

# Обзор технологий интеллектуализации ГИС

Сорокин Р.П.

НИЛ ООГИС

SPIIRAS

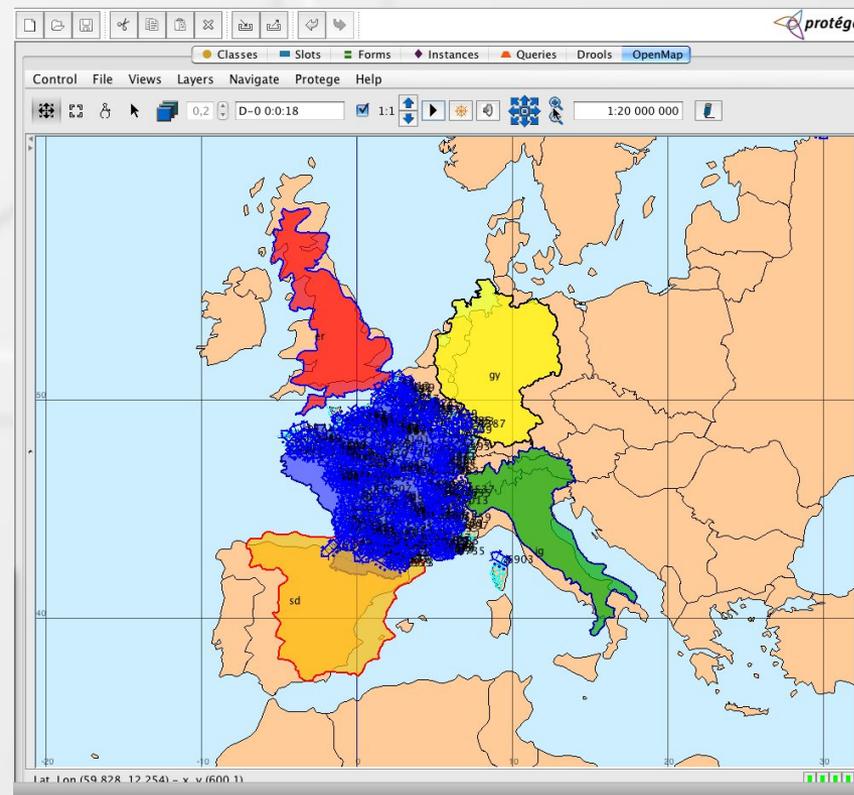
- Интеллектуальные геоинформационные системы
- Сложные пространственные процессы
- Архитектура с открытым исходным кодом
- Онтология сценариев
- Наборы правил
- Реализация
- Дополнительные технологии

SPIIRAS

*Интеллектуальная геоинформационная система (ИГИС)* предназначена для моделирования и изучения сложных пространственных процессов. Наряду с традиционными геоинформационными технологиями она включает в себя технологии искусственного интеллекта для представления и обработки знаний

*Сложный пространственный процесс* – это объединение взаимосвязанных элементарных пространственных процессов таких как перемещение точечных объектов, перемещение и трансформация протяженных объектов в пространстве.

Для описания сложных пространственных процессов используется понятие *Сценария*



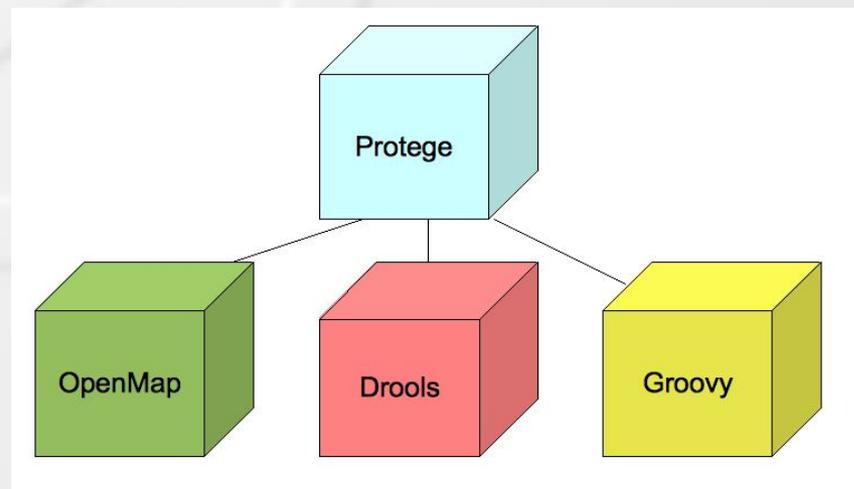
**Protege** – редактор онтологий, используемый для представления знаний из предметной области, включая сценарии моделирования и наборы правил

