VIII International conference on Information Technology and Nanotechnology

23 – 27 May

Samara, Russia

Development of methods, algorithms and program tools for analysis and design of space missions for Earth observation

Atanas Marnov Atanassov

Space Research and Technology Institute
Bulgarian Academy of Sciences

At_M_Atanassov@yahoo.com

The goal of Earth observation satellites' mission is to obtain ground information by satellite-borne sensors, to aid in atmospheric research, meteorology, defence, and commercial applications.

The preparation and implementation of satellite missions go through stages during which the use of software tools for analysis and design is extremely important. In the beginning, the realism of the ideas underlying the mission is checked, after which the possibilities for implementation are studied, various parameters are optimized, algorithms and programs for management and planning are tested.

Contemporary trends in space missions related to Earth observations

Until now/
До скорого

<u>Monolithic systems</u>,
highly integrated, designed to fulfill a set of goals satisfying certain user needs

From now onwards/впредь

<u>Size/masse evolution</u>

- large >1000
- medium 500-1000
- mini 100-500
- micro 10-100
- nano 1-10
- pico 0.1-1
- femto <0.1
- chip

<u>Distributed missions</u>

- satellite [mega-, large-] constellations
- formation flying
- swarm
- partitioned/fractionated satellite
- federated systems

Development of calculation instruments

- Parallel integrator of ordinary differential equations
- Parallel situational analysis solver
- Program model for parallelization of parallel solvers (Union of pools of threads)
- Other tools/solvers are under development too …

Selection optimal integration method

(between methods Runge-Kutta-Fehlberg with error control)

$$y_i = y_{0,i} + h.\sum_{k=0}^{s} c_k g_{k,i} + O_i(h^p) \quad \bar{y}_i = y_{0,i} + \sum_{k=0}^{s+l} \bar{c}_k g_{k,i} + O_i(h^{p+1})$$

$$\bar{O}_i = y_i - \bar{y}_i \approx \sum_{k=0}^{s+l} (c_k - \bar{c}_k)$$ – evaluation of the main member of the error

a) $\bar{O} > \varepsilon.\Delta r$, $\Delta r = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, a **ε** – value of the relative tolerance/error; – a more precise method is selected for (to test if it is enough), the value $c_p.\bar{O}$ is saved; the values of coefficients c_p are determined empirically: c_2=0.005, c_4=0.009, c_5=0.01, c_7=0.02.

b) $c_p.\bar{O} < \varepsilon.\Delta r$, a lower order method is tested.

Used methods – 2/3, 3/4, 4/5, 5/6, 7/8 order

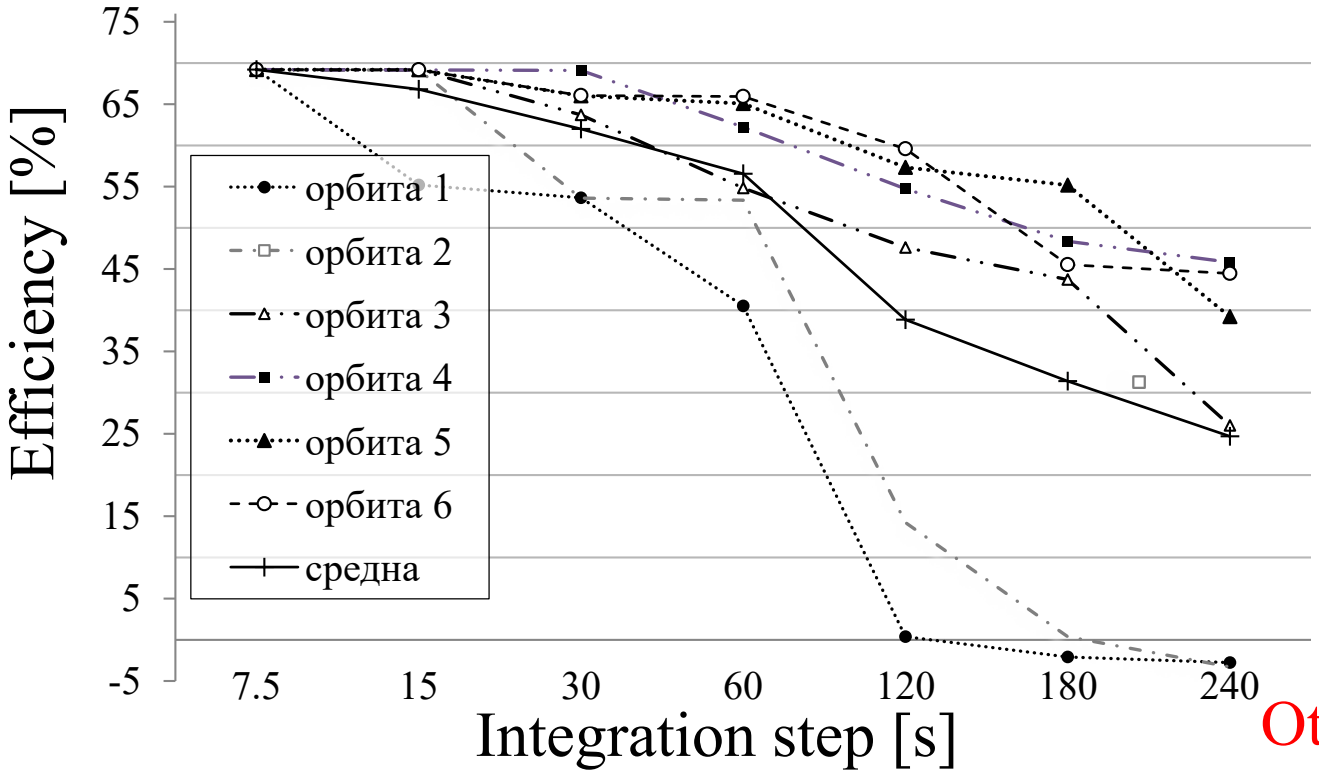
LF Shampine use only two schemes!

Newer, higher-order methods (8, 9, 10) are known, as well as schemes for second-order equations!!!

Efficiency evaluation

$$E = \frac{n_{7/8} - n_{\sim}}{n_{7/8}} . 100$$

orbits	semy-major axes [m]	eccentriciy
1	6 800 000	0.01
2	7 500 000	0.10
3	8 500 000	0.20
4	10 000 000	0.30
5	15 000 000	0.50
6	18 000 000	0.50



$n_{7/8}, n_{\sim}$ – number of calculations of the right hand of the motion equations

