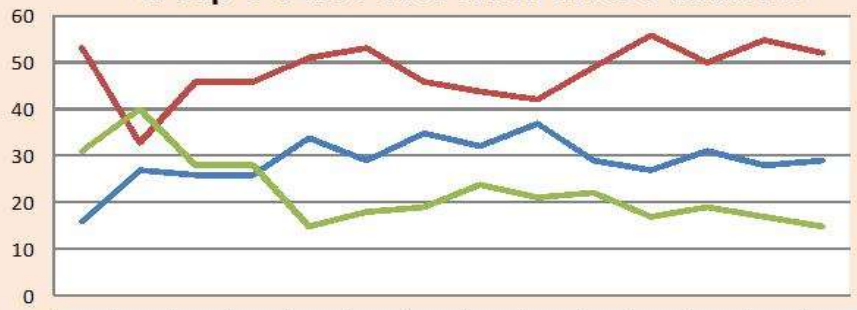
Methodology and Tools for Development and Verification for formal Requirements fUML Models and for Complex Software and Hardware Systems Architecture

Samonov A.V. <a.samonov@mail.ru>

Samonova G.N. <g.samonova@mail.ru>

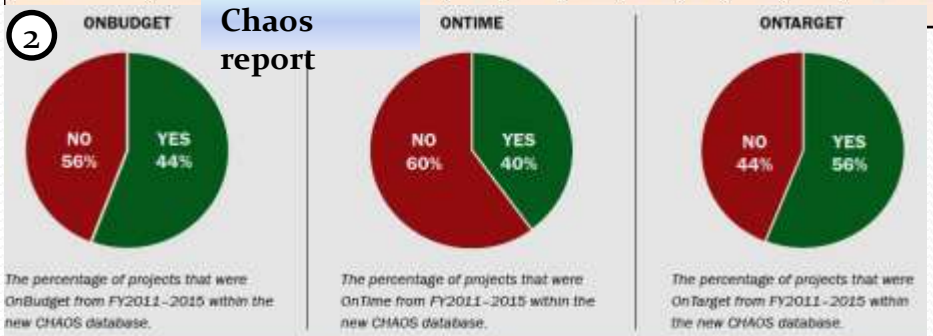
Relevance of the problem

1



	1994	1996	1998	2000	2002	2004	2006	2008	2010	2011	2012	2013	2014	2015
SUCCESSFUL	16	27	26	26	34	29	35	32	37	29	27	31	28	29
CHALLENGED	53	33	46	46	51	53	46	44	42	49	56	50	55	52
FAILED	31	40	28	28	15	18	19	24	21	22	17	19	17	15

2



The percentage of projects that were OnBudget from FY2011–2015 within the new CHAOS database.

The percentage of projects that were OnTime from FY2011–2015 within the new CHAOS database.

The percentage of projects that were OnTarget from FY2011–2015 within the new CHAOS database.

