Projects
- To be continued

Educational Experiment (3)

- Project Contents
  - Project Documentation (at least 60 hours)
    - Problem Definition
    - User Interface Description
    - Justifications
    - Automata and Classes Descriptions
    - Automata and Classes Diagrams
    - Verification Protocols
    - References
  - Source Code (at least 20 hours)
- Anatoly Shalyto spent approximately 10–15 hours per project

Projects Examples

- Games
- Skeleton animation
- Controlling systems
- Graphical User Interfaces
- Parallel problems
- Transliteration
- Many others

Three Examples

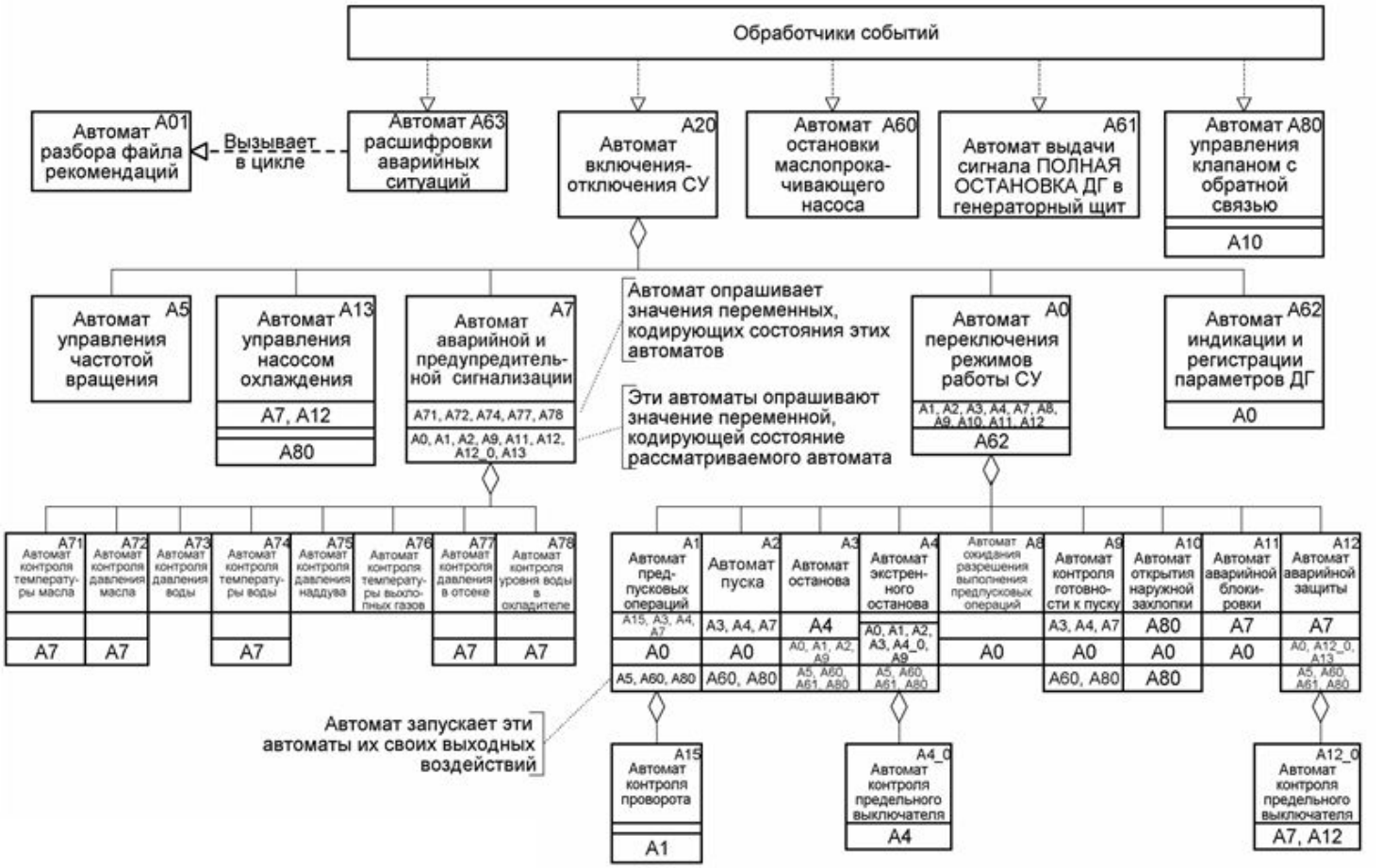
- Diesel-generator modeling
  - Procedural automata programming
- *RoboCode* Agent
  - Object-oriented automata programming
- Visualization Framework
  - Switch-technology based visualization of calculation algorithms
  - Object-oriented realization of procedural algorithms