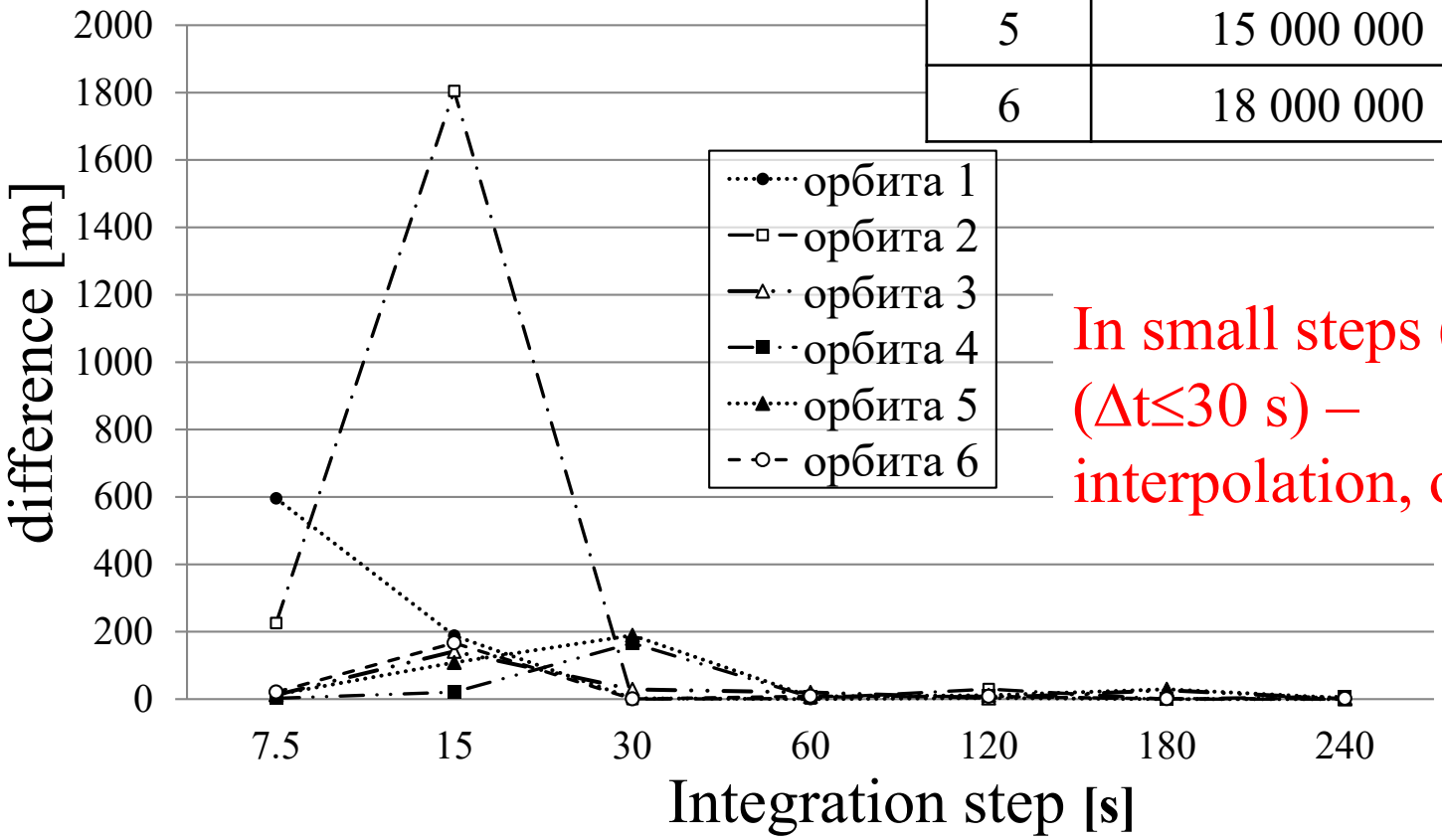
Other orbits – other performance

Stability of the solution

(integration interval - 1 day)

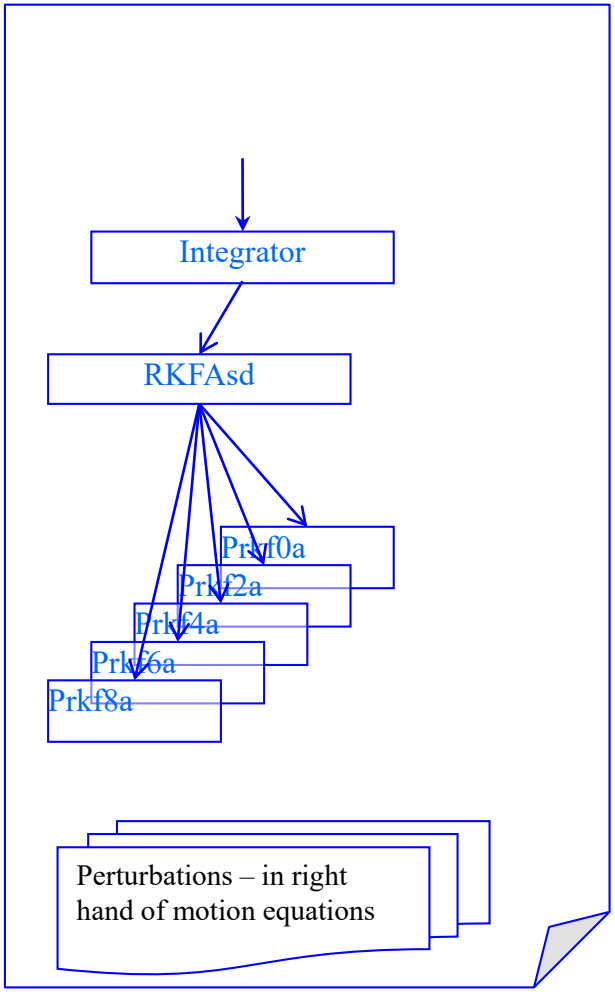
Two solutions are compared; the first is solved with method of 7/8 order and the second- with optimal method selection at the end of the interval

orbits	semy-major axes [м]	eccentriciy
1	6 800 000	0.01
2	7 500 000	0.10
3	8 500 000	0.20
4	10 000 000	0.30
5	15 000 000	0.50
6	18 000 000	0.50



In small steps (for low orbits) ($\Delta t \leq 30$ s) – interpolation, or less tolerance

Integrator's block scheme



Definition of „ situation analysis“

Each situational task can be represented by a predicate function S:

$$SP \Leftrightarrow SP\left(\vec{R}, \{\alpha\}, \{\beta\}, t\right) = \overline{0,1}\ .$$

$\{\vec{R}\} = \langle \vec{r}_1(t), \vec{r}_2(t), \dots \vec{r}_n(t) \rangle$ - set of radius vectors of objects in the model space

$\{\alpha\}$ - sets of vector and/or scalar fields describing the properties of the model space и $\{\beta\}$ – set of constraints

Predicate functions $\{\gamma_i\}$ correspond to the situational conditions $\{\boldsymbol{sc_i}\}$

$$SP = \boldsymbol{sc}_1(t) \wedge \boldsymbol{sc}_2(t) \wedge \dots \wedge \boldsymbol{sc}_n(t) = 1 \quad SP = (\dots(\boldsymbol{sc}_1(t) \wedge \boldsymbol{sc}_2(t)) \wedge \dots) \wedge \boldsymbol{sc}_n(t) = 1$$

Horner’s presentation

Examples for situational conditions

- A satellite passes through the radio visibility zone of the ground station
- A satellite passes over a circular/rectangular area of the earth's surface
- A satellite passes through Earth's shadow or sunlit zone
- A satellite passes over an illuminated / unilluminated part of the earth's surface
- The Moon is over the horizon relatively to the satellite
- Two satellites pass "one above the other" in the time interval $\Delta\tau$ characterizing the phenomenon under study
- Two satellites move "one above the other" over the long part of the trajectories, and the time difference is in the time interval $\Delta\tau$ characterizing the phenomenon under study
- Visibility between two satellites
- The distance between two satellites should be within certain limits or visibility.
- And other

Situational conditions description model

```
MODULE  lib_module

type SitCond
        ! general parameters to each situation conditions
  union
    map
        ! specific parameters about particular situation condition
    end map
…
    map
        ! specific parameters about particular situation condition
    end map
  end union
end type SitCond
…
END MODULE  lib_module
```

Situational task description model

```
MODULE  lib_module
…
type sit_task
  union
    map
      ! parameters specific for each separate situation problem
    end map
    map
      type (SitCond) sit_cond
    end map
  end union
end type sit_task
…
END MODULE  lib_module
```

Situational problems solver
(processor)

A tool for solving a large number of diverse situational tasks, each with more than one condition.

We can consider this tool as a finite state machine, which passes through different states depending on the input strip, which contains information (code) about the type of situational conditions.

The situational analysis solver checks the code of the next condition within the situational task and turns to the corresponding logical function to check the condition.

Part of the subroutine for pool control

```
DO WHILE(.true.);
   k= WaitForSingleObject(ha_beg,WAIT_INFINITE)
  DO WHILE(glb_counter.LT. num_tasks)
      k= WaitForSingleObject(loc_ha_1,WAIT_INFINITE);
    obj_remain= num_tasks-glb_counter
     IF(obj_remain.GT.granule.AND.granule.GT.1) THEN
        loc_counte0= glb_counter+1;
        glb_counter = glb_counter + granule
                    k= SetEvent(loc_ha_1)
       DO loc_counter=loc_counte0,loc_counte0 + granule - 1
           IF(loc_counter.GT. num_tasks) EXIT
              loc_adr= adr + (loc_counter-1)*adr_len;
             CALL Solver(loc_counter,task_descriptor_adr,num_tasks,th_id_num,RHFun)
        END DO
                                                    ELSE ! fine granularity
       glb_counter= glb_counter + 1;
       loc_counter= glb_counter;
         k= SetEvent(loc_ha_1)          ! another thread can take the next subtask
          IF(loc_counter.GT. num_tasks) EXIT
             loc_adr= adr + (loc_counter-1)*adr_len;
            CALL Solver(loc_counter,task_descriptor_adr,num_tasks,th_id_num,RHFun)
           ENDIF
  END DO;
    k= ResetEvent(ha_beg)
    k= SetEvent(ha_end)
END DO;
```

Representation of the situational tasks descriptor

A two-dimensional (allocatable, dynamic) array is used to represent the situational task descriptor in the computer's memory.

