Design and implementation a software for water purification with using automata approach and specification based analysis

Sergey Staroletov (Polzunov Altai State Technical University)

Eighth Workshop
Program Semantics, Specification and Verification:
Theory and Applications
(PSSV 2017, June 26, 2017)

Purpose of the work

- Our university received an order for the research work to design and development software under the UN grant for developing countries in the field of energy conservation.
- A customer is developing a hardware plant providing preparation of distilled water of the given temperature and testing the energy consumption and water consumption of various connected devices (for example, washing machines and dishwashers).

Water purification

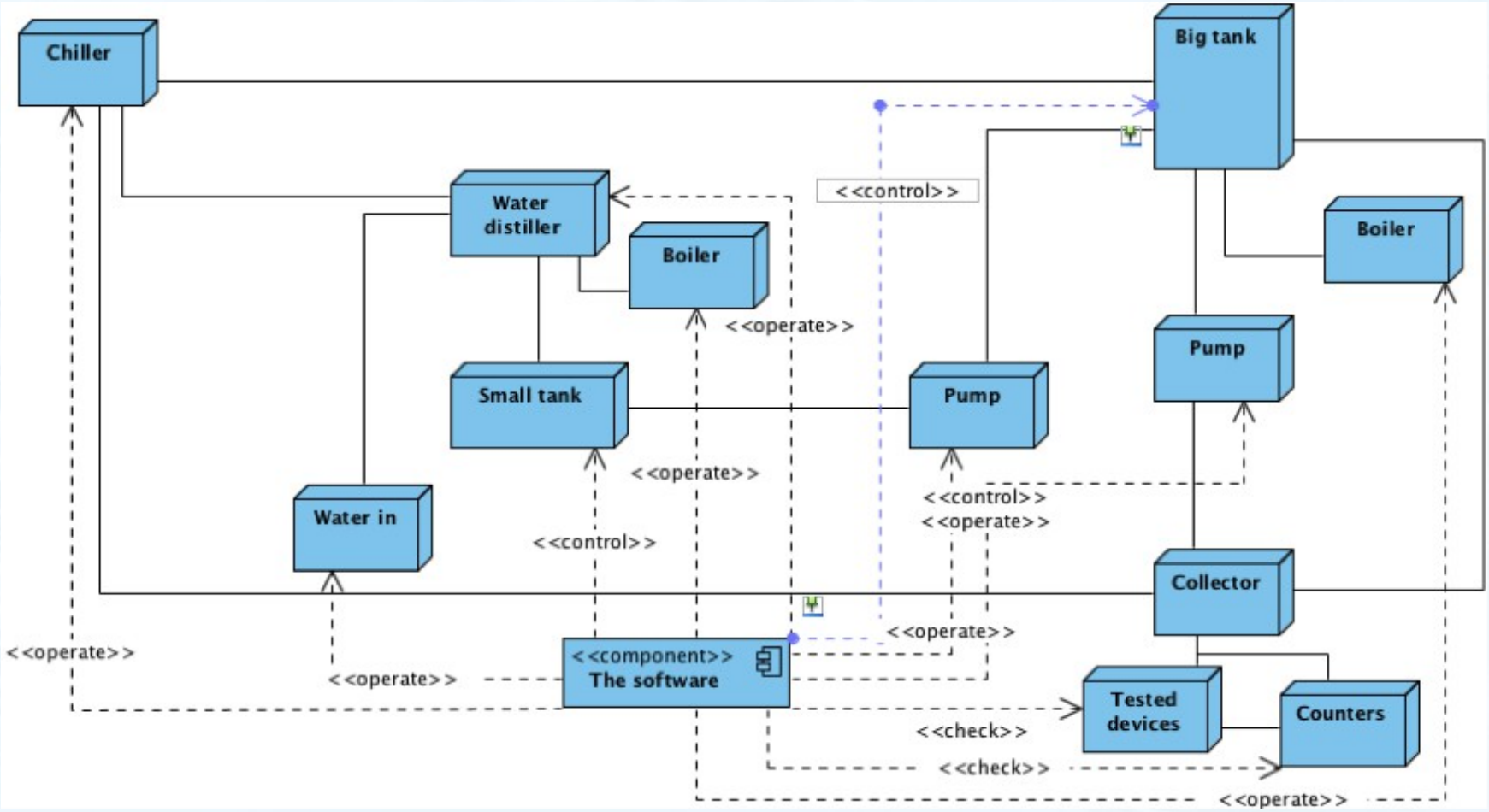
- Create the distillate (heating, collecting, pumping)
- Normalize to a given temperature (circulating, heating, cooling)

Related Work

Supervisory control and data acquisition (SCADA) concept. We build the own system because:

- Customizable SCADA systems: very big price
- Lack of support for existing devices
- Need to implement a software either to run the process, to make graphs and reports and good UI for non-engineers
- Need to license the hardware and software

Hardware components



Statement of the work

- All water preparation and equipment testing should be performed without user intervention by the automatic operations in the hardware stand and the operator's job is only to select the mode, to set parameters and then run the process
- Novelty consists primarily in mandatory to implement algorithms for water purification in this plant and control of hardware devices
- These algorithms must be primarily reliable because water is supplied under high pressure, is heating using amperage of tens of amperes, and all possible exceptional operations must be processed

The reason

- Design the software
- Implement the software
- Get some troubles
- Create a method for creating suchlike high-quality systems with minimum of troubles

A fragment of customer's specification

Algorithm: Distillate cycle.

(Active before filling the large / storage tank)

Step 0: Start preparation ...