Diesel Generator

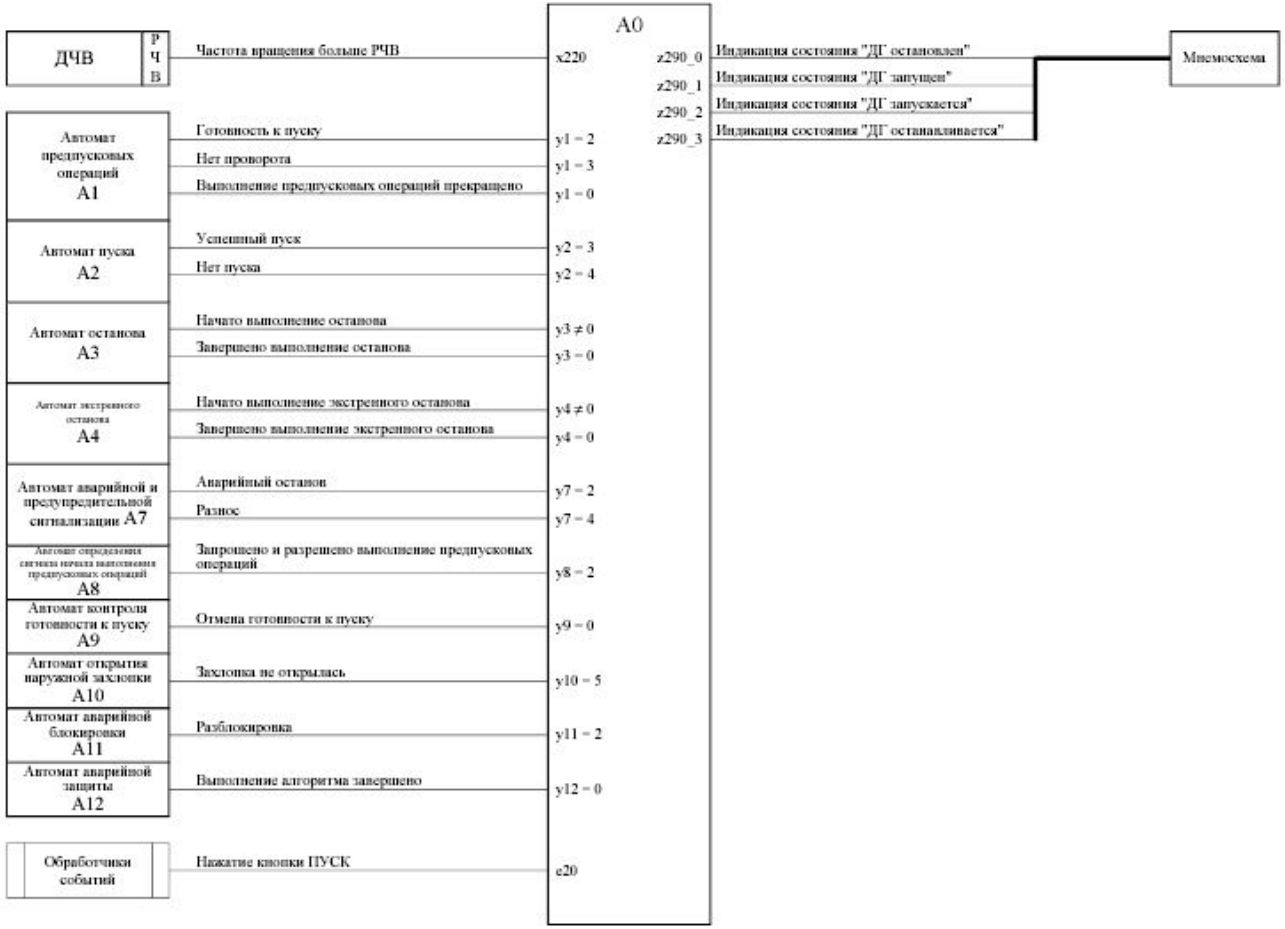
Project Execution Flow

- Analysis
- Structuring (Automata Decomposition)
- Automata Interaction Diagrams
- Automata Verbal Descriptions
- Automata Interface Definition
- Automata Transitional Graph Definition
- Isomorphic Source Code Generation
- Verification logs