The elements of each column contain a description of one situational task. The first (zero) element of the column contains parameters common to the task: serial number of the task; number of situational task conditions; start, end and duration of the time interval when all conditions are met; code of the type of optimization applied, and others...

```
USE  lib_module,only: sit_task
 …
type(sit_task)  sit_prob( 0 : max_num_cond , num_prob)    ! max_num_cond -
                                                          ! количество условий
```

Sources of irregular calculations

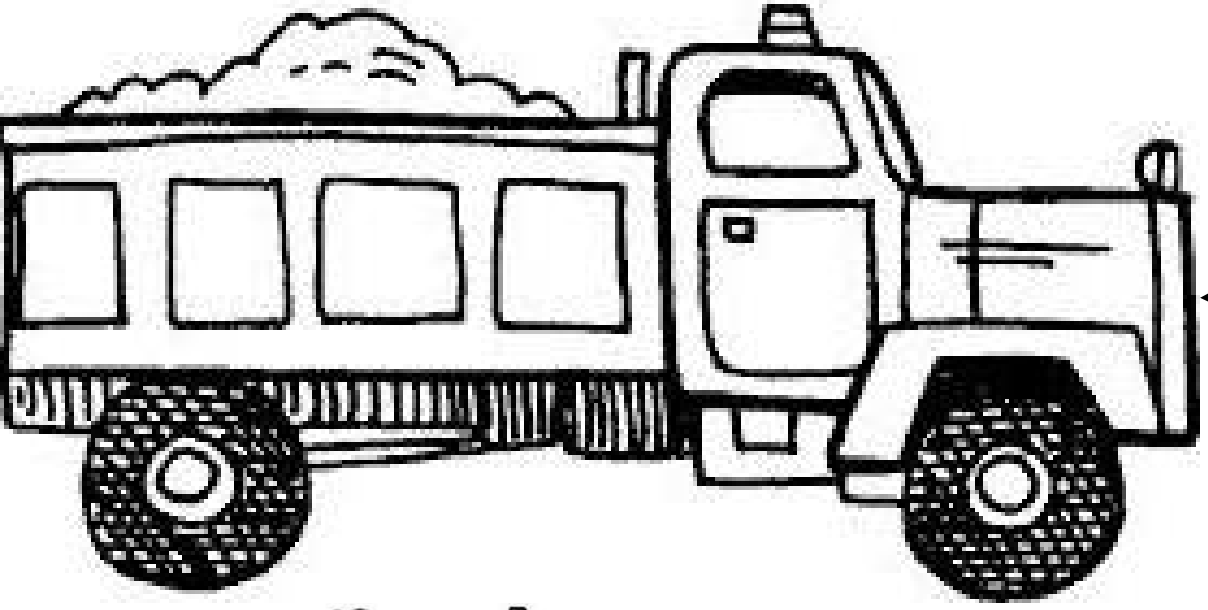
- Applying various integration schemes (automatic selection) depending on the parameters of the orbits for different objects, as well as to varying parts of the orbits for each object;
- Different motion models for each object;
- Different number and type of situational conditions for each situational task;
- Various schemes for solving situational problems depending on the geometric and physical parameters and restrictions (situational conditions), as well as the location in orbit.

Reasons for the use of parallel algorithms and calculations (common to all computing tools)

- Simultaneous integration of the orbits of many satellites and space debris;
- Complex motion models (more perturbations);
- A large number of situational problems are solved with situational conditions that are difficult to calculate;
- The analyzes are performed for long time intervals;
- The simulation is repeated many times with different parameters.

Irregular problems

Loading of Tracks' is irregular problem because of their different sizes.