**OpenMap** – библиотека ГИС, используемая для визуального представления пространственных данных на фоне электронных карт

**Drools** – основанная на правилах машина логического вывода, используемая для интерпретации сценариев и наборов правил

**Groovy** – язык программирования вспомогательных функций

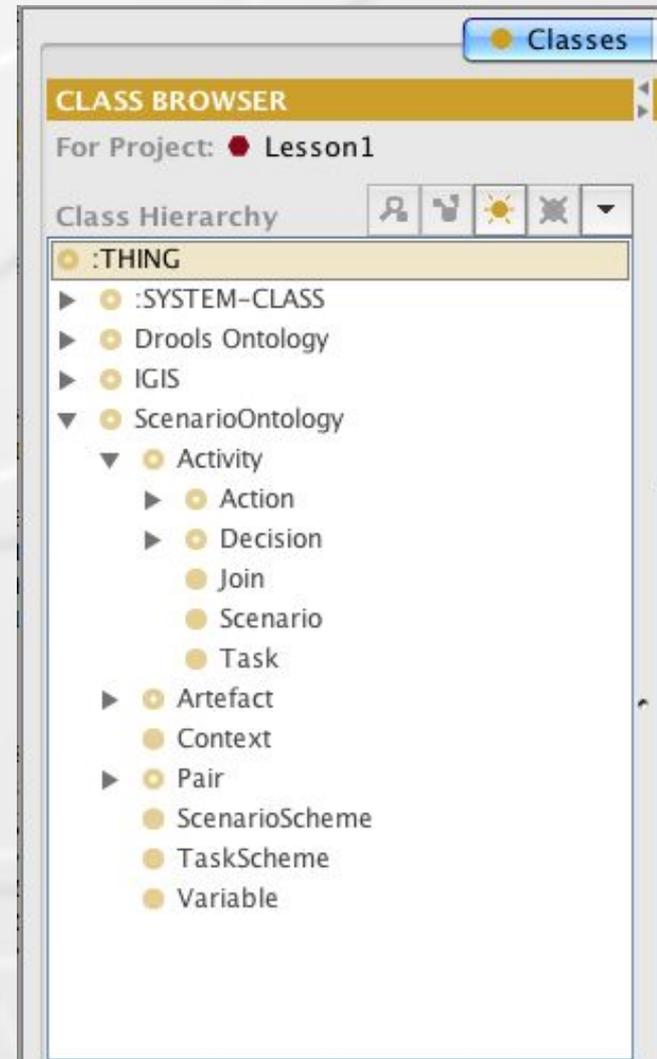
**Clojure** – язык программирования сложных задач



Онтология есть формальное описание понятий предметной области в виде классов, свойств или атрибутов понятий в виде слотов классов и ограничений в виде граней слотов.

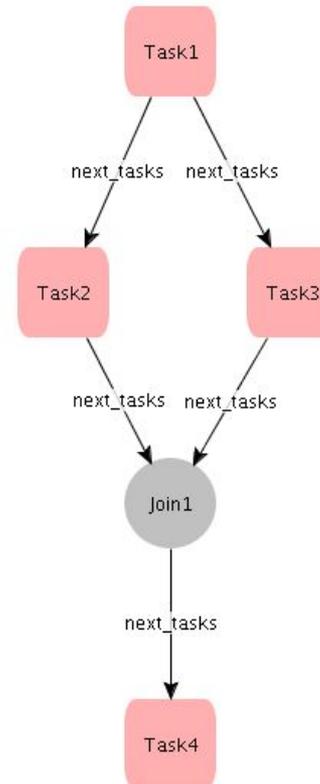
Онтология вместе с множеством индивидуальных представителей (объектов) классов образует *Базу знаний*.

Предметная область моделирования сложных пространственных процессов может быть описана с использованием понятия сценария процесса и других, конкретизирующих его понятий.