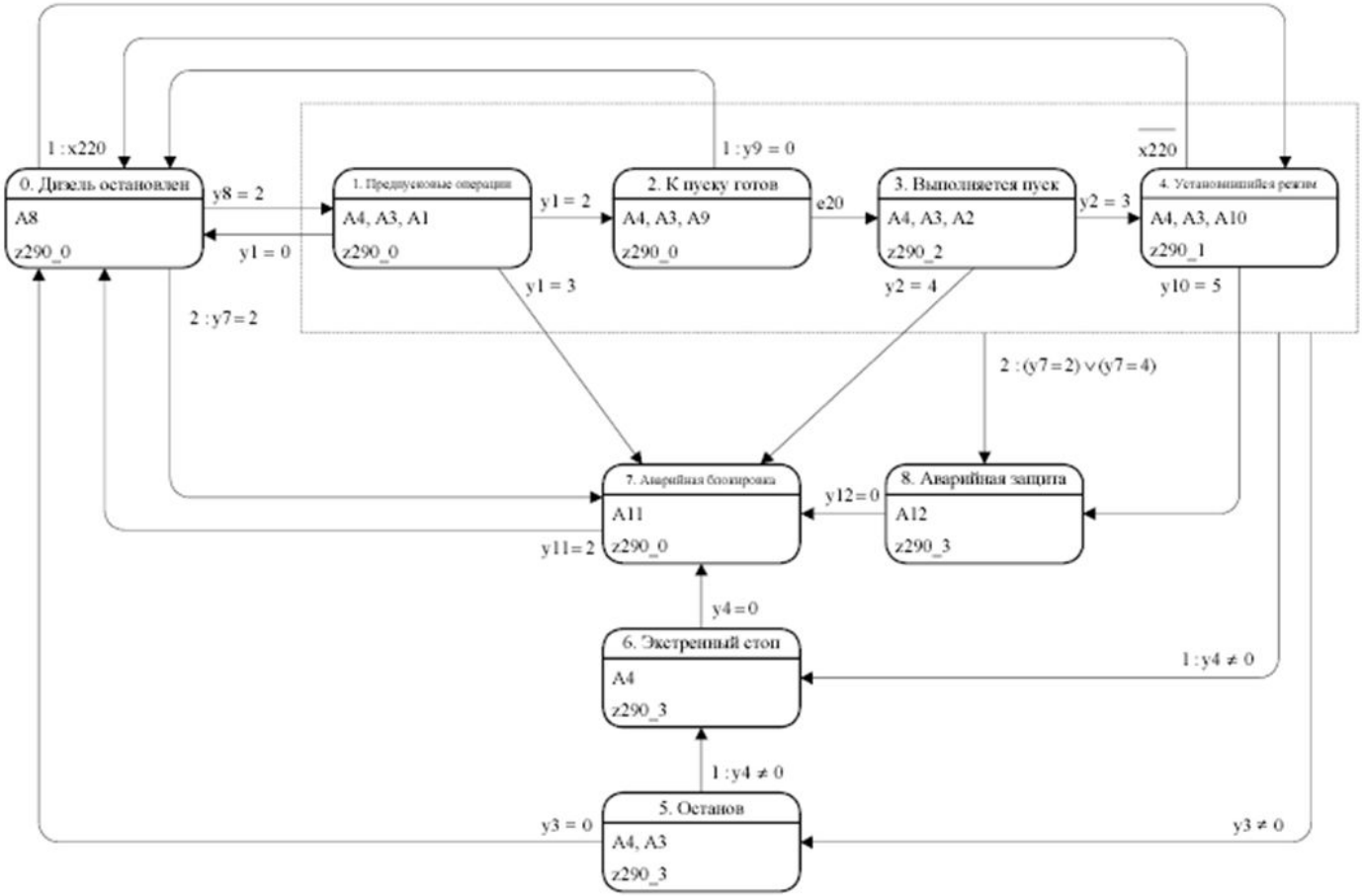
Diesel-generator
Automata Interaction Diagram