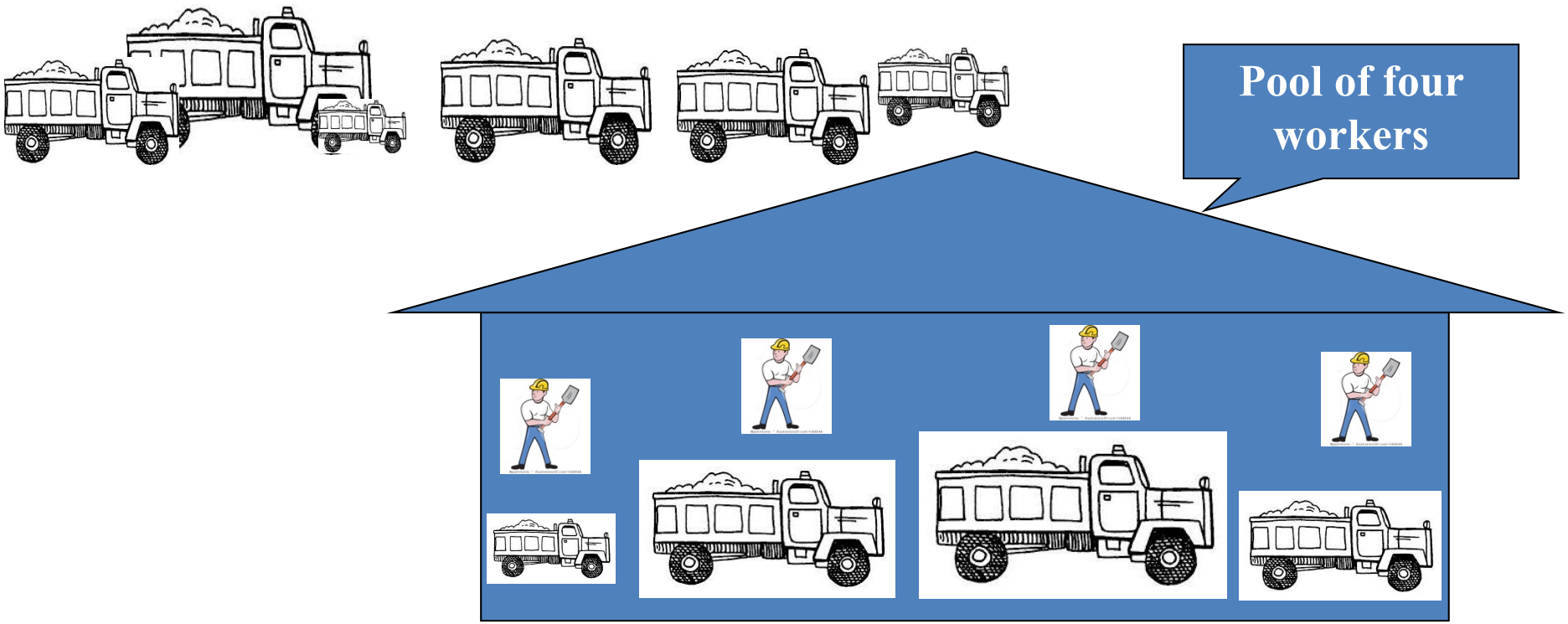


One big task

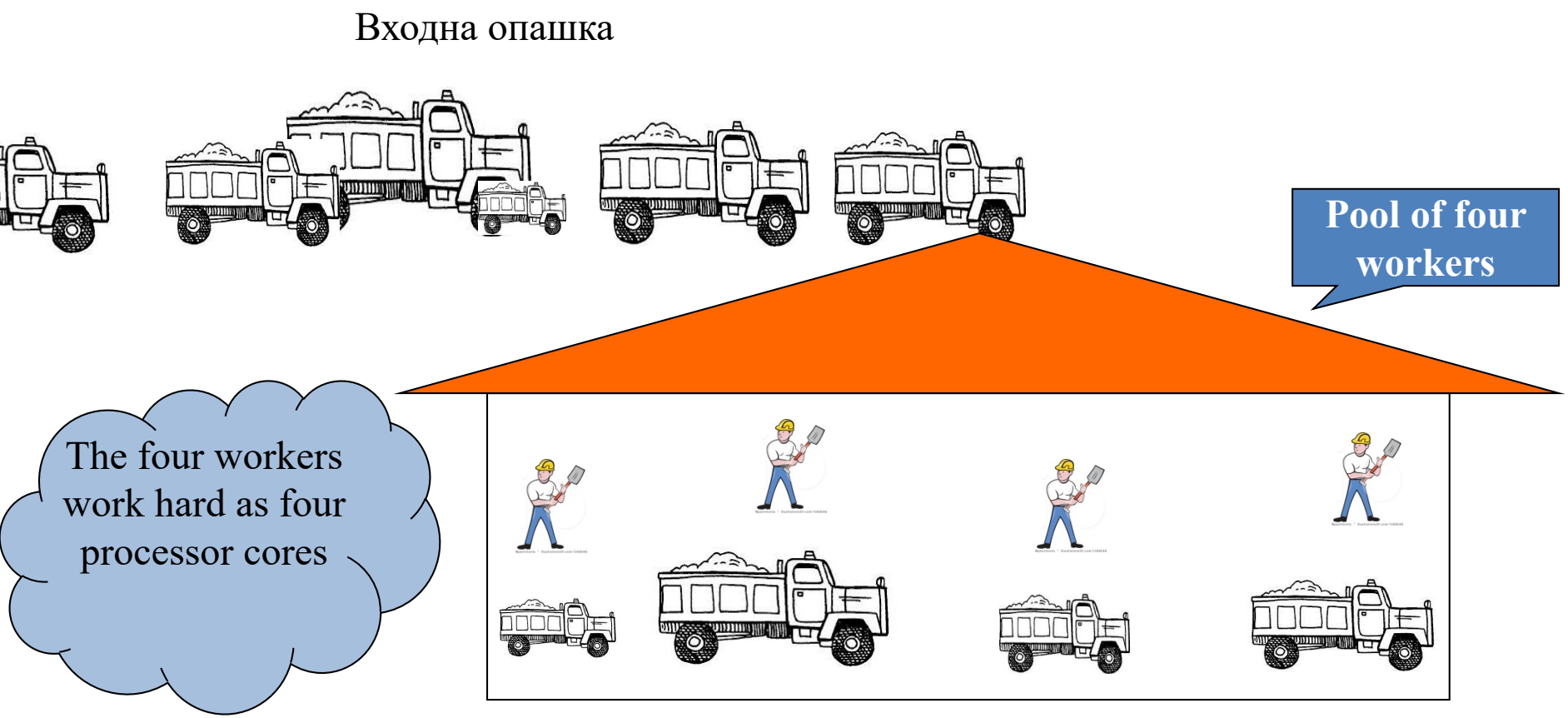
Pool of many little tasks

Conception about “Pool of threads” model

Input queue (from subtask)



Conception about “Pool of threads” model



Tools are needed to synchronize between workers and trucks!!!

Parallel calculation tool creation

Subroutine for creating a pool of threads

Name of buffer subroutine

```
CALL  CreatePoolThreads(SatelliteIntegrator, &
                num_threads,     thread_par,        ha_1)
```

Number of threads

Pool parameters

threads synchro-nization event

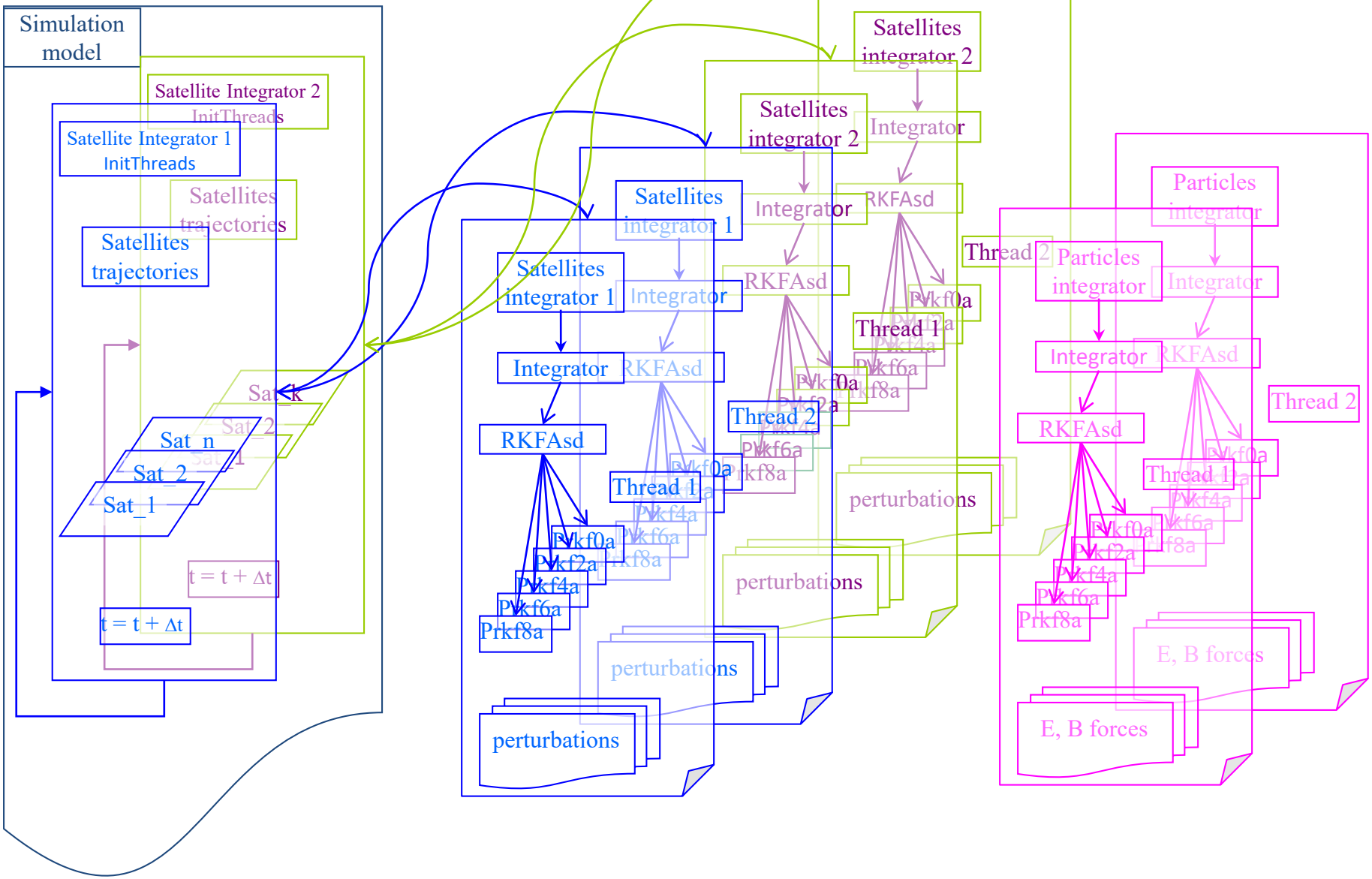
```
integer      thread_par(4, num_int_threads)
        …
thread_par(2,i)= i   ! i=1, num_int_threads
thread_par(1,i)= CreateThread(security,stack,BufferSubroutineName,LOC(thread_par(2,i)), &
                                                CREATE_SUSPENDED,thread_id)
thread_par(3,i)= CreateEvent (security1,.false.,.true.,0)
thread_par(4,i)= CreateEvent (security1,.true.,.false.,0)
```

Data transmission to objects

```
CALL  Data_AI(work_adr,transfer_adr,AI_GlbCount_addr, &
              NumObj,LOC(t),LOC(dt), &
              xvn_adr,xvk_adr,eps_adr, &
              adr_perturbations,len_Grv_model);
CALL  Preparation_AI(NumThreads, Granularity, &
                     PoolParam_addr,AI_ha);
```

The subroutines **Data_AI** and **Preparation_AI** load control data, parameters and addresses in common area, which is contained in buffer subroutine