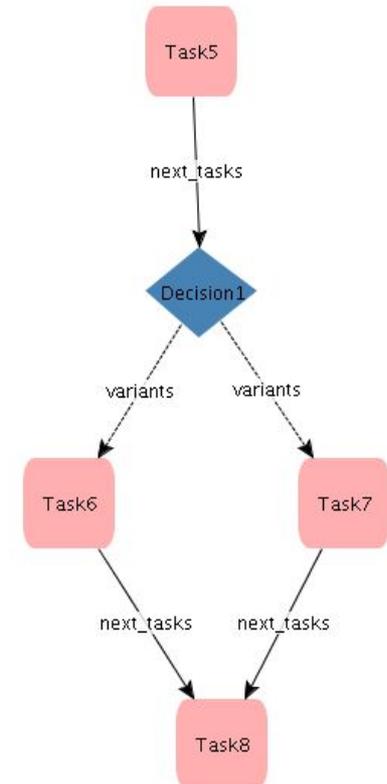


*Сценарий* есть некоторое обобщение понятия алгоритма. Он состоит из блоков задач и блоков принятия решений, связанных линиями потока. *Задачи* в сценарии могут выполняться как последовательно так и параллельно.

Concurrent execution



Sequential execution

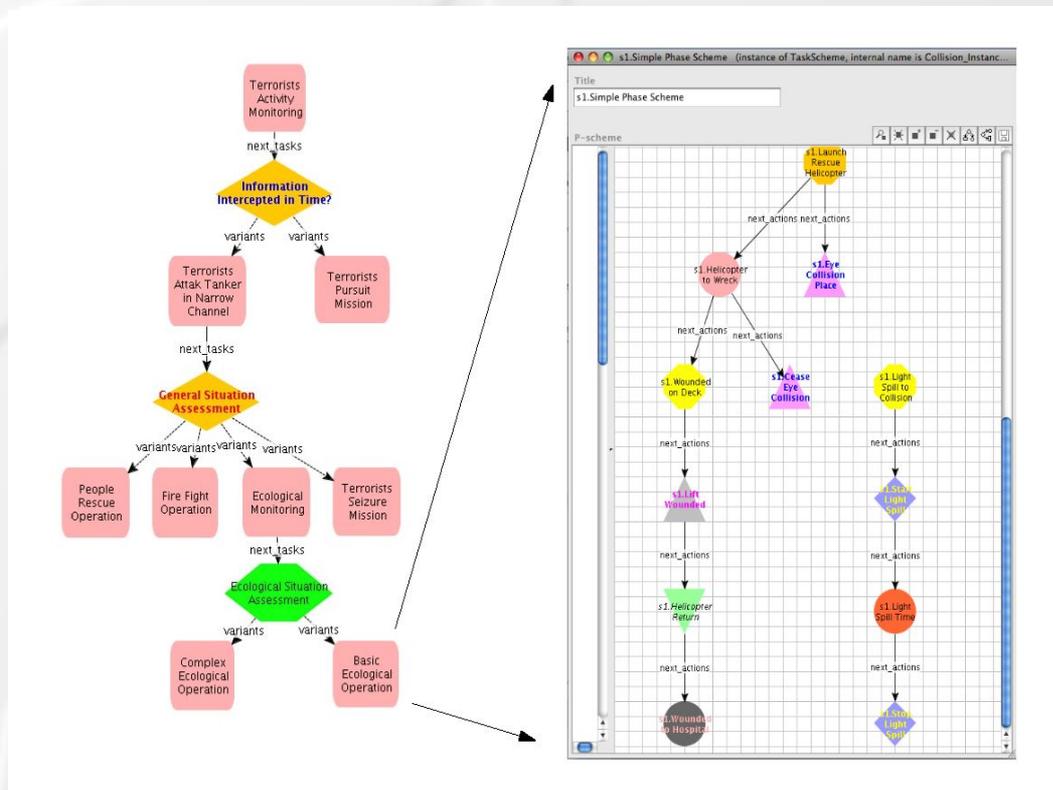


# Задачи, Решения и Действия

*Задача* есть совокупность действий, выполняемых последовательно или параллельно. Действия в схеме задачи также связаны линиями потока управления.

*Решение* есть точка, в которой сценарий может изменить направление своего дальнейшего выполнения в зависимости от некоторого условия и проследовать далее по одной из ветвей.

*Действия* являются элементарными строительными блоками сценариев, они представляют собой конкретную деятельность и могут быть реализованы по-разному. Действия могут иметь свои собственные подсценарии. Таким образом сложные сценарии могут образовывать иерархии из встроенных более простых сценариев.



# Наборы Правил

Наборы правил используются в тех случаях, когда общий алгоритм пространственного процесса заранее не известен, зато могут быть сформулированы сравнительно простые правила, регулирующие его отдельные части.

*Правило* представляет собой структуру из 2х частей – условия и действия. Когда условия выполняются, правило срабатывает и действия выполняются.