Diesel-generator
Automata Interface Diagram



Diesel-generator
Automata Transition Graph



Diesel-generator
Isomorphic Source Code

```
void A0(int e, dg_t *dg) {
    int y_old = dg->y0;
    switch(dg->y0) {
        case 0:
            A8(e, dg);
            if (x220(dg)) dg->y0 = 4;
            else if (dg->y7 == 2) dg->y0 = 7;
            else if (dg->y8 == 2) dg->y0 = 1;
            break;
        case 1:
            A4(e, dg); A3(e, dg); A1(e, dg);
            if (dg->y4 != 0) dg->y0 = 6;
            else if (dg->y7 == 2 || dg->y7 == 4) dg->y0 = 8;
            else if (dg->y3 != 0) dg->y0 = 5;
            else if (dg->y1 == 0) dg->y0 = 0;
            else if (dg->y1 == 3) dg->y0 = 7;

    ...
```

Diesel-generator
Verification Logs

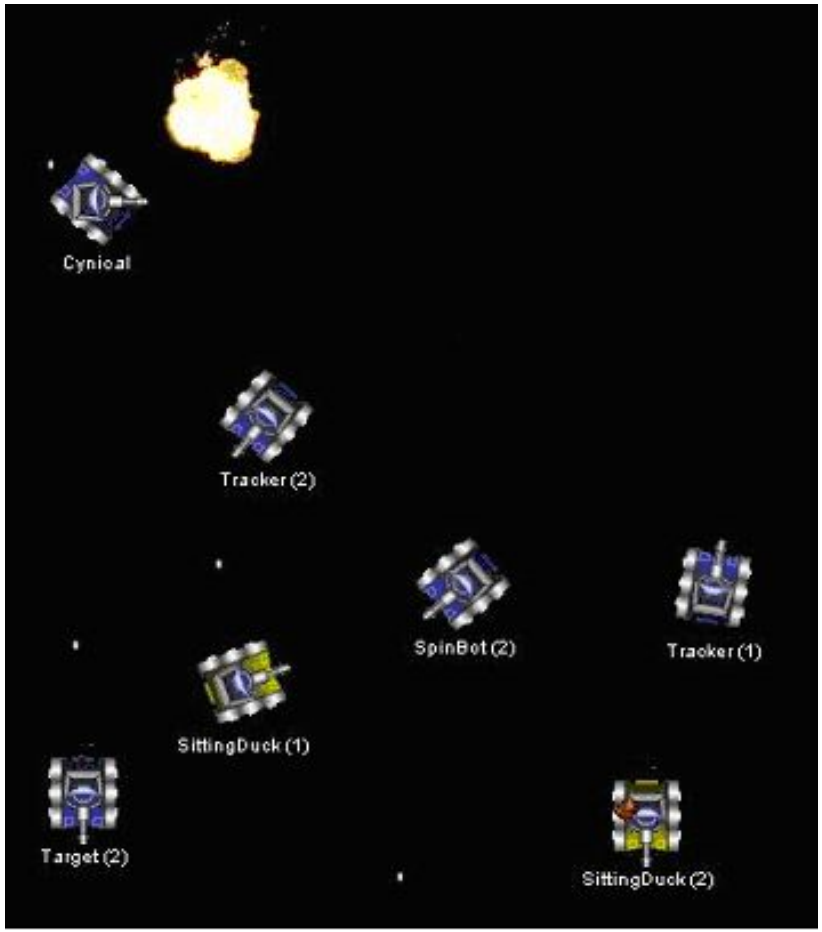
```
11:34:02.507{ DG1: A20: started at state 2 with event e10
11:34:02.507{ DG1:  A7: started at state 0 with event e10
11:34:02.507{ DG1: A71: started at state 0 with event e10
11:34:02.507> DG1: x320 – lubricting oil temperature less than Tmm
11:34:02.507> DG1: x330 - lubricting oil temperature greater than Tmpr
11:34:02.507} DG1: A71: stopped at state 0
11:34:02.507{ DG1: A72: started at state 0 with event e10
11:34:02.507> DG1: x220 – rotation frequency greater than RCV
11:34:02.507} DG1: A72: stopped at state 0
11:34:02.507{ DG1: A73: started at state 0 with event e10
11:34:02.507> DG1: x220 – rotation frequency greater than RCV
11:34:02.507} DG1: A73: stopped at state 0
11:34:02.507{ DG1: A74: started at state 0 with event e10
11:34:02.507> DG1: x430 – water temperature less than Tvm
11:34:02.507> DG1: x440 - water temperature greater than Tvpr
…
11:34:02.517} DG1: A20 stopped at state 0
```