Functional scheme of actual (parallel) integrator satellite motion equations



Polyformism (1)

```
MODULE RN
…
 type        task_descriptor_template_integrator
  integer        num_obj,num_equestions
  integer        adr1,adr2
  integer        adr_Grv_model,len_Grv_model
  character        izbor*1
 end type   task_descriptor_template_integrator
!
 type        task_descriptor_template_situations
  integer        num_sat
  integer        t_adr,dt_adr,xvn_adr,xvk_adr
  integer        max_num_sit,num_sit_prob
  integer        sci_problem_adr,len_sci_task
  integer        TrajectParam_adr,TrajectParam_len
 end type   task_descriptor_template_situations
…
```

b). Потребителски тип / клас за **АПИСОДУ**

c). Потребителски тип / клас за **ППСА**

Polyformism (2)

(name definition)

```
MODULE RN
...
 INTERFACE      UPC
  SUBROUTINE  Pool_threads_Control_1(th_id_num,num_threads,ha_1,adr_glb_counter, &
                  hread_par_local,lgranul,rkfasd_UPC,task_descriptor_adr,numobj,pertur)
   external      rkfasd_UPC,pertur
   integer th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local(4),lgranul
   integer            task_descriptor_adr,numobj
  END SUBROUTINE  Pool_threads_Control_1

  SUBROUTINE    Pool_threads_Control_2(th_id_num,num_threads,ha_1,adr_glb_counter, &
                     hread_par_local,lgranul,Psitanal_UPC,task_descriptor_adr,num_sit_prob)
   external          Psitanal_UPC
   integer       th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local(4),lgranul
   integer           task_descriptor_adr,num_sit_prob
  END SUBROUTINE Pool_threads_Control_2
 END INTERFACE  UPC
...
```

а). дефиниция на полиморфизъм

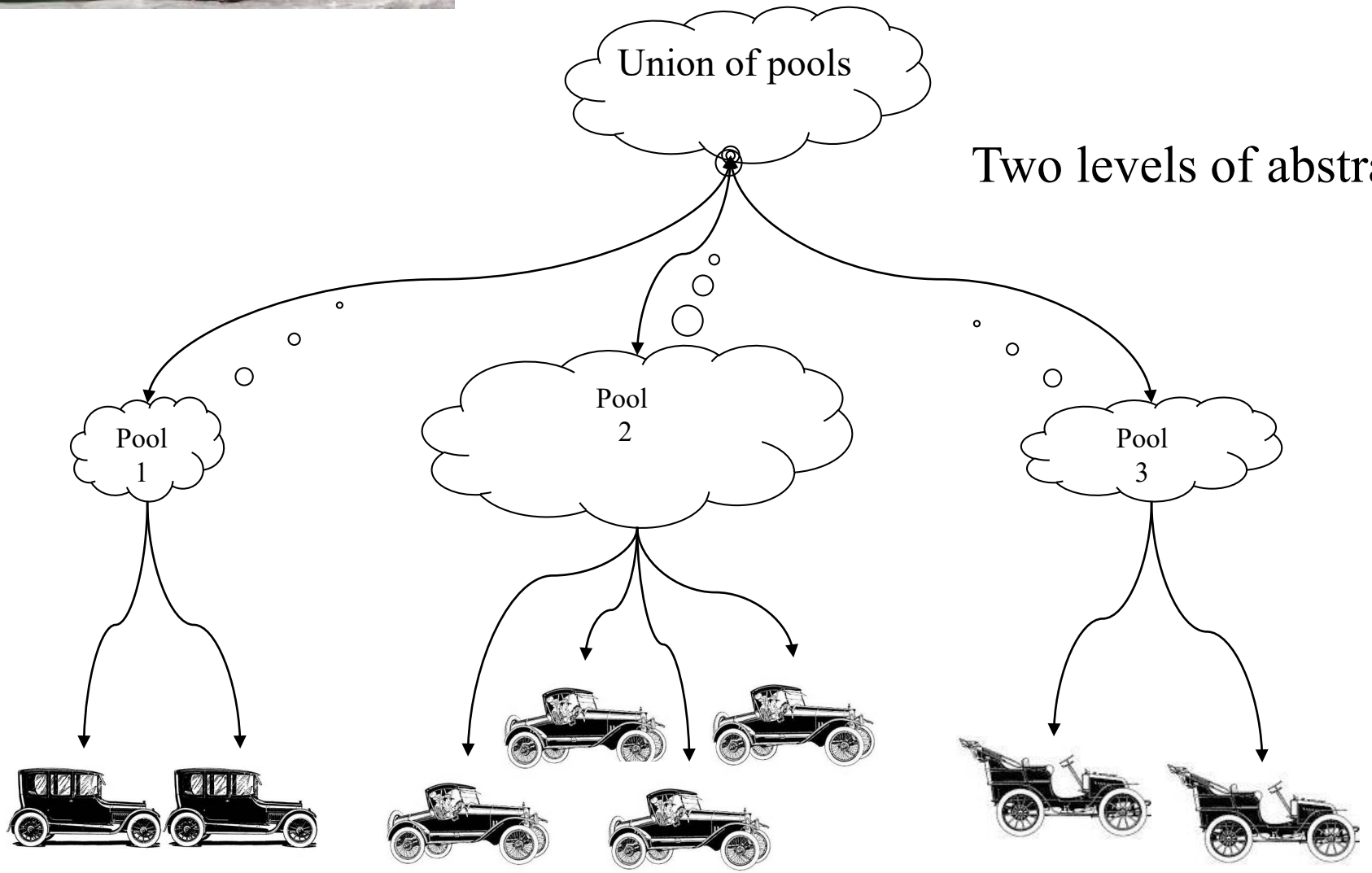
Polyformism (3)

(buffere subroutines)

```
SUBROUTINE SatelliteIntegratorUPC(th_id_num)
…
 CALL
UPC(th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local, &
                    lgranul,rkfasd_UPC,task_descriptor_adr,numobj,pertur)
 END SUBROUTINE  SatelliteIntegratorUPC


SUBROUTINE SituationProcessorUPC(th_id_num)
…
 CALL
UPC(th_id_num,num_threads,ha_1,adr_glb_counter,thread_par_local, &
     granule,Psitanal_UPC,local_task_descriptor_adr,local_num_sit_prob)
END SUBROUTINE SituationProcessorUPC
```

Union of pool of threads



Two levels of abstraction

Creation and control of "Union of Pools"

```
CALL InitUnionPools(pool2_thread_par,num_pool2_threads, &
                    max_init_threads2,pool2_glb_counter,union_atr)
CALL InitUnionPools(AI_1_thread_par,num_pool1_threads, &
                    max_init_threads1,pool1_glb_counter,union_atr)
```

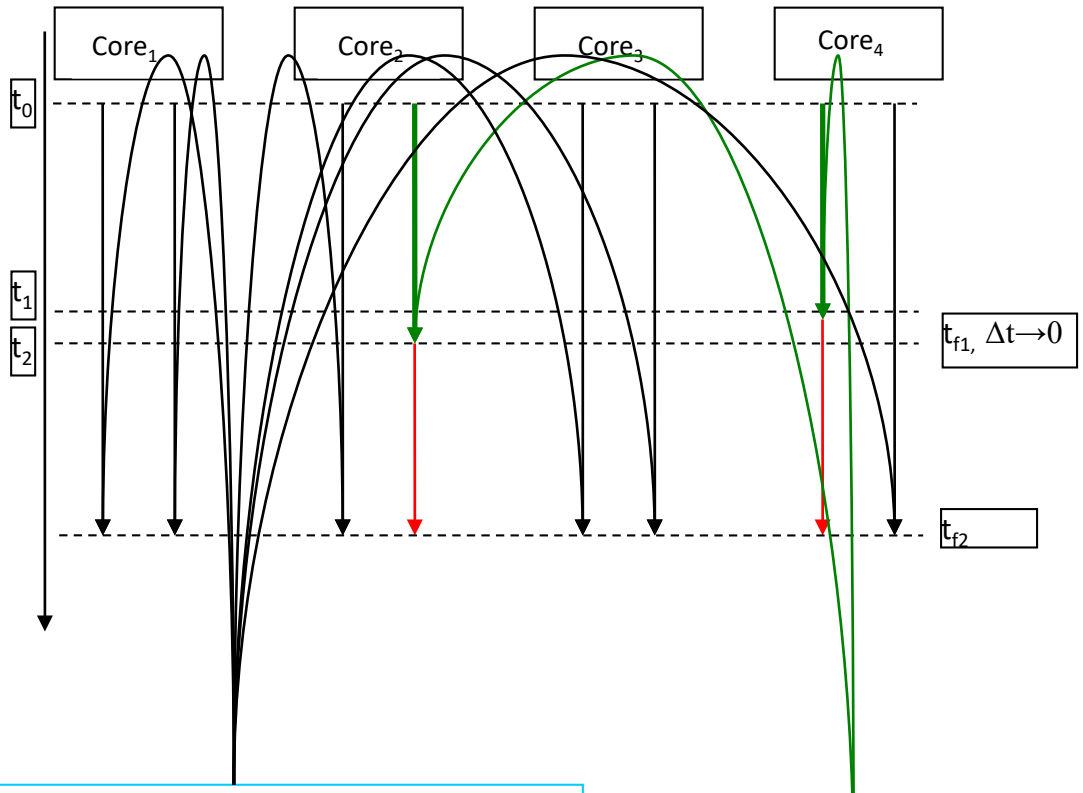
pool[n]_thread_par - pool parameters
num_pool[n]_threads- number of created threads
max_init_threads[n] - initial activated threads
pool[n]_glb_counter – pool counter
union_atr - array, contains parameters of the Union