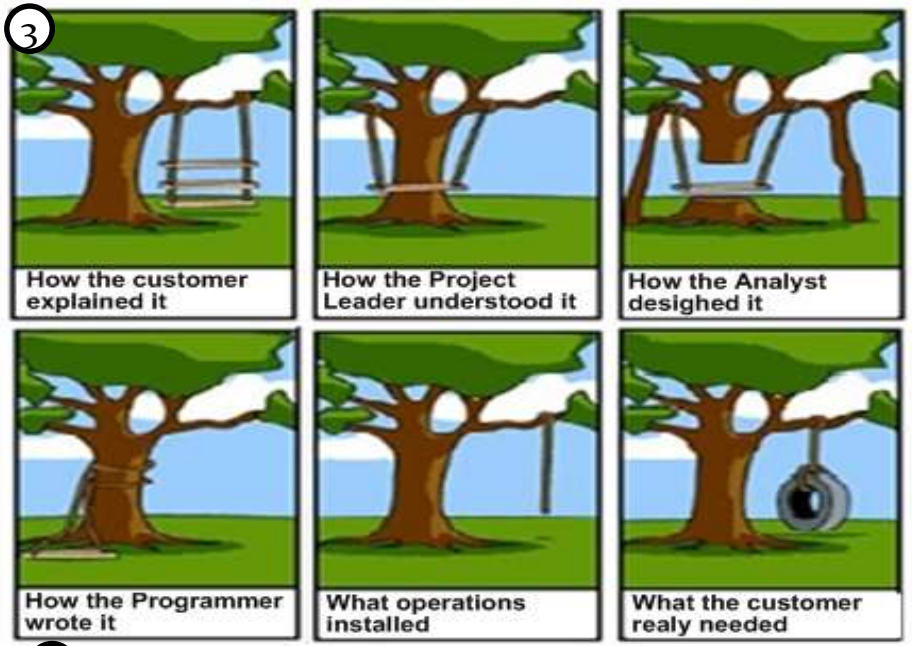
Chaos report

TRADITIONAL RESOLUTION FOR ALL PROJECTS

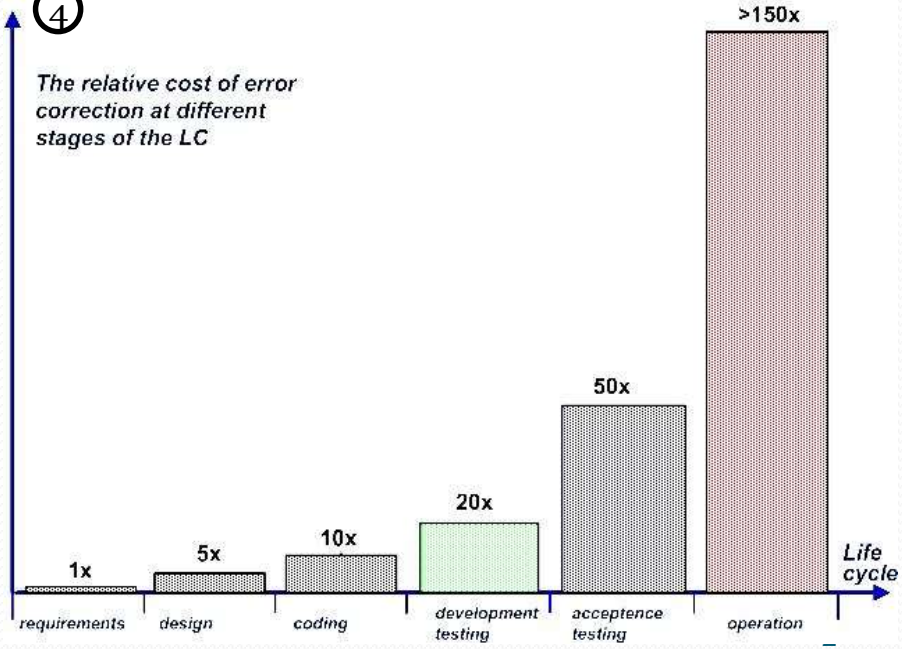
	2011	2012	2013	2014	2015
SUCCESSFUL	39%	37%	41%	36%	36%
CHALLENGED	39%	46%	40%	47%	45%
FAILED	22%	17%	19%	17%	19%

The Traditional resolution of all software projects from FY2011–2015 within the new CHAOS database.