Step 1: The water pressure is checked at the input

(check sensor No ...) ...

Step 2: Filling the distiller

The level sensor in the distiller is monitored for

(time of opening the valve) after opening valve N...

Yes / parameters OK:

1) Valve ... is closed;

2) Transition to step 3.

No / the parameters are not normal:

1) Message (Filling of filling of the distiller);

2) The automatic mode is switched off.

Step 3: Distillation

1) The heating of the distiller is activated (Button ...);

2) Parameters are monitored:

- amperage;

- temperatures of the distiller;

- filling sensor;

- the sensor for filling the storage tank.

Yes / parameters OK:

1) Accumulation distillate in the tank

2) Executing step 3 again

No / the parameters are not normal:

- The amperage exceeds the set maximum:

1) The heating of the distiller is switched off;

2) The transition to 3.1.

- The amperage with the fill sensor turned on is below the set minimum:

1) The heating of the distiller is switched off;

2) The transition to 3.2.

- The amperage with the filling sensor switched off is below the set minimum:

Go to step 1.

- The temperature is higher than the given maximum:

1) The heating of the distiller is switched off;

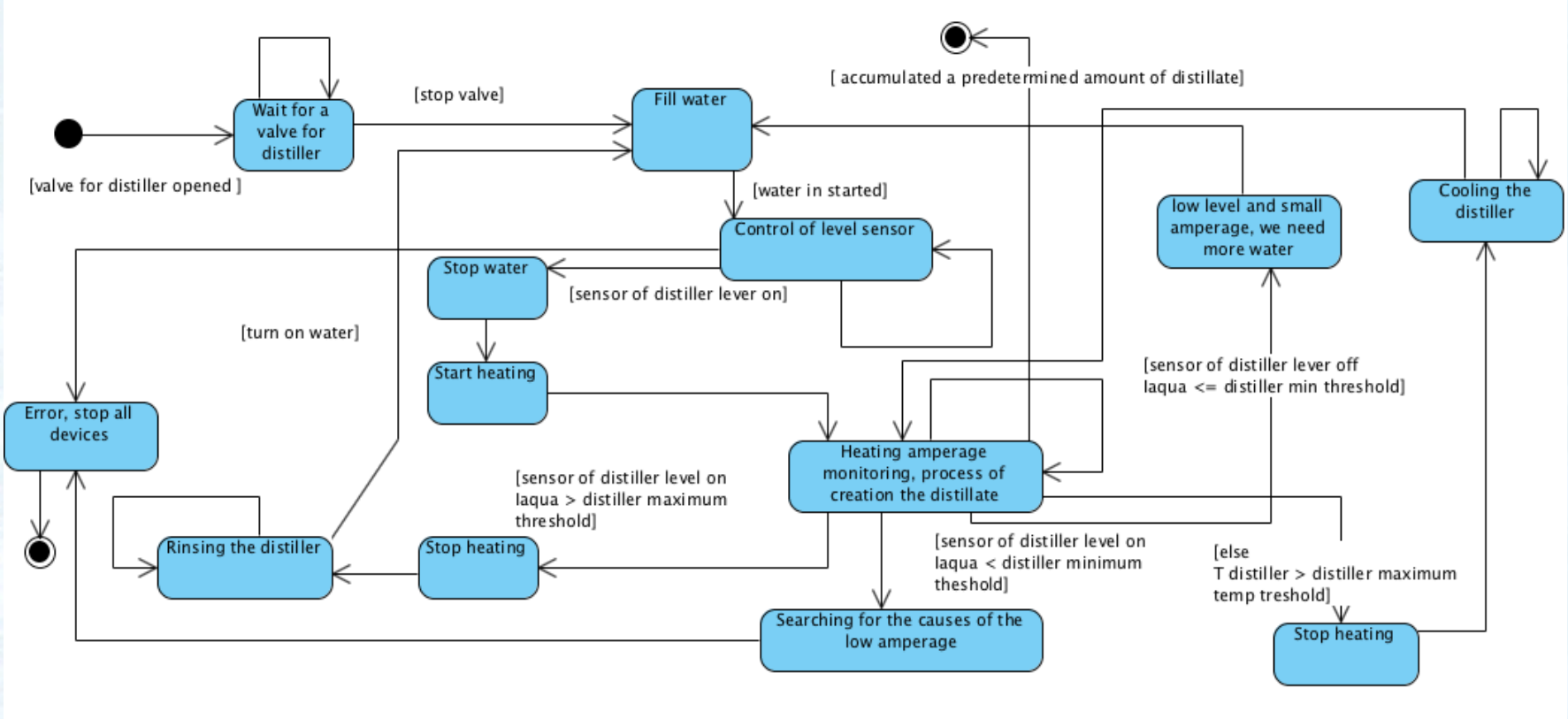
2) The transition to 3.3.

Step 3.1: Draining of salted water ...