Other name – other union

```
CALL DynamicPoolsControl(union_atr)
```

Tis subroutine starts and synchronizes the threads of the included pools

Application of "Union of pool of threads" model



Двух поточный пул решает свою задачу и освобождает два ядра, которые сразу же занимают два вновь активированных потока из другого пула.

The first pool was created by eight threads, of which only six are initially active. Two new threads are activated after the completion of the second pool.

The second pool of two threads starts after the first, but ends before

- Модел за дефиниране
- управление

Development of tools for models creation

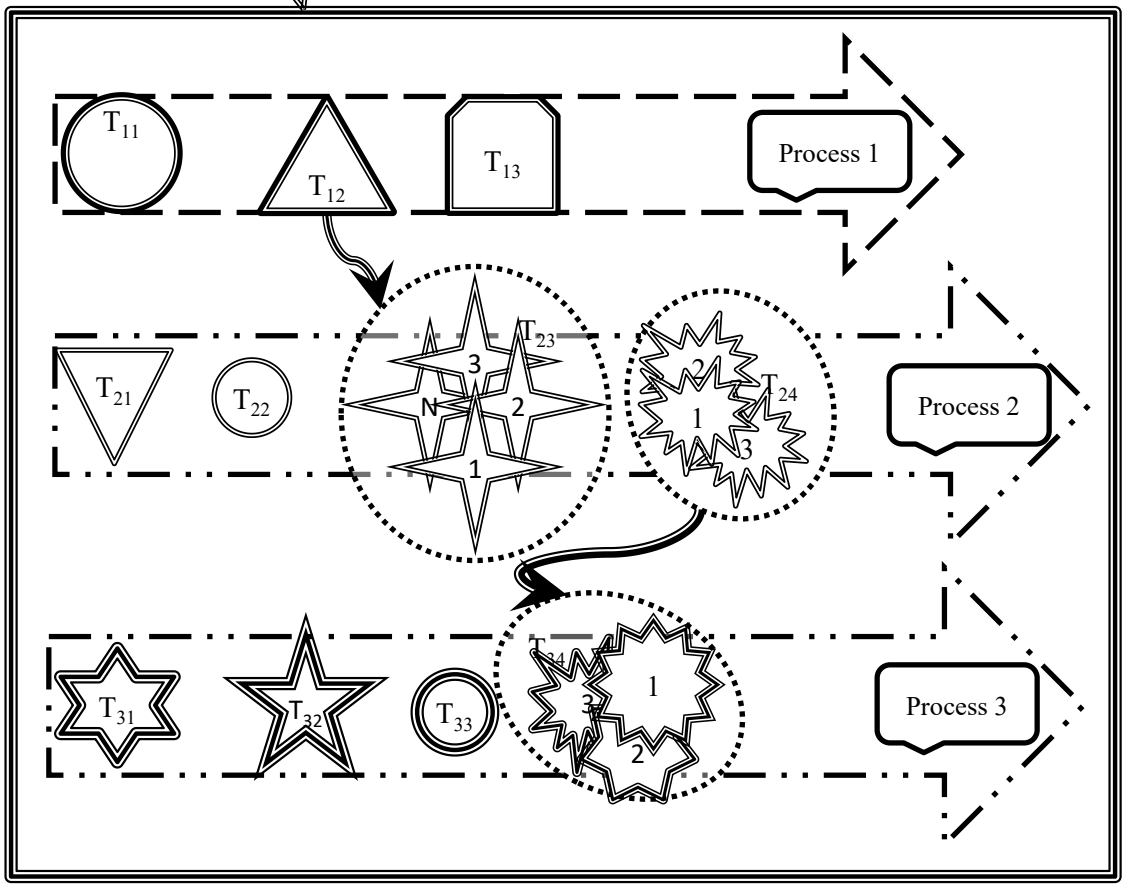
Creation of models of considered objects

- Models of various multi-satellite sistems
- Models of space debris
- Calculation models of geometric and physical quantities along the trajectory
- Situational problems model
- Other models are under development …