## Rule1:

```
when
  Flu infected in town A and
  Flight S takeoff from town A
then
  Flu infected on board S
```

## Rule2:

```
when
  Flu infected on board S and
  Flight S landing in town B
then
  Flu infected in town B
```

## Rule3:

```
when
  Flu infected in town A
then
  Flu infected move on grownd inside dispersing circle R
```

## Rule4:

```
when
  Flu infected move on grownd inside dispersing circle R
  Dispersing circle cover town A
then
  Flu infected in town A
```

Flu on Board (instance of Rule, internal name is Lesson1\_Class4)

Title: Rule1  No-loop  Auto-focus

Saliency:  Dialect: java  Duration:  Agenda-group:

When: Flu( title == "IN\_TOWN", \$town: label )  
Flight( \$flight: title, status == "TAKEOFF", from == \$town )

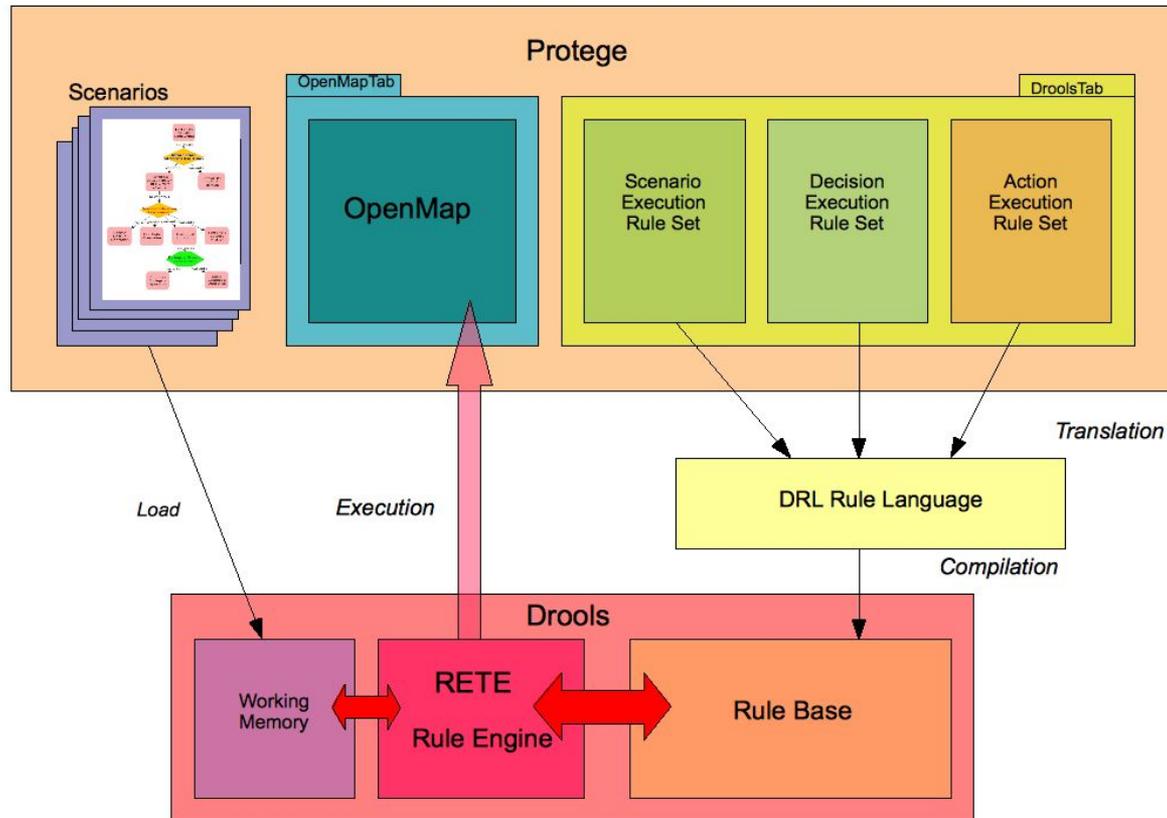
Activation-group:

Date-expires:

Then: Flu flu = new Flu();  
flu.title = "ON\_BOARD";  
flu.label = \$flight;  
insert( flu );

Date-effective:

Единый, основанный на правилах, подход используется для реализации выполнения сценариев и работы наборов правил



## Предпосылки

Большинство реальных действий являются длительными

Компьютерное имитационное моделирование есть дискретный процесс, соответствующий повторению циклов с интервалом  $\Delta t$ , во время которого выполняются вычисления, проверяются условия и моделируются события