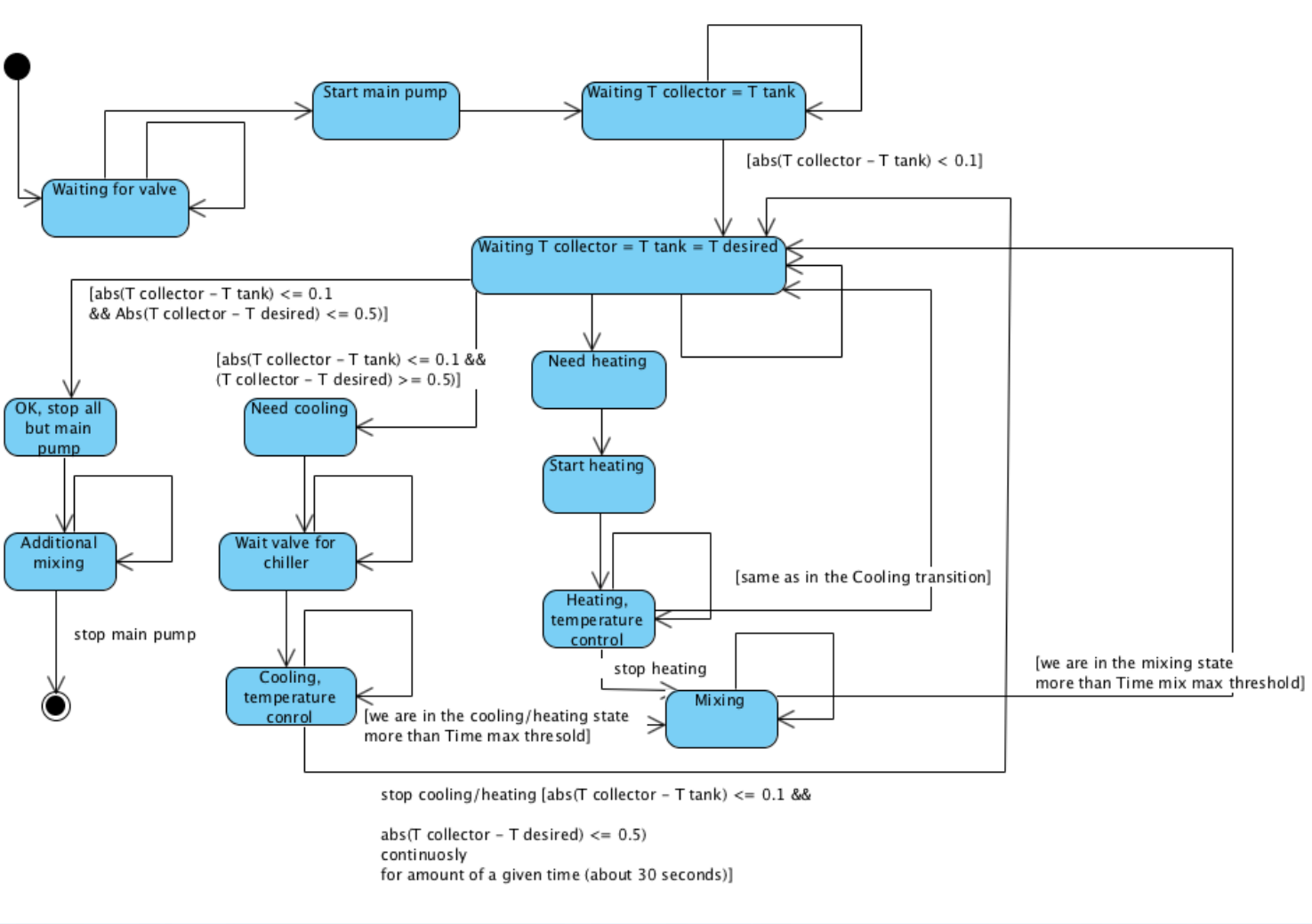
Automata approach

- So we need the automata approach here
- We see steps like the states, transitions, actions and guard conditions
- The process of algorithm design is to draw state machine diagrams from the spec
- The process of developing – just write the code to follow the steps after device layer will be done
- The automatons can also be analyzed and verified