System of systems

Presentation

Multiphysics simulation tools



Features

- multidimensional
- heterogeneous
- interconnected

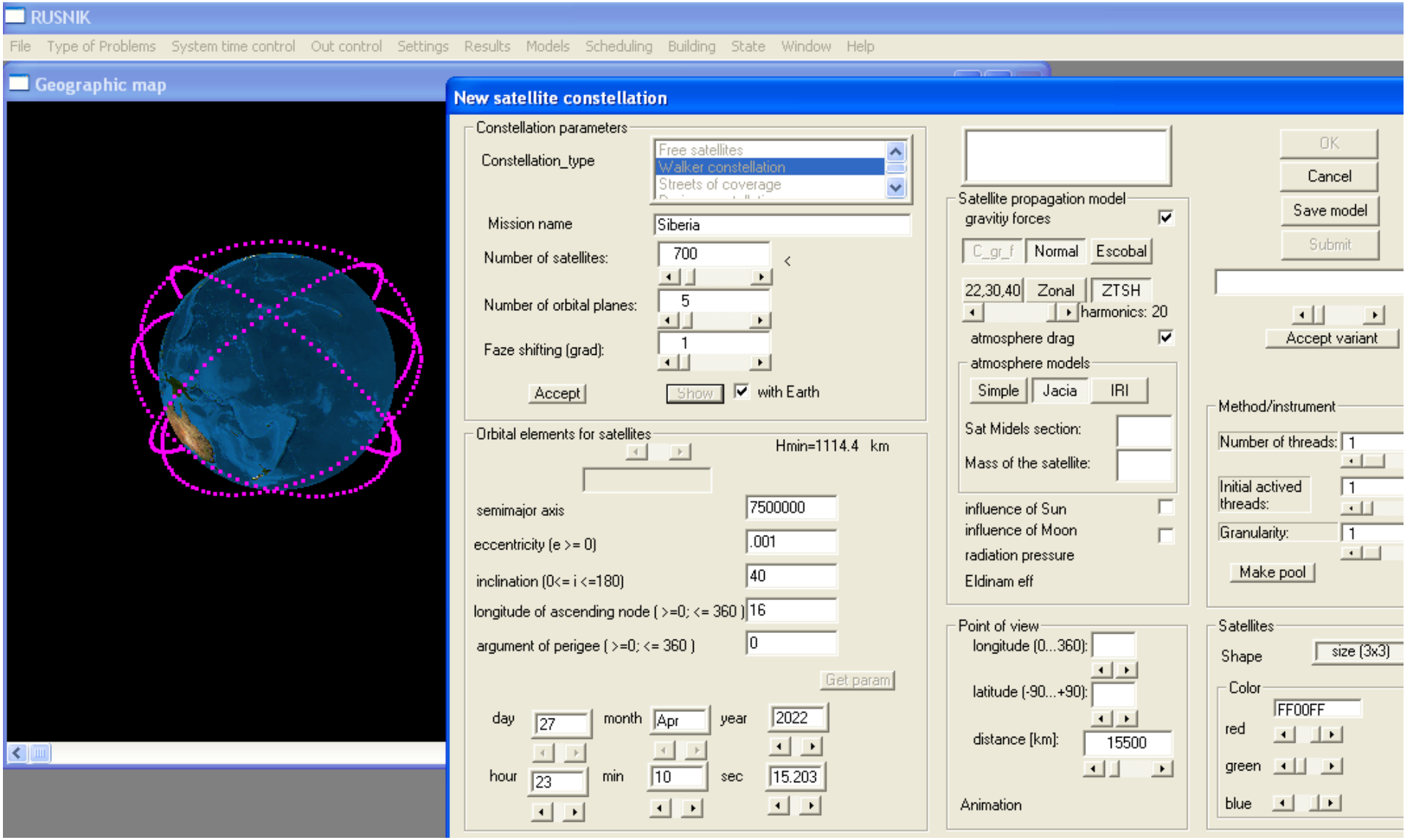
Computer simulation

- multiphysics
- multiscales
- Multisolvers

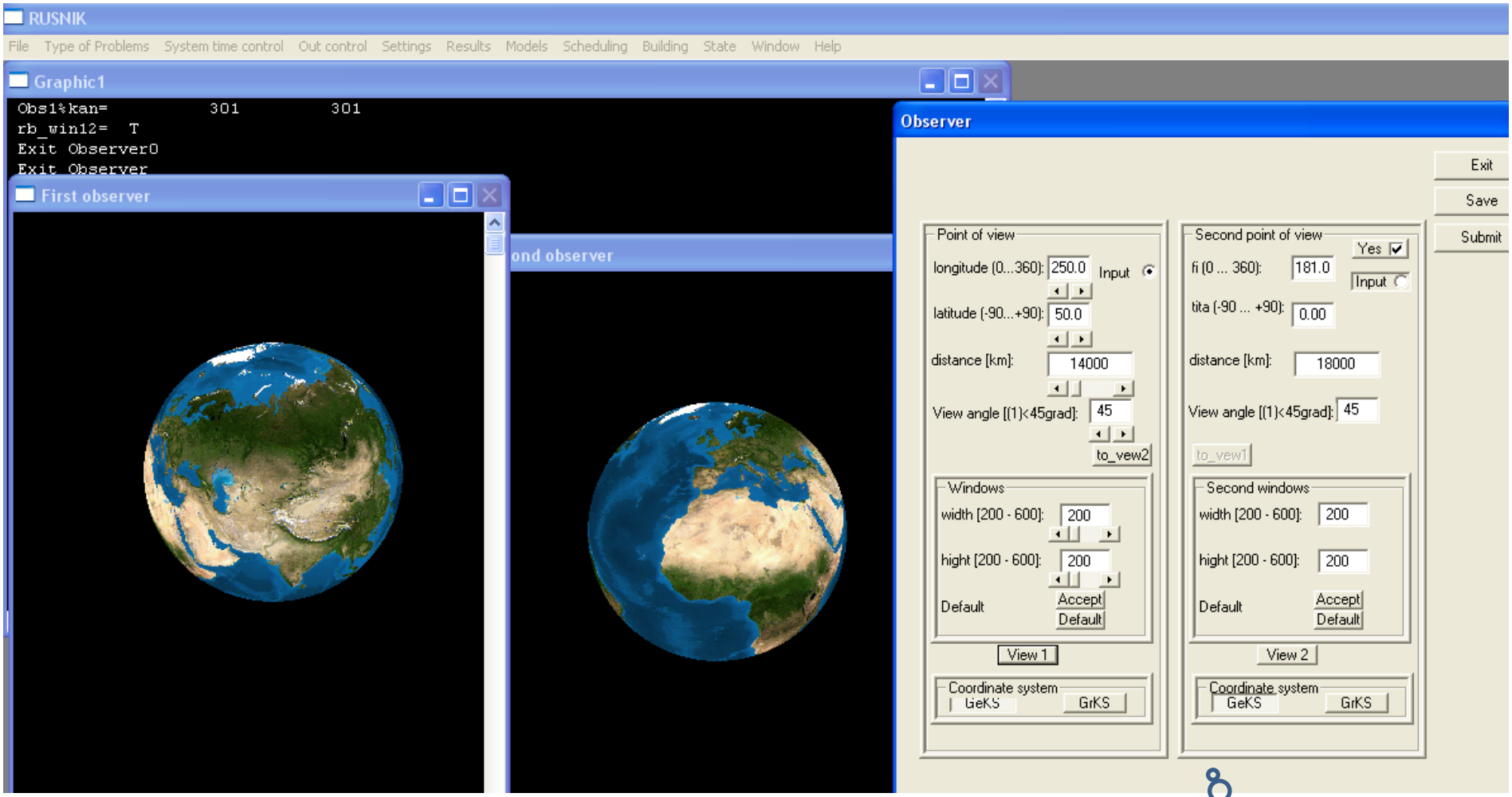
- object oriented programming
- parallelism

Opinion is directed to development of semantic and connections between different models

Satellite constellation creation



Models of the Observers creation



Observers „look“ at the Earth from different points of view

Parameters of the Observers

Pattern „Observer“

```
module
  type     Observer
    logical   Earth        ! To image (.true.) or not to image (.false.) the Earth.
    logical   satellites
    logical   Sky          ! To image (.true.) or not to image (.false.) the sky.
    integer   KS           ! 1- GeKS, 2- GrKS
    integer   fi,tita      ! fi- longitude and tita- latitude
    real*8    distance     ! distance to object
    real*8    l            ! coordinate of  “eye” of the observer in coordinate system
    integer  view_angle
    integer  kan           ! Number of Observer’s window
    logical   active, free! .true.OR.false.
    union                  ! parameters of the windows!
      map
        integer   width,high
      end map
      map
        integer   win(2)
      end map
    end union
  end type  Observer
end  module
```

Describes the scene of the action and the way it will be presented

Development of toll for multiphysics simulations

- Descriptors of different types objects
- Multiphysics simulation model creation
- Simulation model interpretation

Descriptors of different types objects

The different descriptors are distributed at different levels according to the order in which the respective objects will be "active" during the simulation.

Level_0 – contains descriptors of all calculation instruments

Level_1 – contains descriptors of satellite systems, space debris, "Observers on Earth"

Level_2 – contains descriptors of trajectory model calculations, "Observers" on satellite systems and space debris

…

```
CALL  add_object(num_obj,descriptor_adr_old = descriptor_adr, &
                 descriptor_adr_new= descriptor_adr, &
                 level__1= Descriptor)
```

add_object- uses polimorphism

Descriptors of different types objects
(example- satellite constellation)

```
IVP_par_SpMis%kind                      = 1
IVP_par_SpMis%sort                      = 1
IVP_par_SpMis%name                      = s_const_name
IVP_par_SpMis%IVP%integ_index           = num_AIs;
IVP_par_SpMis%IVP%num_objects           = Numsat !num_sat
IVP_par_SpMis%IVP%t_adr                 = LOC(t);
IVP_par_SpMis%IVP%dt_adr                = LOC(dt);
IVP_par_SpMis%IVP%xvn_adr               = xvn_adr;
IVP_par_SpMis%IVP%xvk_adr               = xvk_adr;
IVP_par_SpMis%IVP%eps_adr               = eps_adr;
IVP_par_SpMis%IVP%adr_Grv_model         = adr_perturbations;
IVP_par_SpMis%IVP%len_Grv_model         = len_Grv_model
IVP_par_SpMis%IVP%transfer_data_adr     = transfer_adr
IVP_par_SpMis%IVP%work_data_adr         =  work_adr
```