RoboCode Agent

http://robocode.alphaworks.ibm.com

Top Five

1. GlowBlowMelee 1.1
2. Cigaret 1.20
3. **Cynical**
4. GlowBlow
5. **Cynical_3**

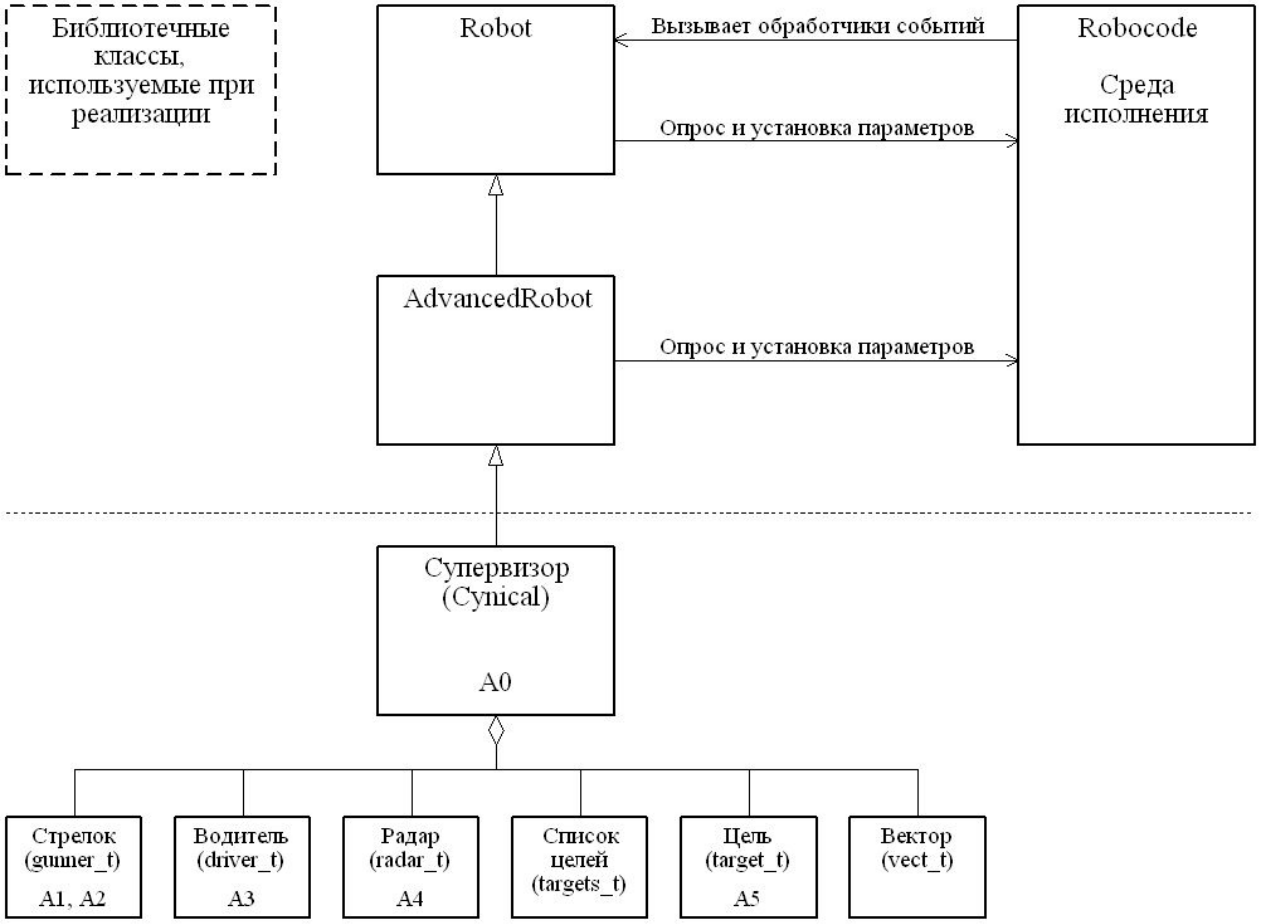


RoboCode Agent