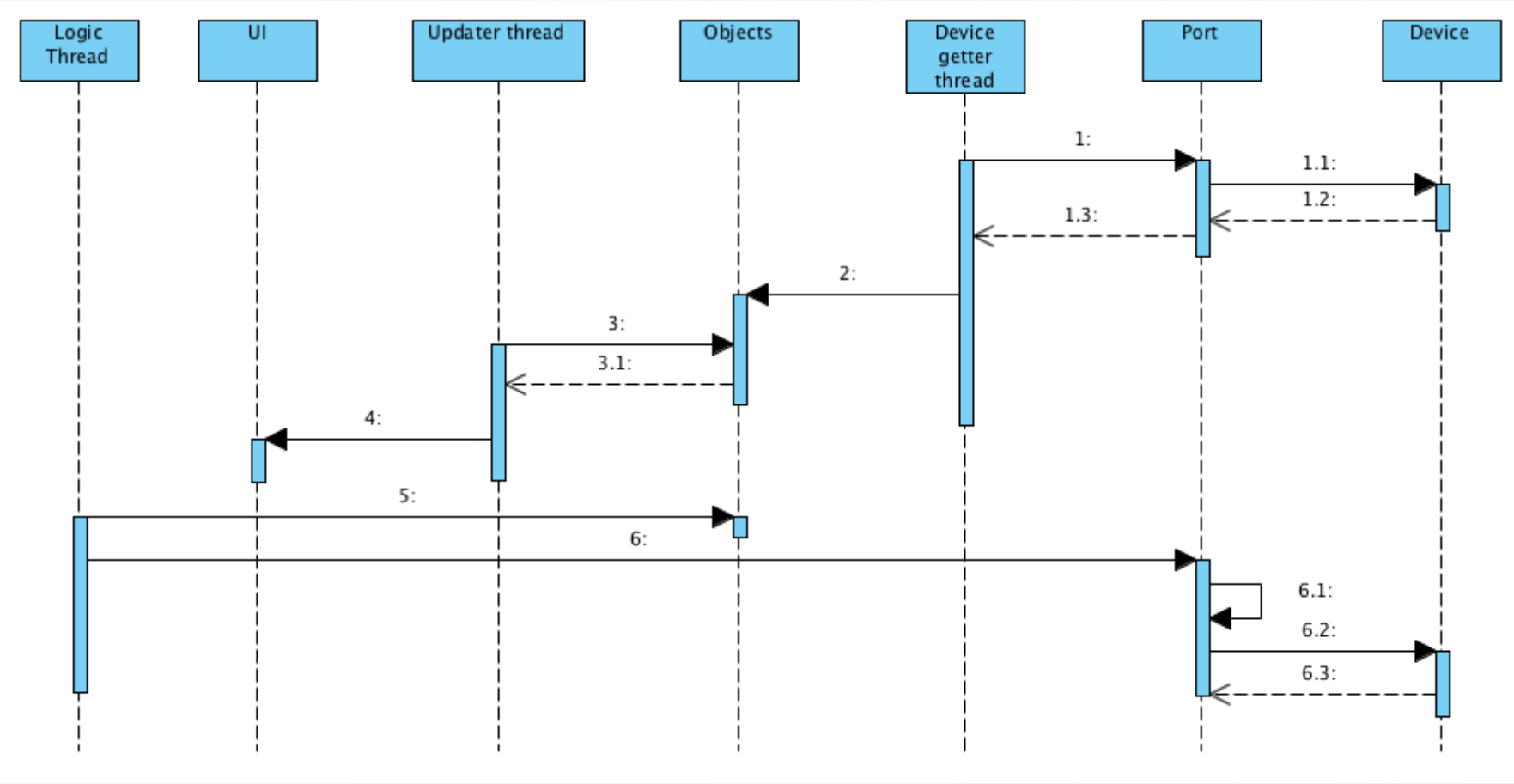
Creating the distiller process



Normalization process



Inside


Logic Thread
UI
Updater thread
Objects
Device getter thread
Port
Device
1:
1.1:
1.2:
1.3:
2:
3:
3.1:
4:
5:
6:
6.1:
6.2:
6.3:

Requirements

- Actions to start/stop devices do not affect the current process of obtaining information
- Check pumping: if we got some level of water in the small tank all the water must be pumped to the storage tank
- Distillate preparing cycle will be completed, or an error message will be displayed
- Distillate normalization cycle will be completed, or an error message will be issued
- Will not fill water above the edge of distiller / tanks
- Pumps do not work without water
- You cannot start a heater / distiller if the level of water in the tank is small
- In each mode, water flows are redirected correctly
- Some devices can be turned on/off after some time after the control impact and the system must work correctly

Ways to satisfy the requirements

- An additional thread for watching to the devices state, control and stop the process
- Additional sensors
- PID controllers
- Preliminary verification of the processes given by its automatons

Verification

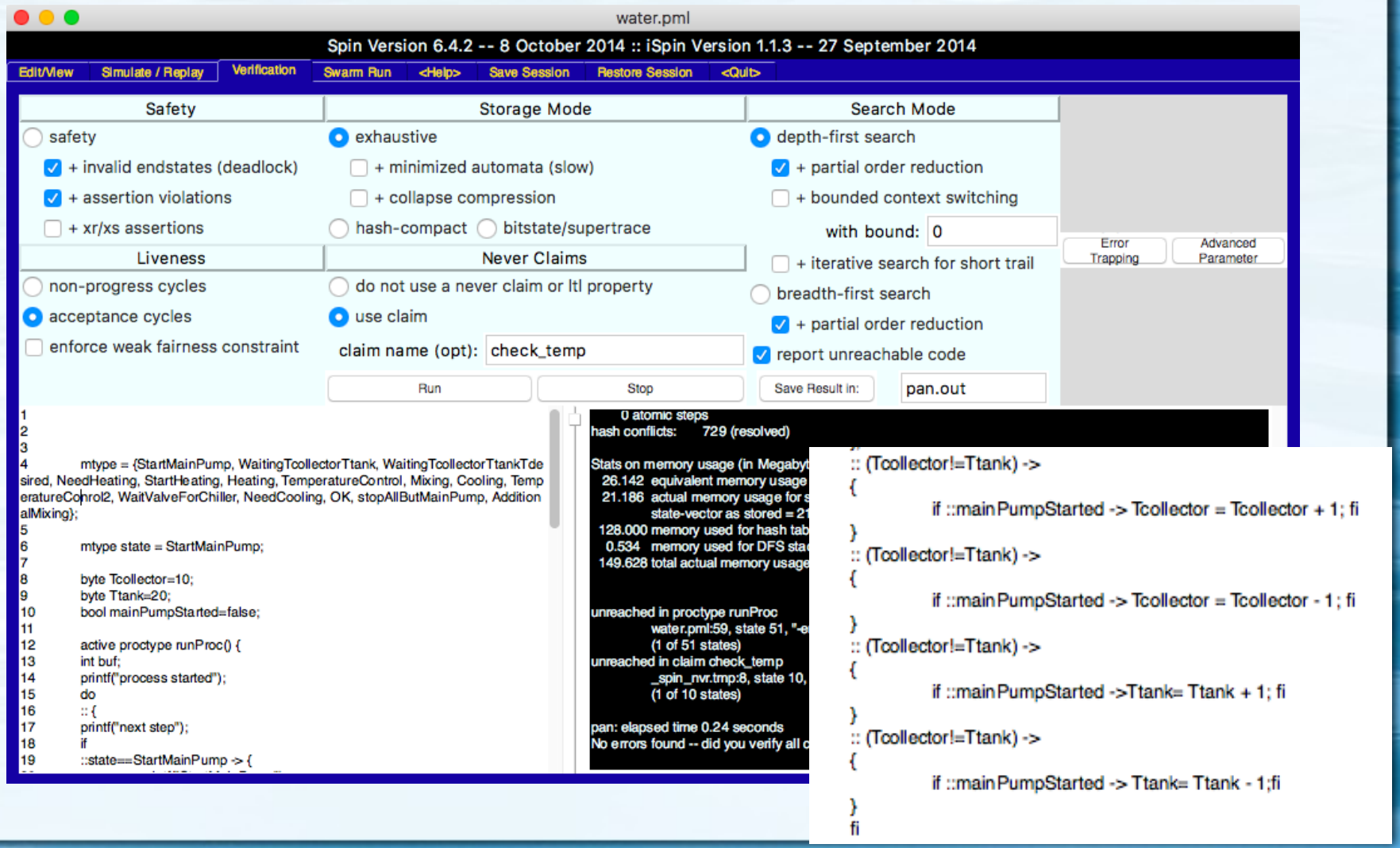
- The automaton model in Promela is received from the spec
- LTLs for reachability of each state