Multiphysics simulation model creation

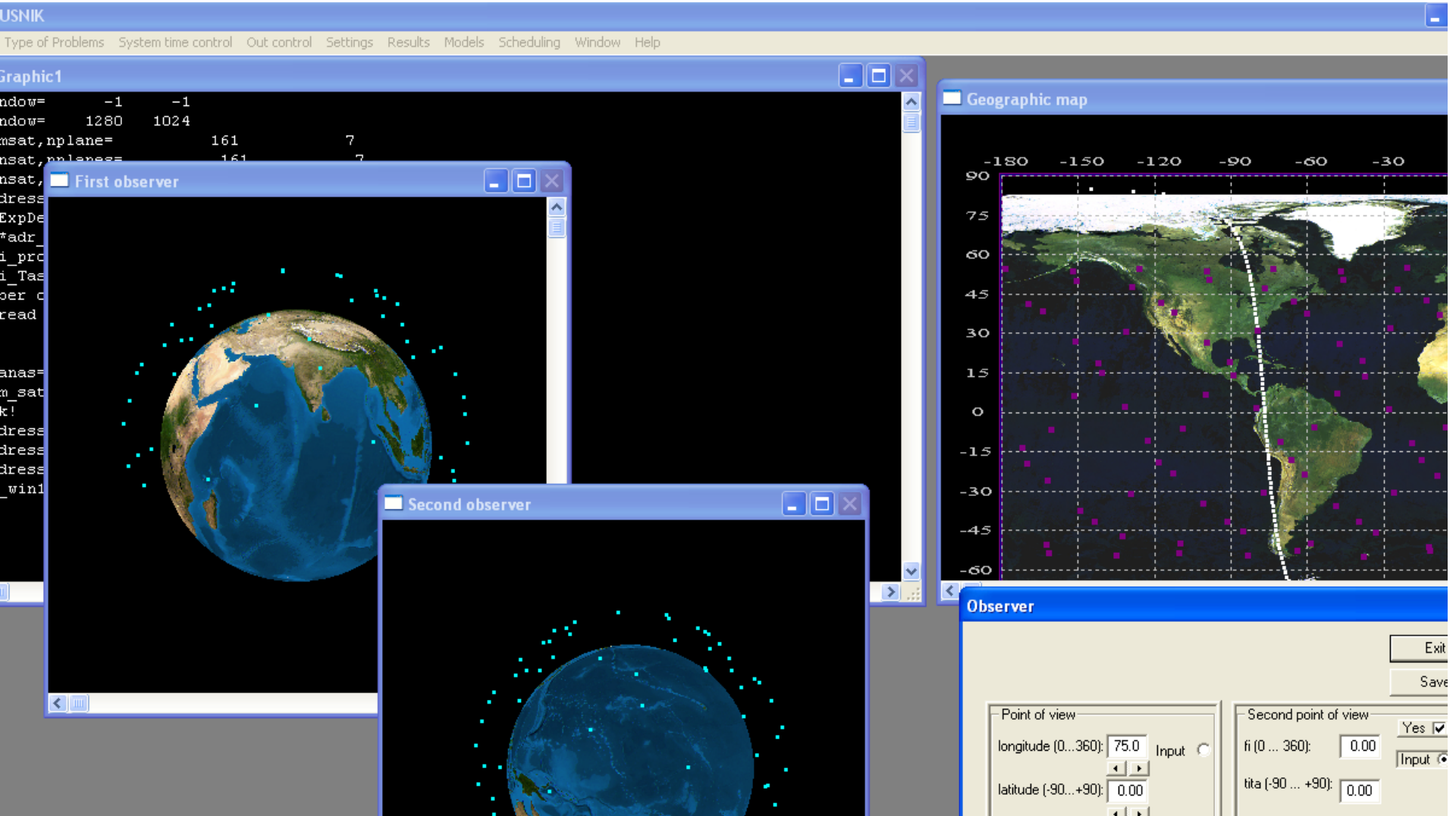
One multiphysic model includes models of objects of different types. The relationships between the models are based on their descriptors. One model can be used to create several objects with different parameters.

When creating an object that will participate in a multiphysic simulation, some of the parameters of its descriptor are copied from the descriptors of the corresponding already created objects.

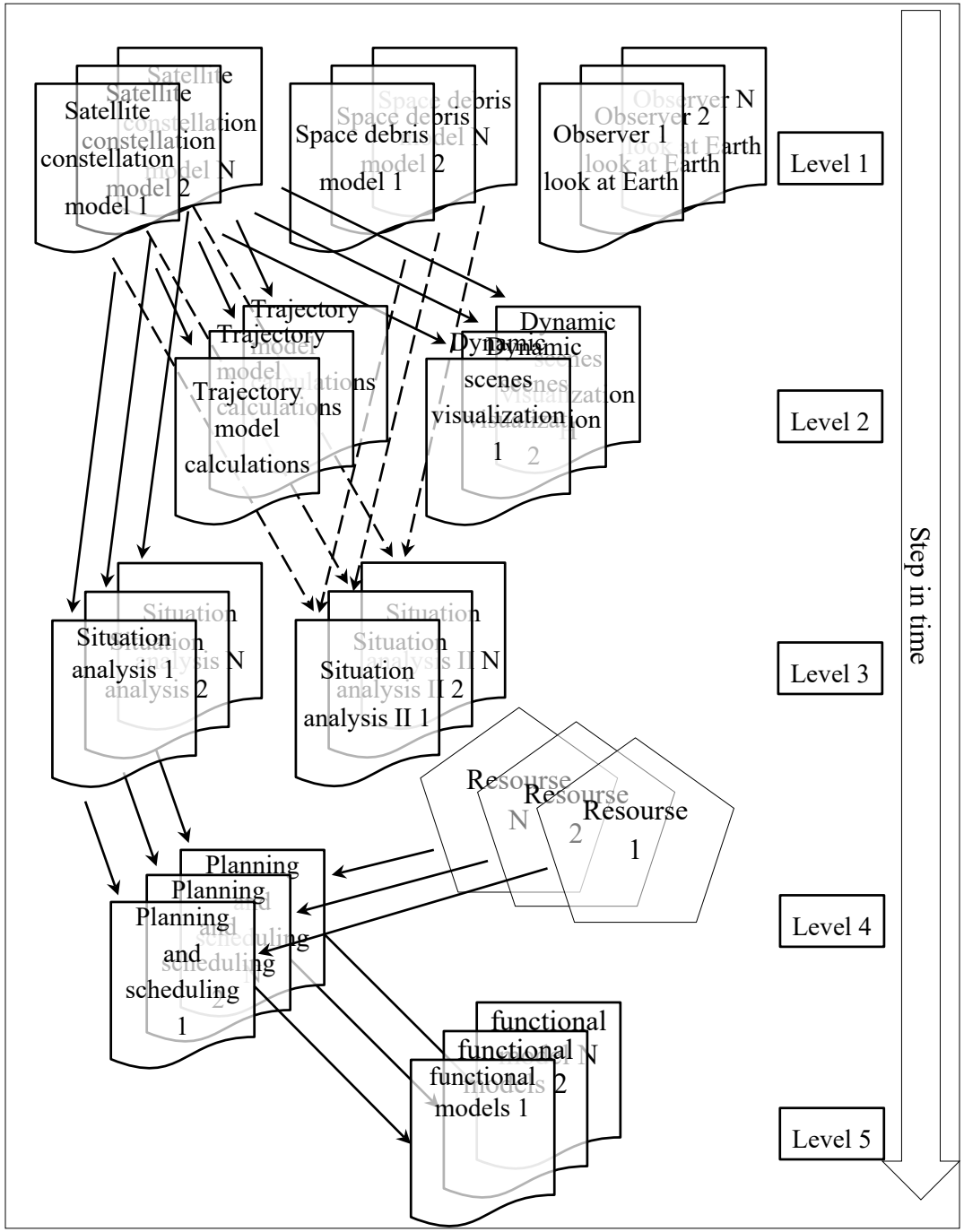
Example: some equal or different types multi-satellite systems and space debris – general population and swarm, result from satellites destruction; some situation analysis solvers for each of satellite systems, and so on …

Simulation with change of Observers' parameters

Observers' parameters can be changed in the course of the simulation



System architecture semantic



Parallelization of processes

Simjulation

Level 1

Union of pools 1

- Integration of motion equations (satellite systems and space debris
- Observer 1 - Earth
- Observer 2 - Earth

Level 2

Union of pools 2

- Calculation of geometric or physic model quantities
- Observer 1 - satellites
- Observer 2 - satellites

...

Step in time