## Реализация из 2х правил

*Правило Старта* описывает условия запуска действия и установку начальных значений параметров выполнения действия

*Правило Повторения* описывает, что нужно делать в каждом цикле  $\Delta t$  для выполнения действия, включая проверку условия окончания

```
act: Arrive( status == "START", Sobj: object, $lat: latitude, $lon: longitude, $speed: speed )
```

```
act: Arrive( status == "REPEAT", Sobj: object, $lat: latitude, $lon: longitude, $speed: speed, $rad: radius )
DroClock()
```

```
NavOb obj = (NavOb) runa.getMapOb( Sobj );
if( obj != null )
{
  go( obj, MapOb.getDeg( $lat ), MapOb.getDeg( $lon ), $speed.floatValue() );
  act.status = "REPEAT";
  update( act );
}
else
{
  act.status = "FAILED";
  update( act );
}
```

```
NavOb obj = (NavOb) runa.getMapOb( Sobj );
if( obj != null )
{
  float lat = MapOb.getDeg( $lat );
  float lon = MapOb.getDeg( $lon );
  if ( obj.near( lat, lon, $rad.floatValue() ) || obj.abaft( lat, lon ) )
  {
    obj.setLatitude( $lat );
    obj.setLongitude( $lon );
    obj.setSpeed( (float)0 );
    act.status = "DONE";
    update( act );
  }
  else
    obj.setCourse( (int) obj.bearingsDeg( lat, lon ) );
}
else
{
  act.status = "FAILED";
  update( act );
}
```

Name	Cardinality	Type	Other Facets
latitude	single	String	
longitude	single	String	
next_actions	multiple	Instance of Action	
notice_after	single	Instance of Notice	
notice_before	single	Instance of Notice	
object	single	Instance	
radius	single	Float	
run	single	Integer	
speed	single	Float	
status	single	String	
super_run	single	Integer	
title	single	String	

*Правило Решения* в исполнительной части содержит программу выбора направления продолжения сценария в зависимости от выполнения к данному моменту условия, связанного с данным решением.

The screenshot shows a window titled "TDecision (instance of DecisionClass)". It contains the following elements:

- Name:** TDecision
- Role:** Concrete
- Dialect:** java
- When:**

```
decision : TDecision( status == "START", $variants: variants, $script: script, $run: run, $super_run: super_run )
```
- Then:**

```
Object[] vars = $variants.toArray();
Instance variant = (Instance) vars[ evalTRule( $script, decision.ctxmap, runa ) ];
Activity next = nextForDecision( variant, decision.ctxmap, $run, $super_run, traceLevel );
insert( next );
retract( decision );
```
- Template Slots:** A table listing slots with their cardinality and types.

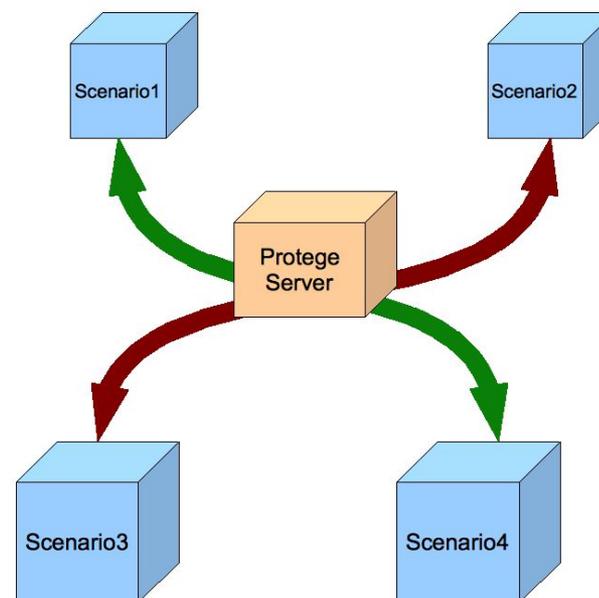
Name	Cardina...	Type	Other Facets
notice_after	single	Instance of Notice	
notice_before	single	Instance of Notice	
run	single	Integer	
script	single	Instance of Script	
status	single	String	
super_run	single	Integer	
title	single	String	
variants	multiple	Instance of Decision ...	

*Сервер Protege* изначально предназначен для удаленного редактирования онтологий, используя Java RMI.

Разработана простая онтология обмена сигналами, включающая класс *Signal*. Она хранится на сервере Protege.