3

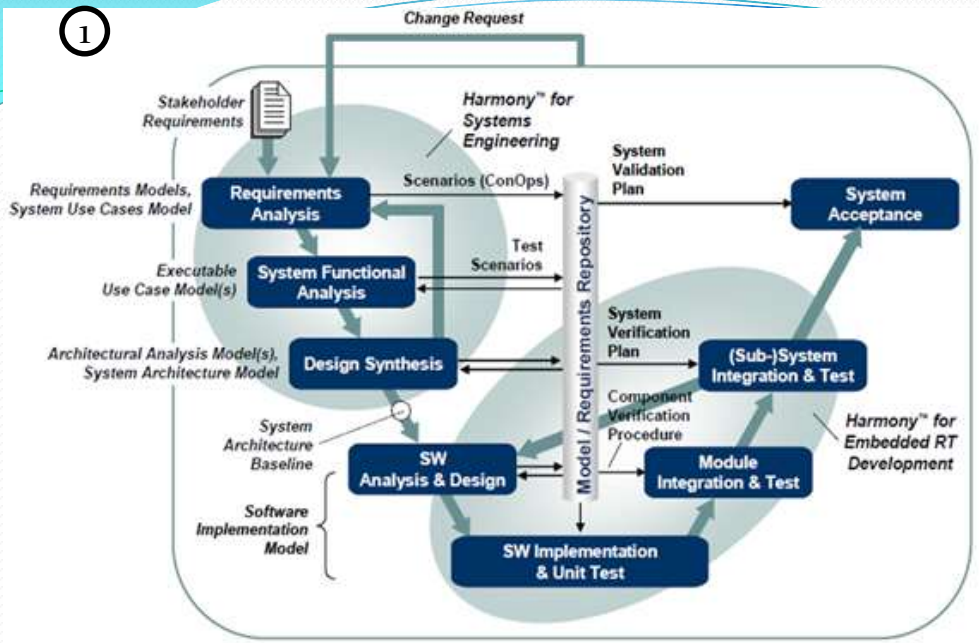


4

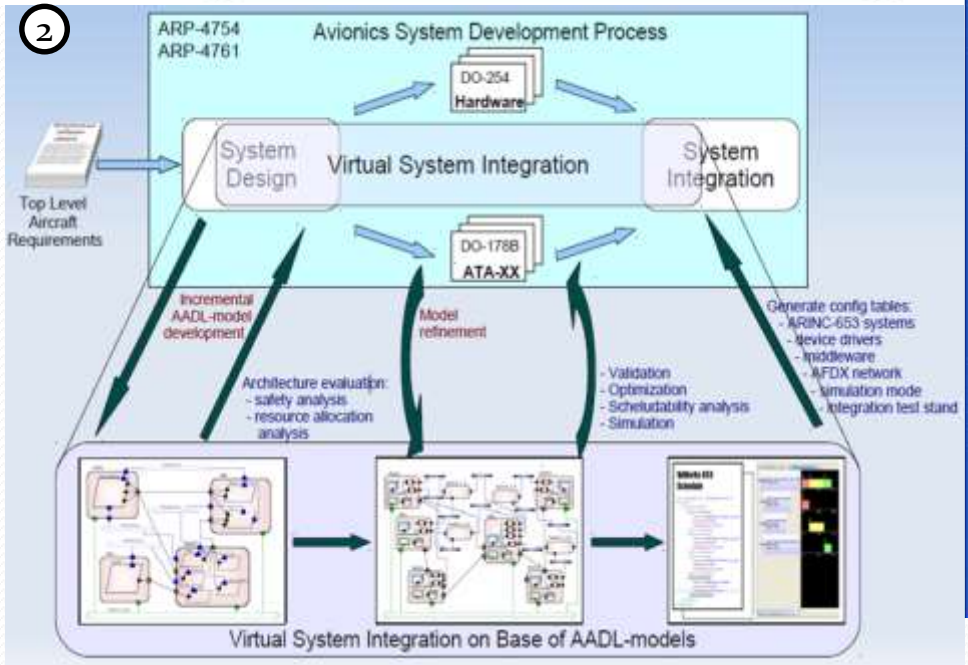


Relevance of the problem: modern technologies, methods, tools …

1



2

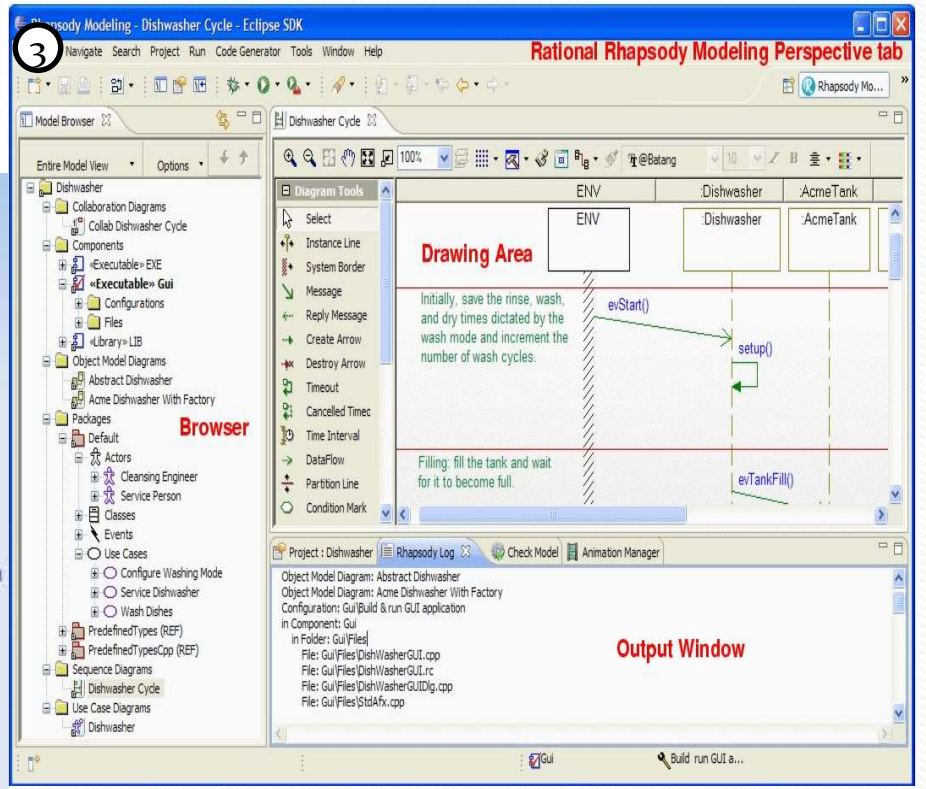


Obstacles :

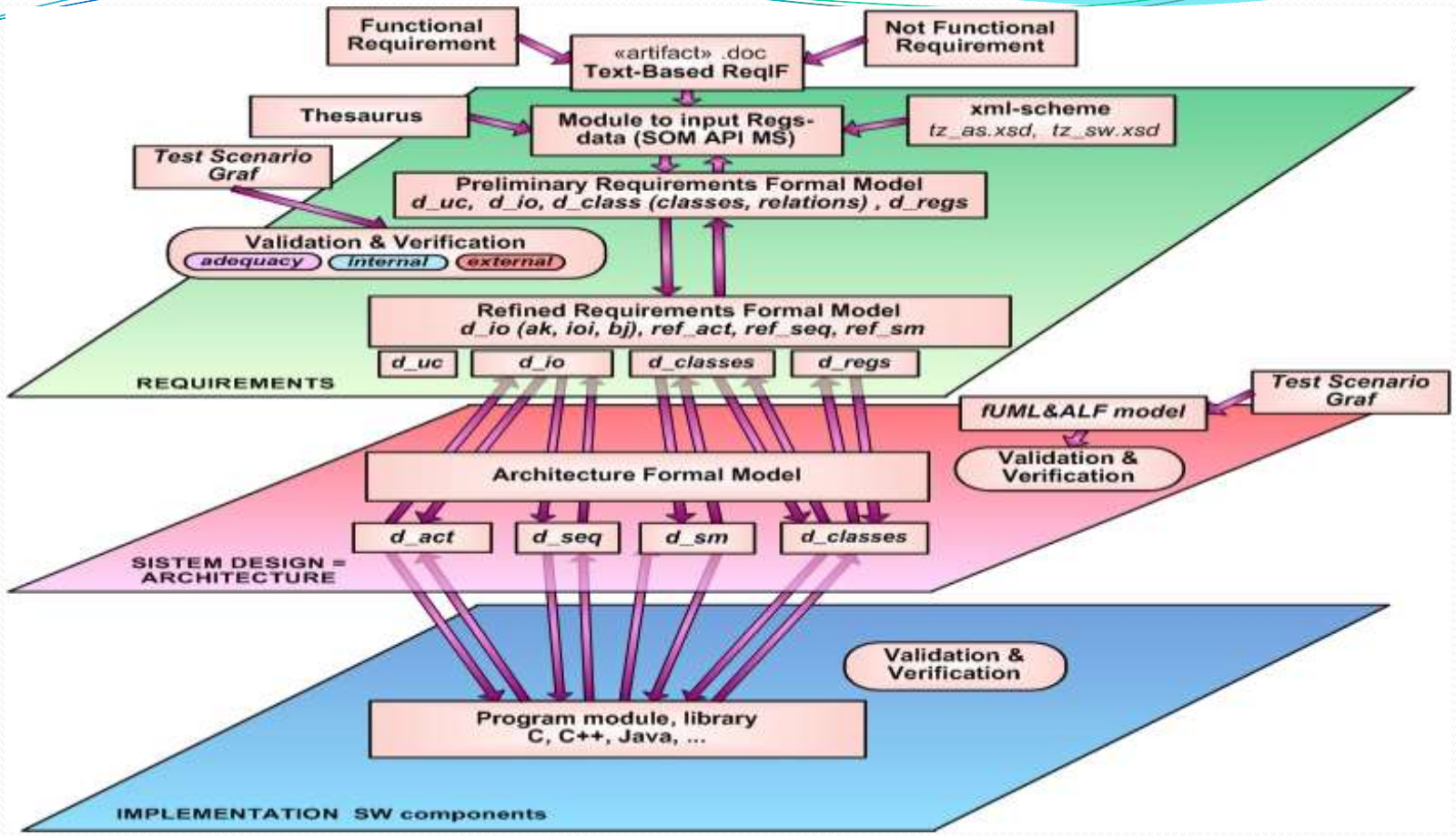
1) objective complexity of the problem of constructing a formal representation of the system requirements on the basis of their original informal presentation;