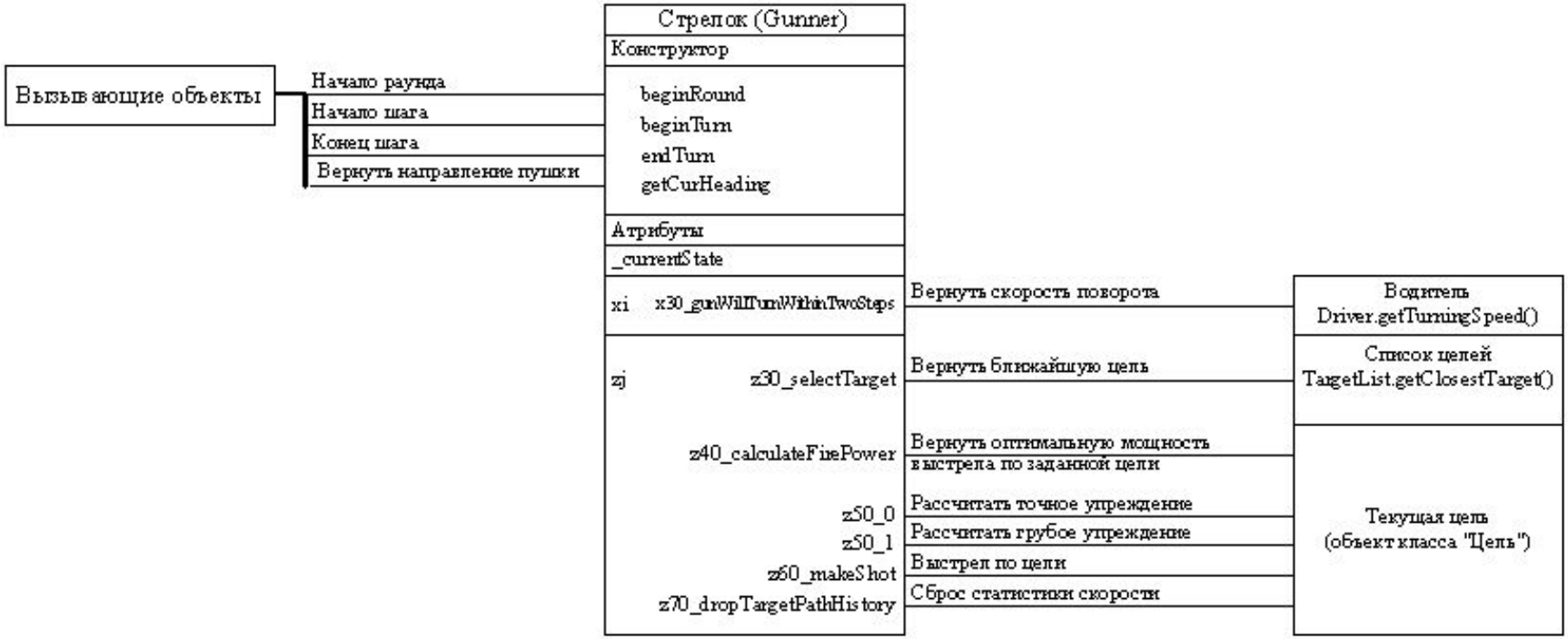
Project Execution Flow

- Analysis
- Structuring (Class Decomposition)
- Classes Diagram
- All the stages from diesel-generator project execution flow