  E (state == <state>)
- LTLs to check additional properties

  i.e. G (state == WaitingTcollectorTtankTdesired) ->

  (Tcollector == Ttank)
- Simulation to show to the customer how the water treatment algorithms are work

Verification



Main issues

1. We cannot simulate the device layer, so we need to test on real devices.

2. It is hard to ask any specification from the customers

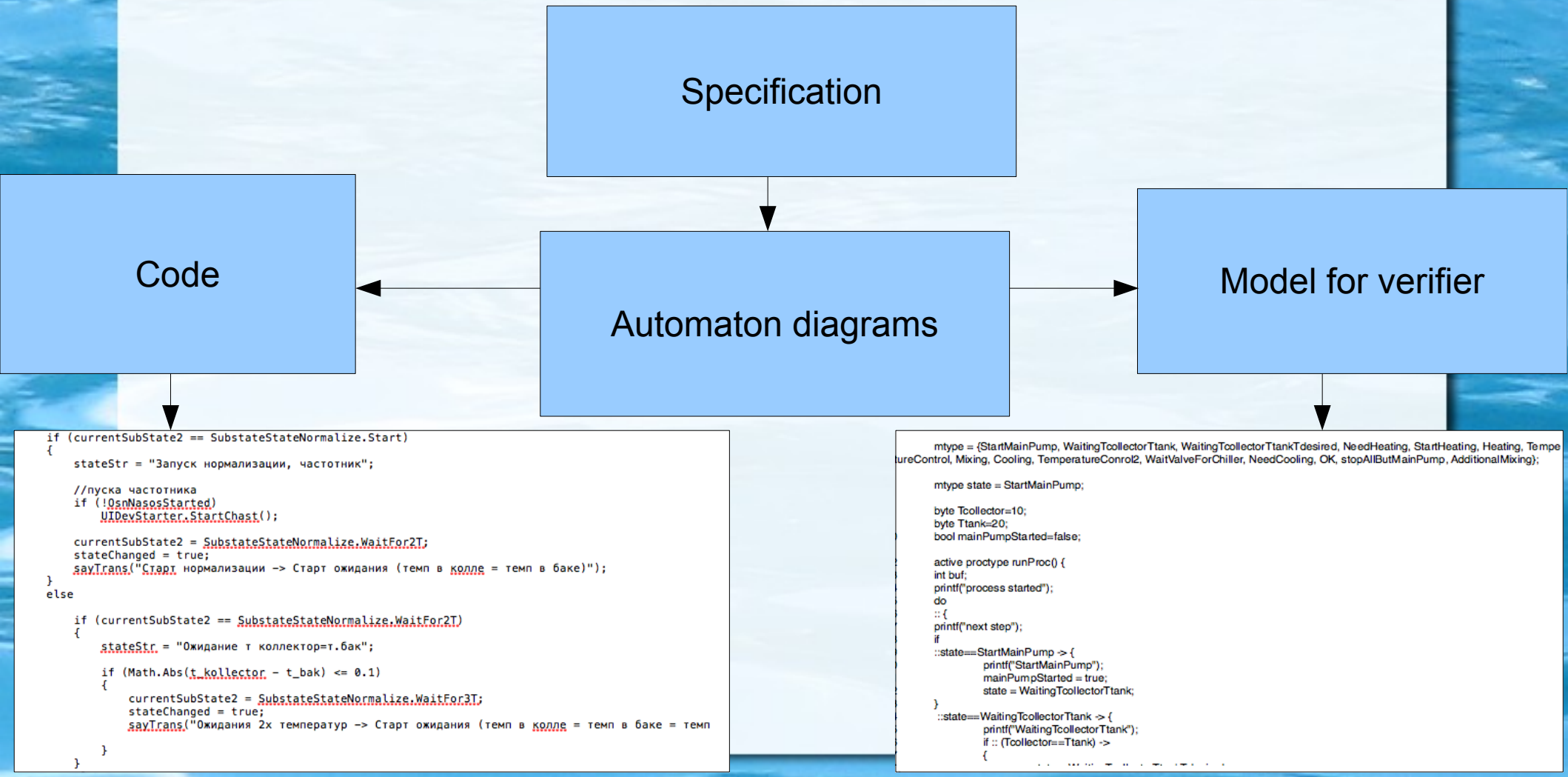
3. The specification of control algorithms was given from the customer only after the overall internal architecture implementation because he is not sure about the functionality of the hardware so it was difficult to estimate the effort and the software cost before starting the project.