2) availability of a wide range of languages and tools proposed for building models of analysis, architecture and implementation of the system, in the absence of clear and specific rules and recommendations for their application;

3

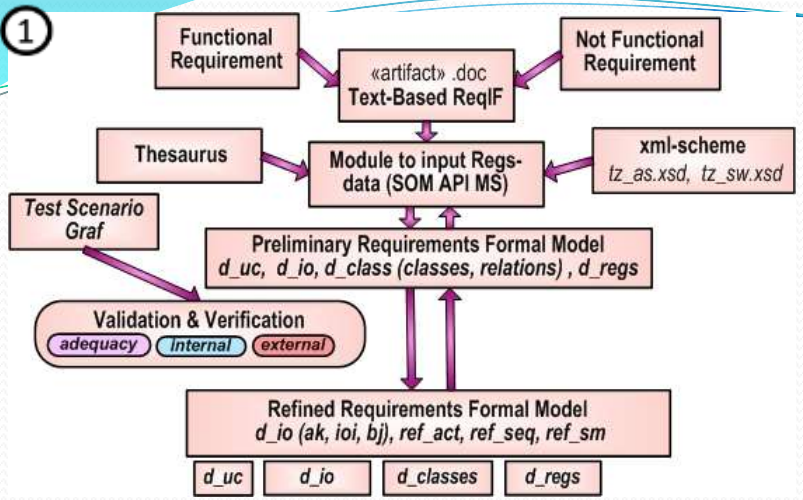


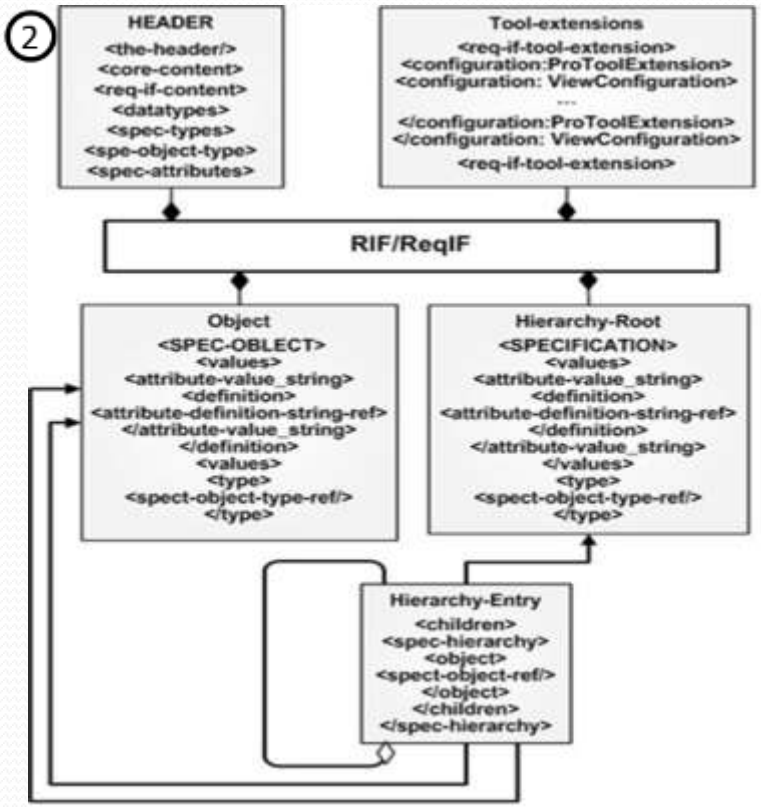
Goals and objectives of the study. The choice of development tools