RoboCode Agent
Classes Diagram

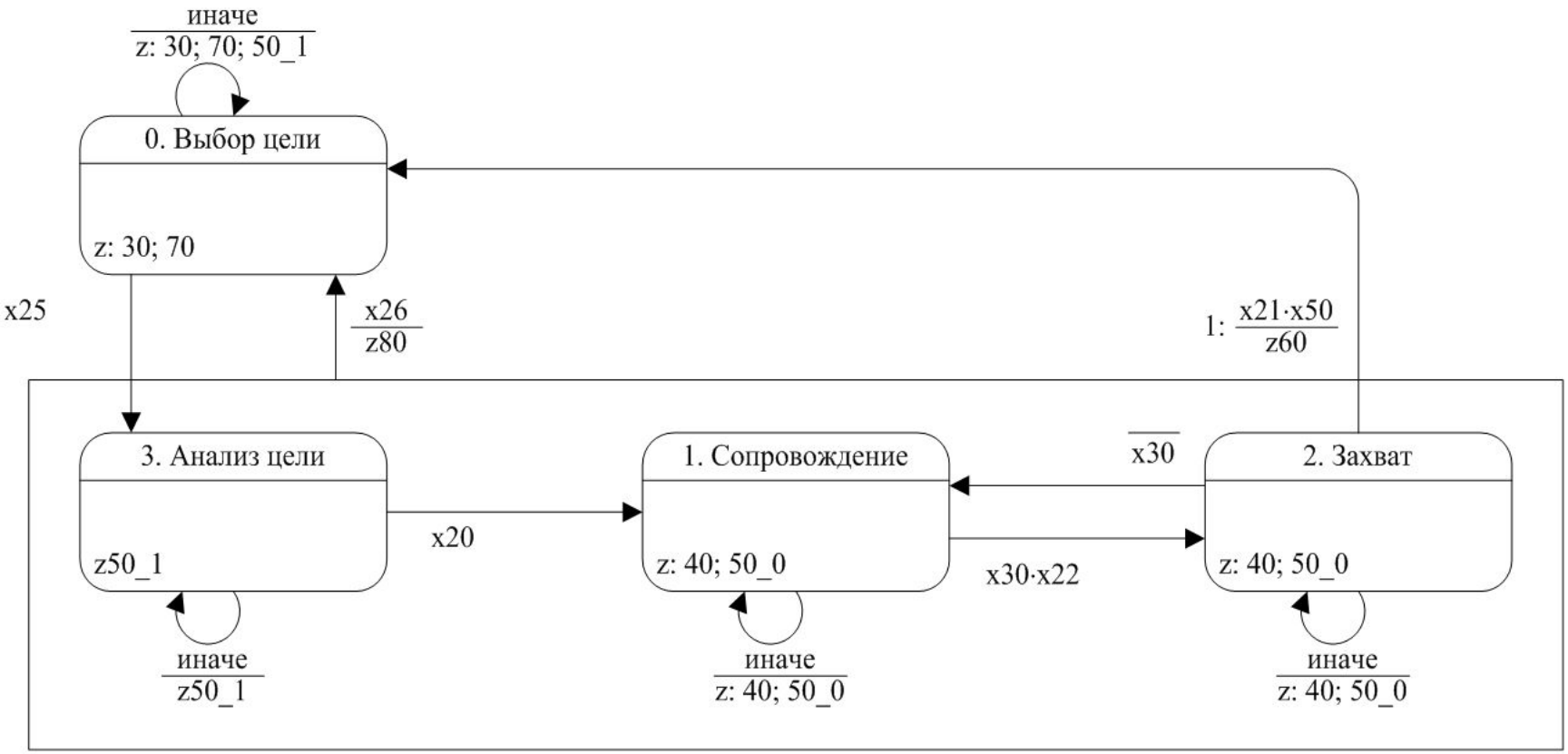


RoboCode Agent
Class Structure Diagram



RoboCode Agent
Automata Interface

RoboCode Agent
Transition Graph



RoboCode Agent
Debugging through Protocols

——— 0 ——— **Начальный шаг (событие 9)**

Для объекта 'Супервизор':

{ A0(Supervisor): Автомат A0(Supervisor) запущен в состоянии State 1 с событием e10

* z10_2: Инициализация в начале шага.

} A0(Supervisor): Автомат A0(Supervisor) завершил свою работу в состоянии State 1

Для объекта 'Стрелок':

{ A1(Gunner): Автомат A1(Gunner) запущен в состоянии State 0 с событием e10

i x25: Цель выбрана? - НЕТ.

* z30: Выбрать цель.

* z70: Сбросить историю маневрирования цели.

* z50_1: Рассчитать приблизительное упреждение и направить пушку.

} A1(Gunner): Автомат A1(Gunner) завершил свою работу в состоянии State 0

Для объекта 'Радар':

{ A4(Radar): Автомат A4(Radar) запущен в состоянии State 0 с событием e10

i x70: Цикл сканирования завершен? - НЕТ.

i x70: Цикл сканирования завершен? - НЕТ.

* z100_0: Повернуть радар влево.

} A4(Radar): Автомат A4(Radar) завершил свою работу в состоянии State 0

Для объекта 'Водитель':

{ A3(Driver): Автомат A3(Driver) запущен в состоянии State 0 с событием e10

i x100: Враг близко? - ДА.

i x110: Сработал таймер T110? - ДА.

* z200_0: Инициализация движения по траектории 'Маятник'.

* z200_1: Добавить случайную составляющую к траектории 'Маятник'.

* z200_2: Определить направление и скорость движения 'Маятник'.