4. The specification of control algorithms was changed several times after testing and verification the functionality because of lacks in the specification.

5. Current implementation of system is a user-space application that has not very good response time (~0.5s lack). Is is better to build the control algorithms as modules for a real-time OS.

Some my suggestions to improve the process of developing suchlike control systems based on specification analysis

Automatic code generation




```
if (currentSubState2 == SubstateStateNormalize.Start)
{
    stateStr = "Запуск нормализации, частотник";

    //пуска частотника
    if (!OsnNasosStarted)
        UIDevStarter.StartChast();

    currentSubState2 = SubstateStateNormalize.WaitFor2T;
    stateChanged = true;
    sayTrans("Старт нормализации -> Старт ожидания (темп в колле = темп в баке)");
}
else

    if (currentSubState2 == SubstateStateNormalize.WaitFor2T)
    {
        stateStr = "Ожидание т коллектор=т.бак";

        if (Math.Abs(t_kollector - t_bak) <= 0.1)
        {
            currentSubState2 = SubstateStateNormalize.WaitFor3T;
            stateChanged = true;
            sayTrans("Ожидания 2х температур -> Старт ожидания (темп в колле = темп в баке = темп

        }
    }
```

```
mtype = {StartMainPump, WaitingTcollectorTtank, WaitingTcollectorTtankTdesired, NeedHeating, StartHeating, Heating, Tempe
tureControl, Mixing, Cooling, TemperatureConrol2, WaitValveForChiller, NeedCooling, OK, stopAllButMainPump, AdditionalMixing};

mtype state = StartMainPump;

byte Tcollector=10;
byte Ttank=20;
bool mainPumpStarted=false;

active proctype runProc() {
int buf;
printf("process started");
do
:: {
printf("next step");
if
::state==StartMainPump -> {
        printf("StartMainPump");
        mainPumpStarted = true;
        state = WaitingTcollectorTtank;
}
 ::state==WaitingTcollectorTtank -> {
        printf("WaitingTcollectorTtank");
        if :: (Tcollector==Ttank) ->
        {
```

BDD

- We don't have exactly final specification
- Specification is growing, and the developing processes is continued at the same time
- Behavior driven developing is an industrial approach* for software developing based on scenarios written on Gherkin language

*M.Wynne, A.Hellesoy, S.Tooke. The Cucumber Book, Second Edition. Behaviour-Driven Development for Testers and Developers.

A BDD process

1. Specification of a feature

```
Feature: Software Calculator
  In order to avoid using + - / *
  As a math idiot
  I want to create a software calculator
  Scenario:
    Given I have my software calculator
    When I have entered 10 as first operand
    And I have entered 20 as second operand
    And I press 'Add'
    Then The result should be 30
```

```
Feature: Addition
  In order to avoid using + - / *
  As a math idiot
  I want to create a software calculator

  Scenario:
    Given I have my software calculator
    When I have entered 10 as first operand
    And I have entered 20 as second operand
    And I press 'Add'
    Then The result should be 30
```

Create step definition
Create all steps definition

2. Detect that that feature hasn't been implemented already

3. Generate step definition code in OO language

```
public class MyStepdefs {
    @Given("^I have my software calculator$")
    public void iHaveMySoftwareCalculator() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
    }

    @When("^I have entered (\\d+) as first operand$")
    public void iHaveEnteredAsFirstOperand(int number) throws Throwable {
        // Write code here that turns the phrase above into concrete actions
    }

    @And("^I have entered (\\d+) as second operand$")
    public void iHaveEnteredAsSecondOperand(int number) throws Throwable {
        throw new PendingException();
    }

    @And("^I press 'Add'$")
    public void iPressAdd() throws Throwable {
        throw new PendingException();
    }

    @Then("^The result should be (\\d+)$")
    public void theResultShouldBe(int expected) throws Throwable {
        throw new PendingException();
    }
}
```

```
private Calculator calc;

@Given("^I have
public void iHav
    this.calc = n
}
```

Create class 'Calculator'
Create enum 'Calculator'
Create inner class 'Calculator'
Create interface 'Calculator'
Add Maven Dependency...
Make 'protected'

4. Generate a class

```
@And("^I press 'Add'$")
public void iPressAdd() throws Throwable {
    this.result = calc.add(operand1, operand2);
}

@Then("^The result shoul
public void theResultSho
```

Create method 'add'
Move assignment to field declaration

5. Generate a method Write the code

```
Scenario:
  Given I have my software calculator
  When I have entered 5 as first operand
  And I have entered 10 as second operand
  And I press 'Multiply'
  Then The result should be 50
```