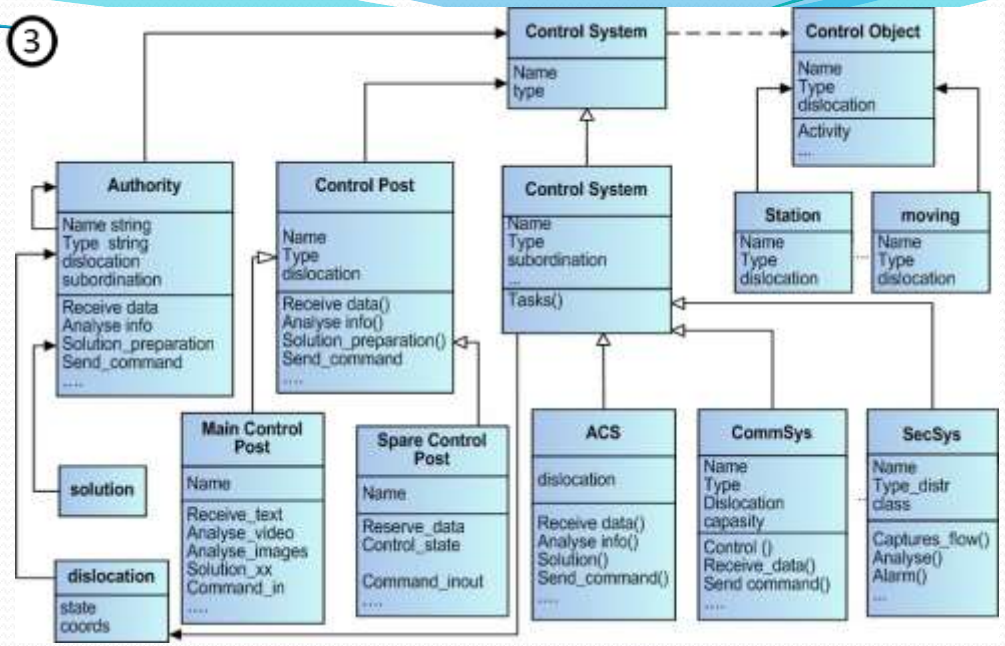


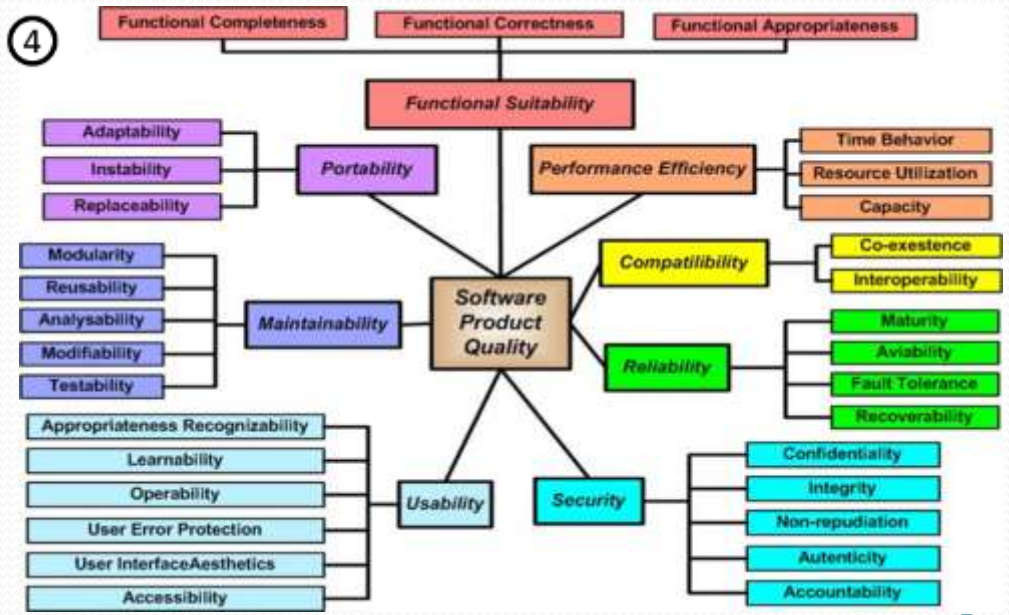
1. To develop a unified conceptual, model, language and tool development and verification environment, we used: – modeling languages UML, fUML, OCL, ALF;
2. To formalize requirements: ontologies of the automated system, quality software, ReqIF, .xsd
3. As means for verification we have chosen: VM fUML, SPIN (Promela), Rodin (Event-B), SMT-Lib, Z3, CVC-4, ALT-ERGO.
4. Eclipse Modeling Framework, Graphical Editing Framework, Papyrus, Moka - development environment and tools: framework, libraries and tools implemented within the Eclipse project:

Step 1: develop requirements model



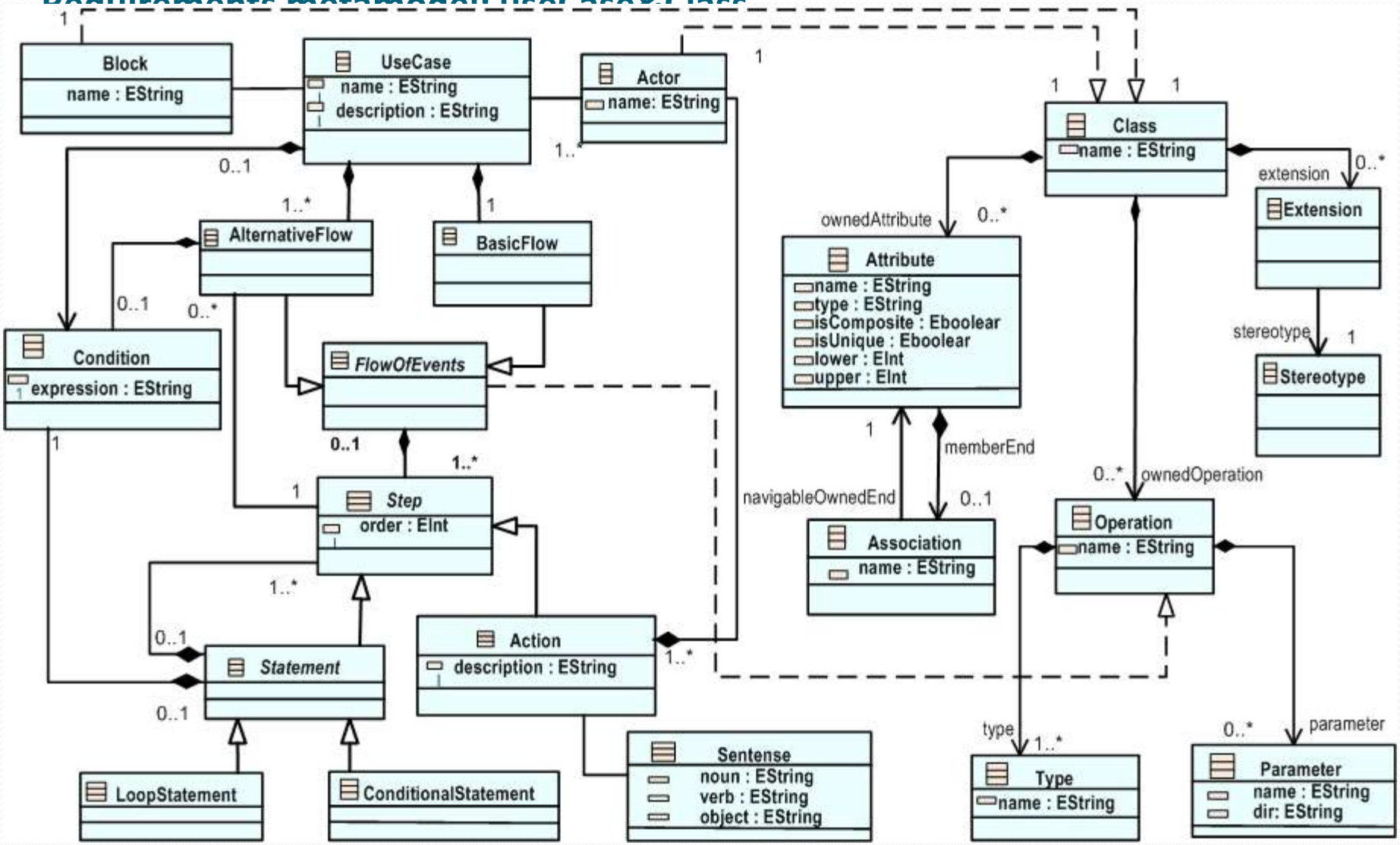




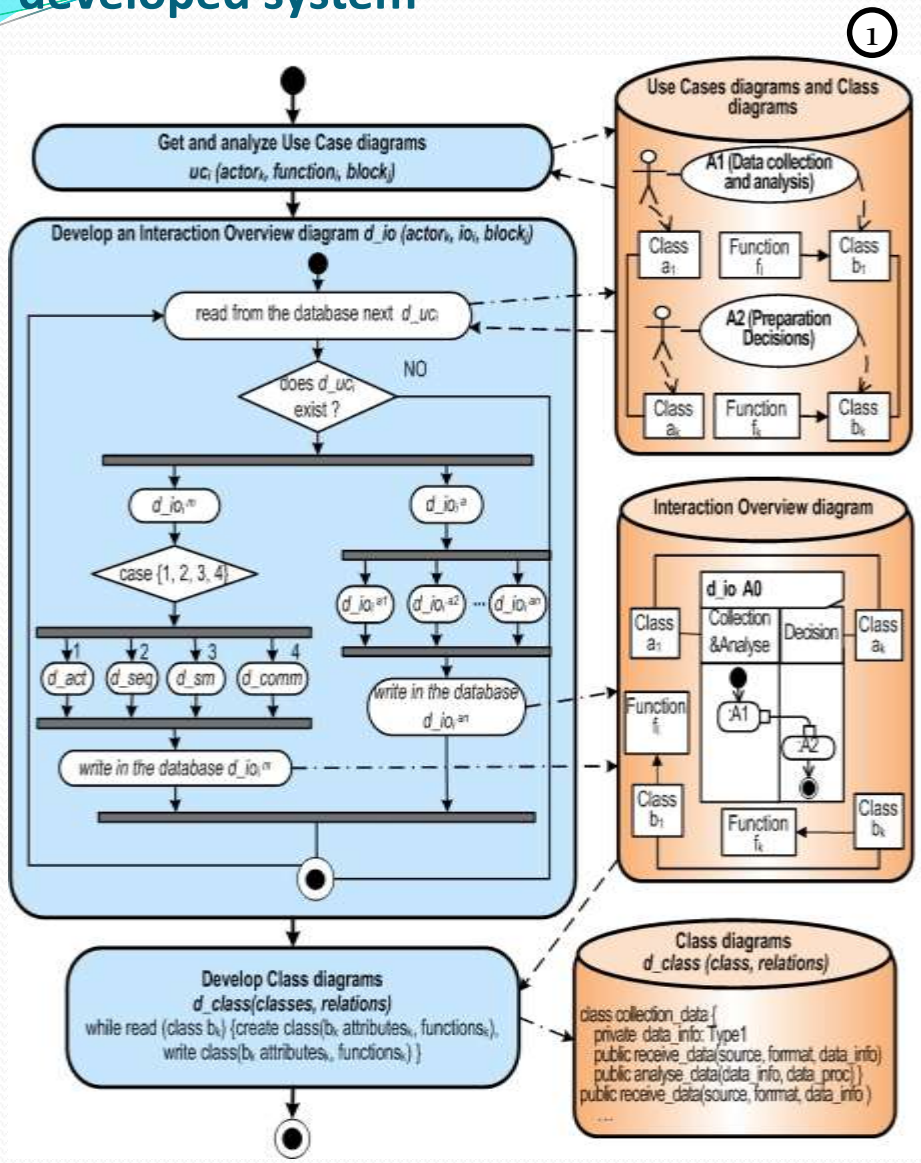


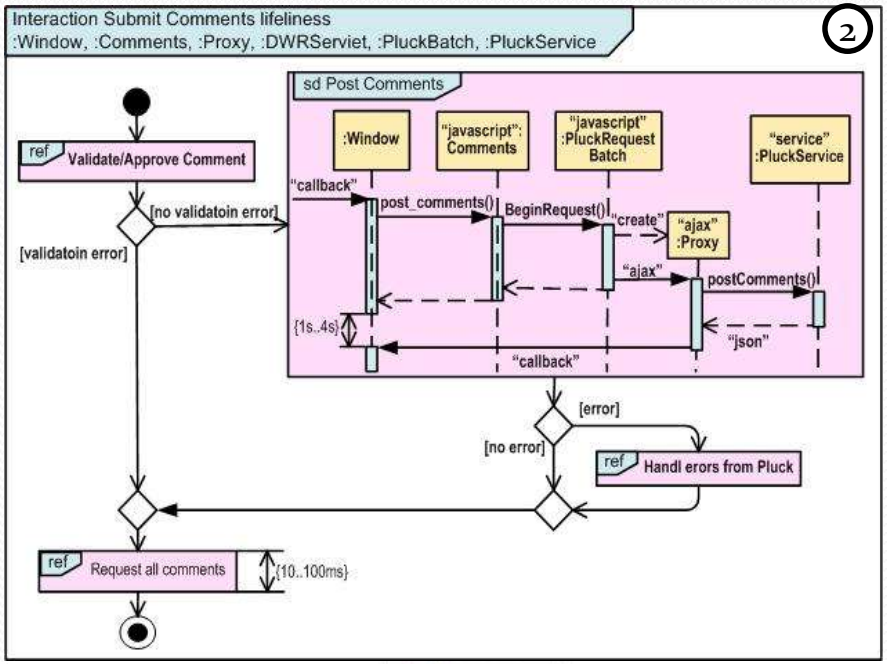
Step 1: develop requirements model.

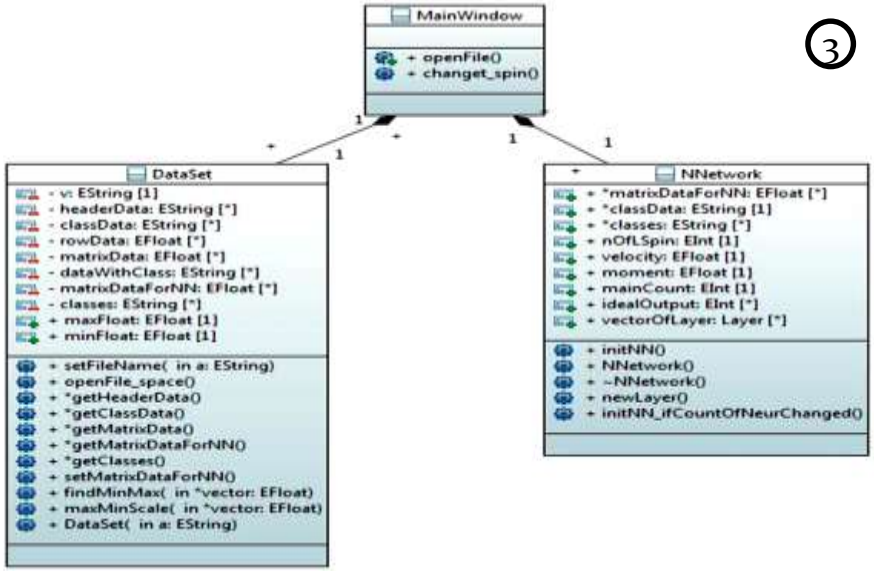