Для отправки сигнала на сервер посылается новый представитель класса *Signal*.

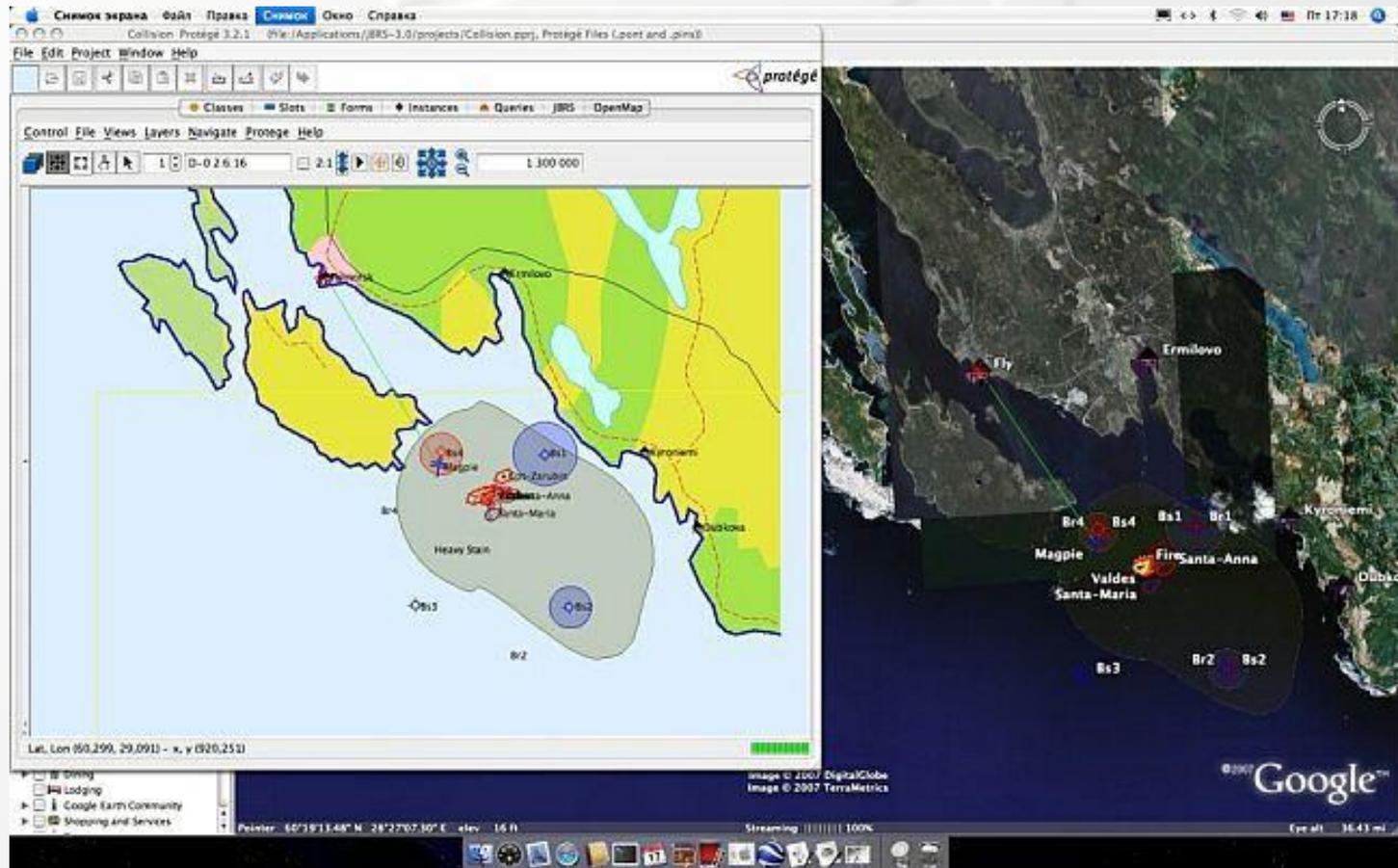
Для получения сигнала другой сценарий читает и удаляет его с сервера.



# Дополнительные Технологии. Google Earth

\*

Для отображения моделирования на фоне программы Google Earth разработан KML сервер, основанный на технологии микро web сервера Jetty





Все разработанные и использованные технологии являются бесплатными с открытым исходным кодом и могут быть свободно скачены из Интернет

Начальный адрес: <http://oogis.ru/droolstab/>

SPIIRAS