6. Creating the next feature

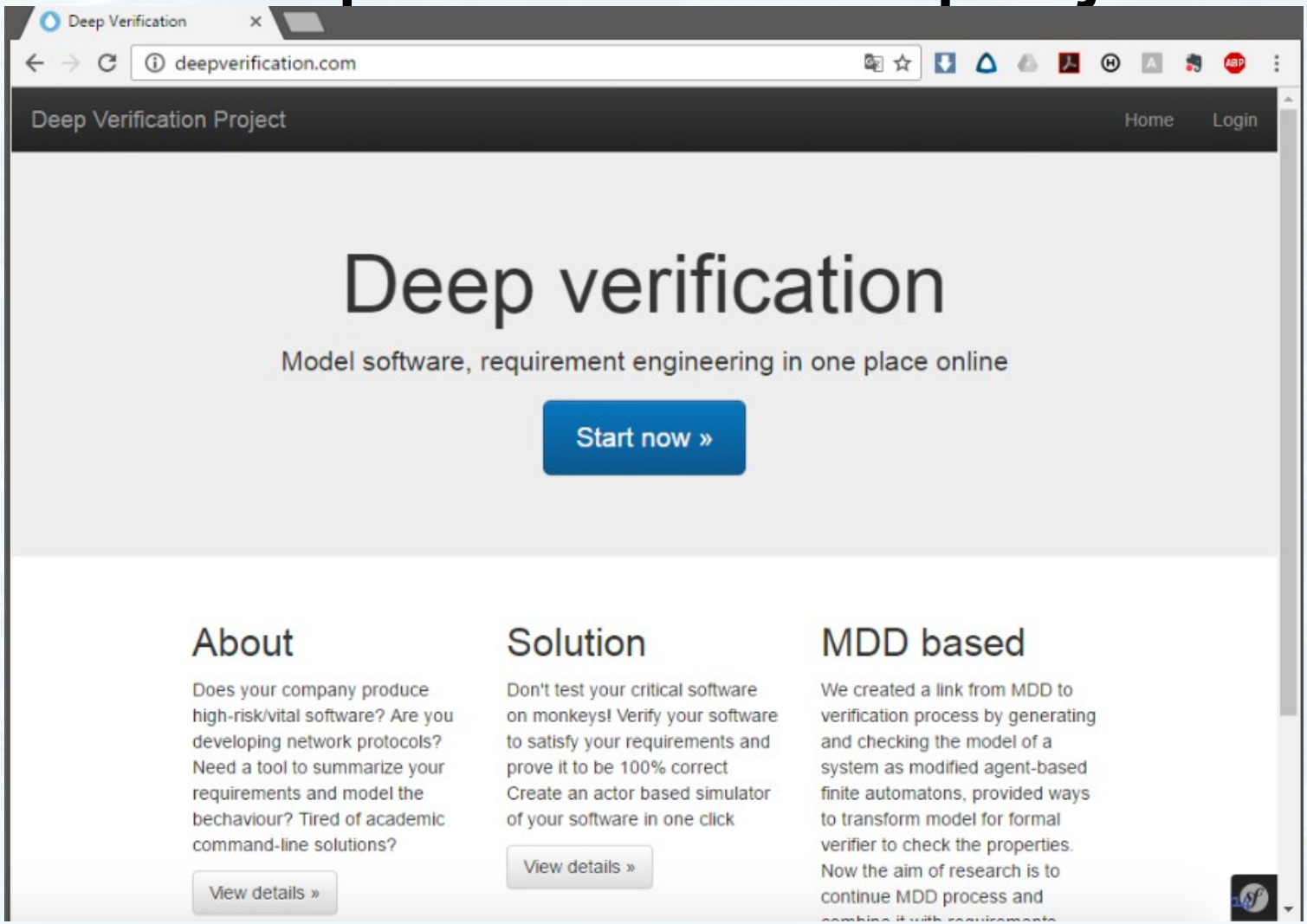
BDD addition

- The BDD language now has no constructs for automaton description
- It is proposed **to add constructs to the BDD for describe the states and transitions and** extend its code-generation classes
- After that the specification in BDD became a formal specification like the specification shown here in the 8th slide
- So the customer (non-programmer) can create this by hand
- In BDD process, when the spec is added/changed it is possible to generate the automaton code and a model for verification

Russian Foundation for Basic Research support

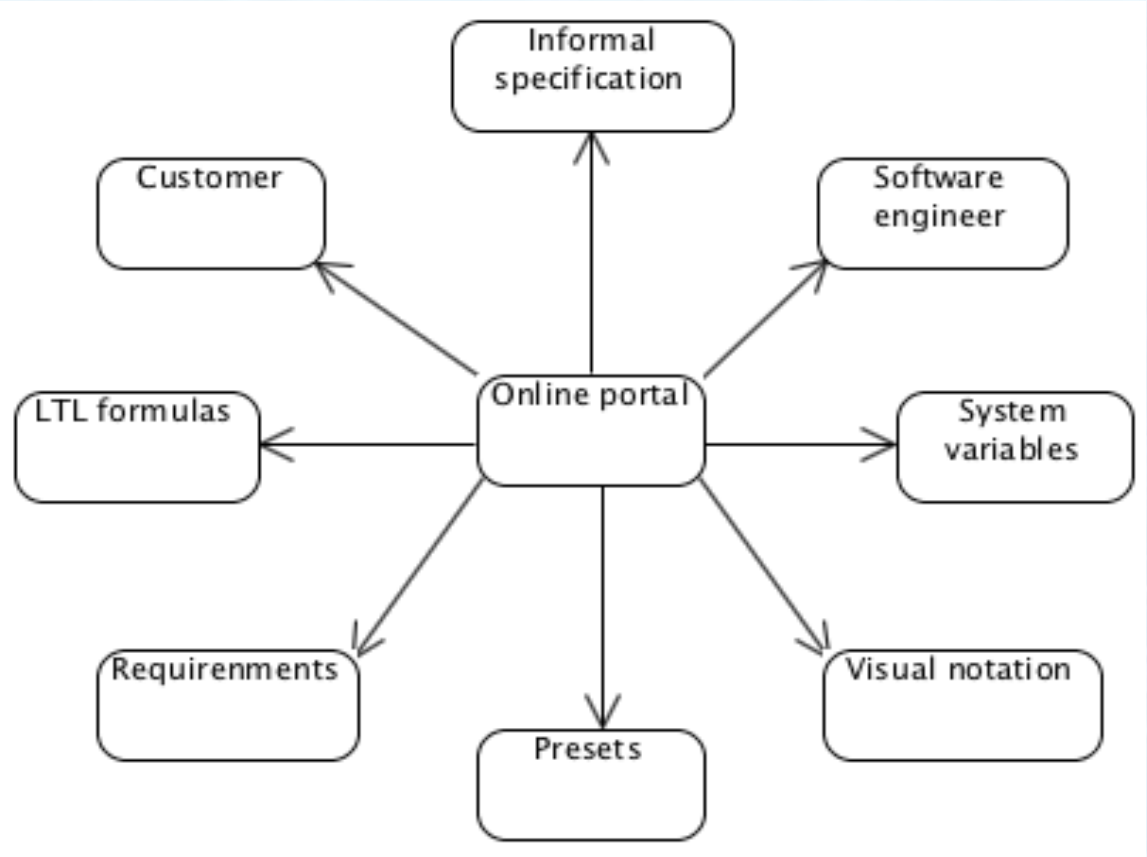
- Grant “Methods for generation of formal models and requirements from technical documentation presented in a natural language and their verification”