Requirements metamodel: useCase&Class



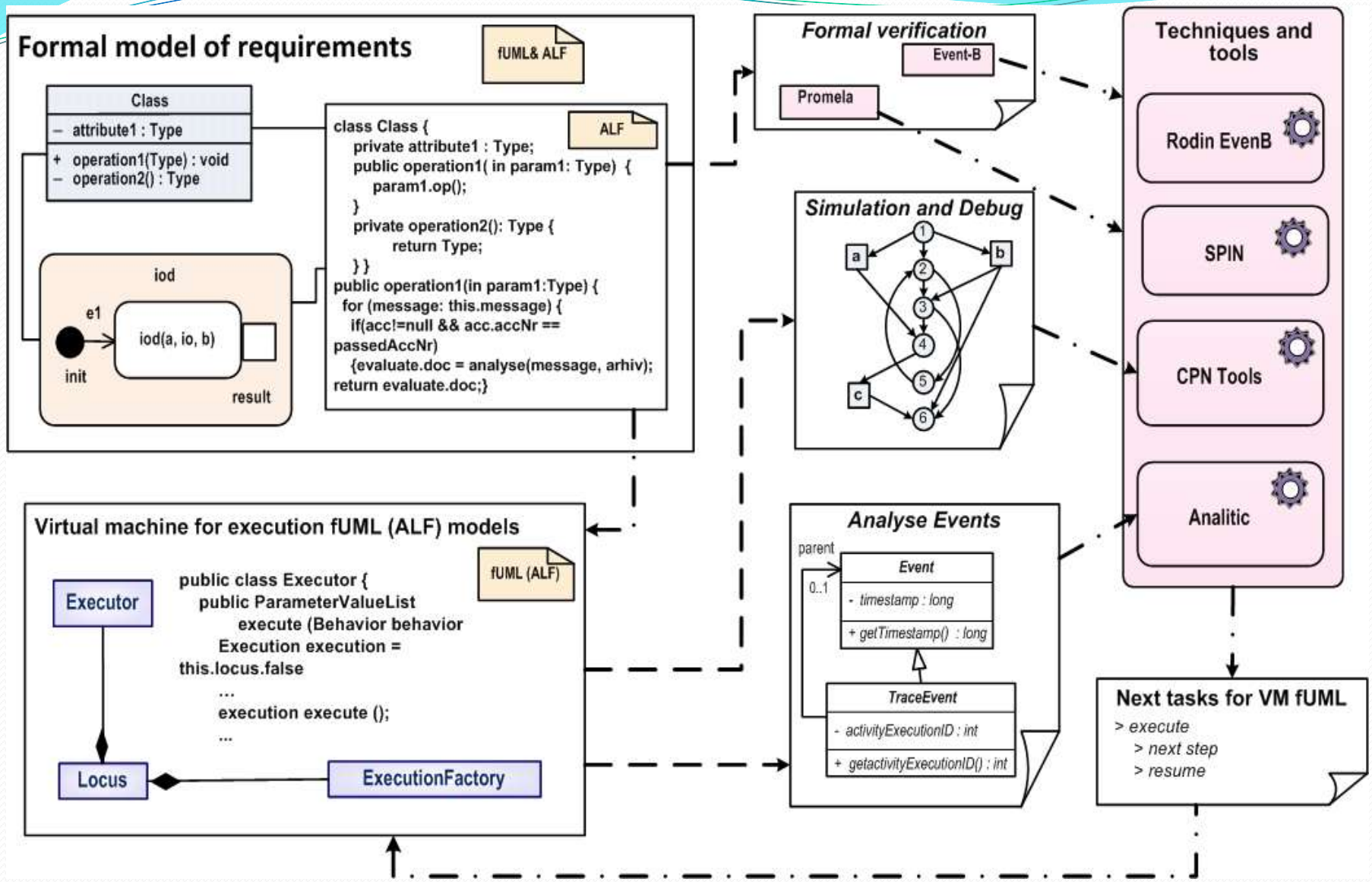
Step 1: develop requirements model. Algorithm for constructing a formal description of the requirements for the developed system