} A3(Driver): Автомат A3(Driver) завершил свою работу в состоянии State 0

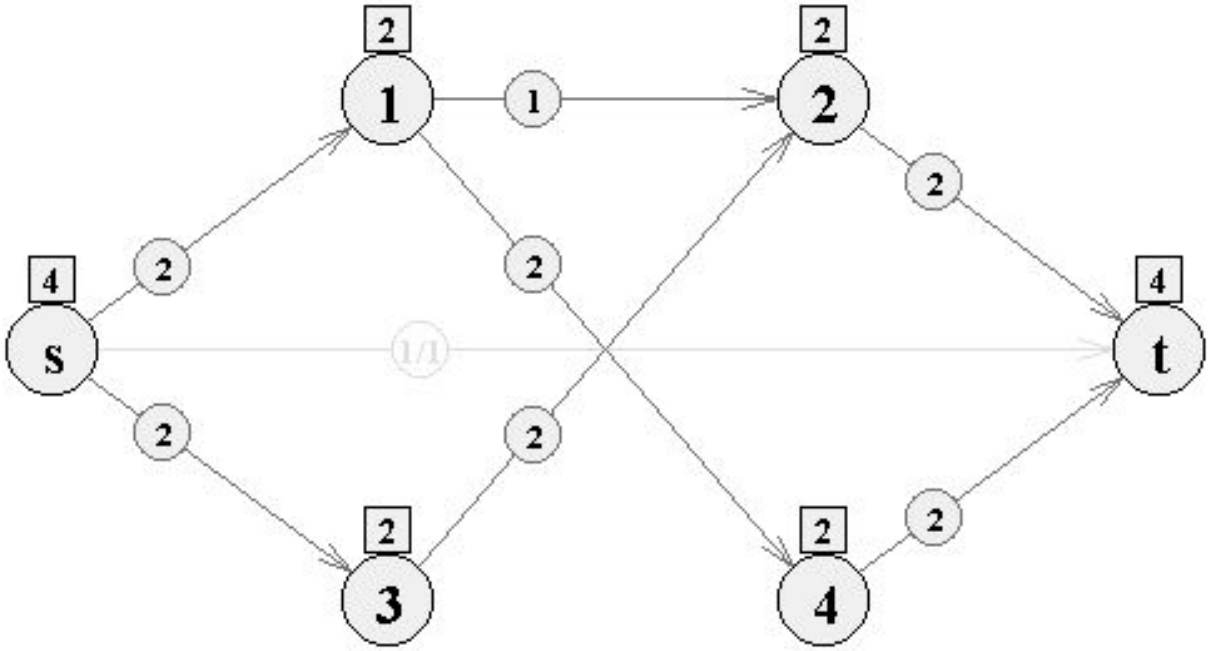
——— 30 ——— **Выстрел по цели**

Для объекта 'Супервизор':

{ A0(Supervisor): Автомат A0(Supervisor) запущен в состоянии State 1 с событием e10

Visualization Framework

Malhotra, Kumar, Maheshwari network flow algorithm



Visualization Framework
Visualizer Structure

- Model
  - Interactive Automata System Automatically Generated by Algorithm's XML–Description
- View
  - User Interface Based on *Vizi* Library
- Controller
  - *Vizi* Library

Visualization Framework
Project Documentation (1)

- Annotation
- Introduction
- Chapter 1. Literature Analysis
- Chapter 2. Algorithm Description
- Chapter 3. Algorithm Implementation
- Chapter 4. Data Model Definition
- Chapter 5. Implementation Transformation
- Chapter 6. User Interface Description

Visualization Framework
Project Documentation (2)

- Chapter 7. Configuration Description
- Conclusions
- References
- Appendixes
  - Algorithm Implementations Source Code
  - Transformed Implementation
  - Visualizer XML–description
  - Generated Source Codes
  - User Interfaces Source Codes

Visualized Algorithms

- Malhotra, Kumar, Maheshwari network flow algorithm
- Dinic's network flow algorithm
- Hopcroft–Karp Bipartite Matching algorithm
- Chu–Liu shortest arborescence of a directed graph
- Algorithms on 2–3 threes
- Bitonic salesman problem
- Ukkonen suffix tree construction algorithm
- Prim minimum spanning tree algorithm
- Simple strings and de Bruin cycles construction algorithms

Links

- Project Examples
  - http://is.ifmo.ru/?i0=projects – **Projects Documentation (Russian)**
  - http://is.ifmo.ru/?i0=projects_en – **Projects Annotations (English)**
  - http://is.ifmo.ru/?i0=works – **Articles (Russian)**
  - http://is.ifmo.ru/?i0=english – **Articles (English)**
  - http://unimod.sourceforge.net/ – **UniMod Project**