A deep verification project

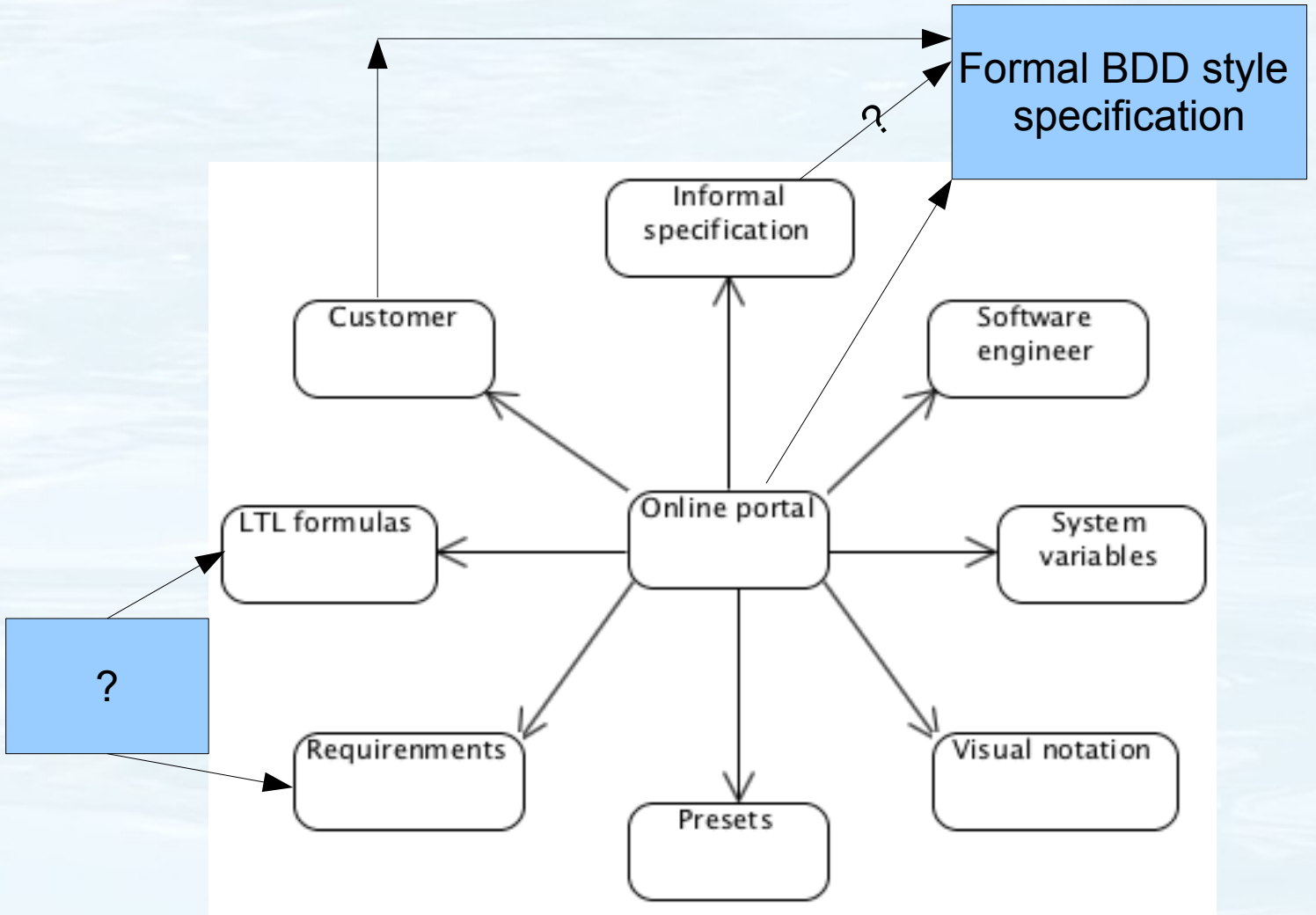


A deep verification project

An Online portal for modeling and requirements engineering of complex distributed software that will provide the links between a **customer**, an **engineer** and **verification tools**



A deep verification project



Design and implementation a software for water purification with using automata approach and specification based analysis Thanks! Q/A

Sergey Staroletov (Polzunov Altai State Technical University)

Eighth Workshop
Program Semantics, Specification and Verification:
Theory and Applications
(PSSV 2017, June 26, 2017)