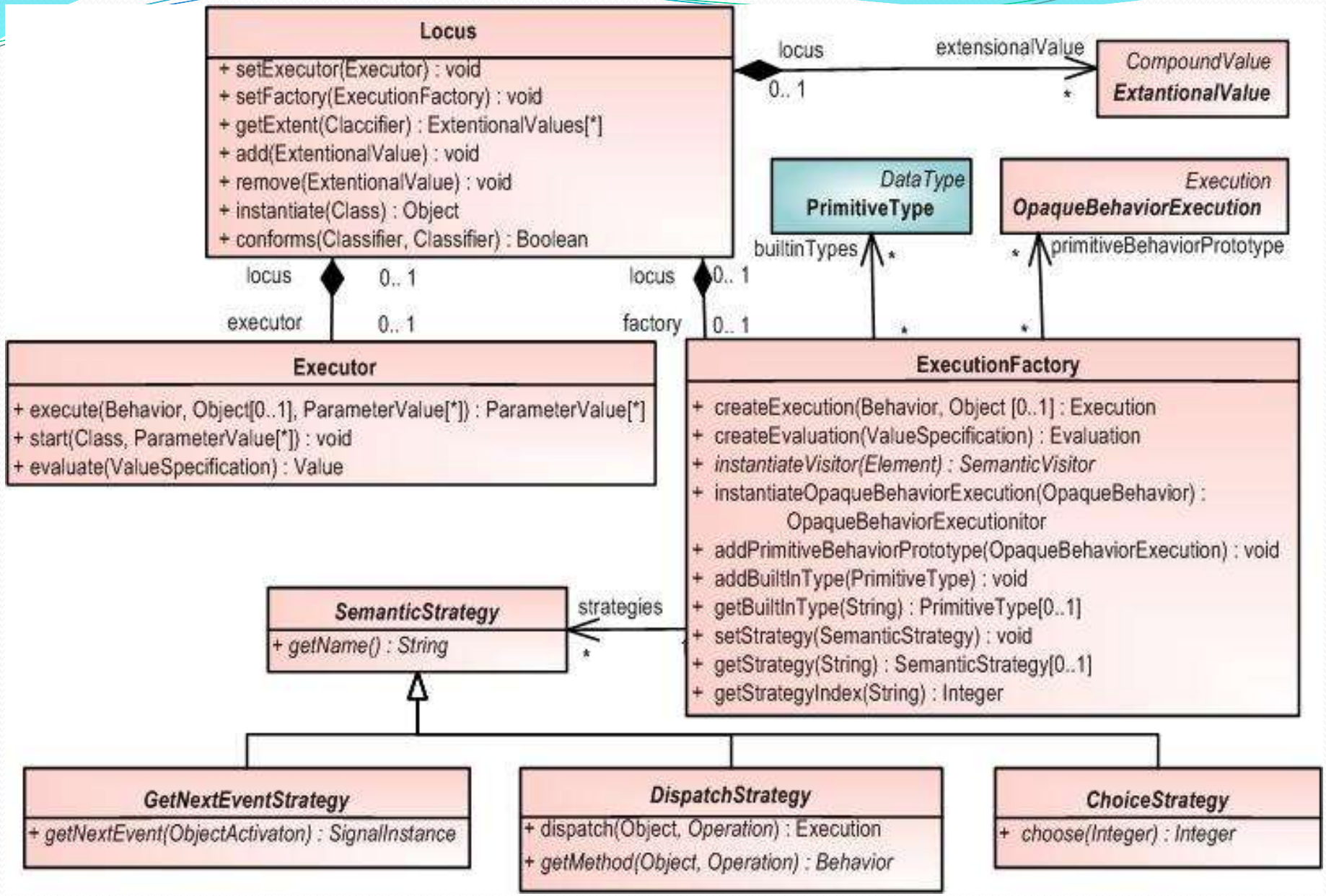




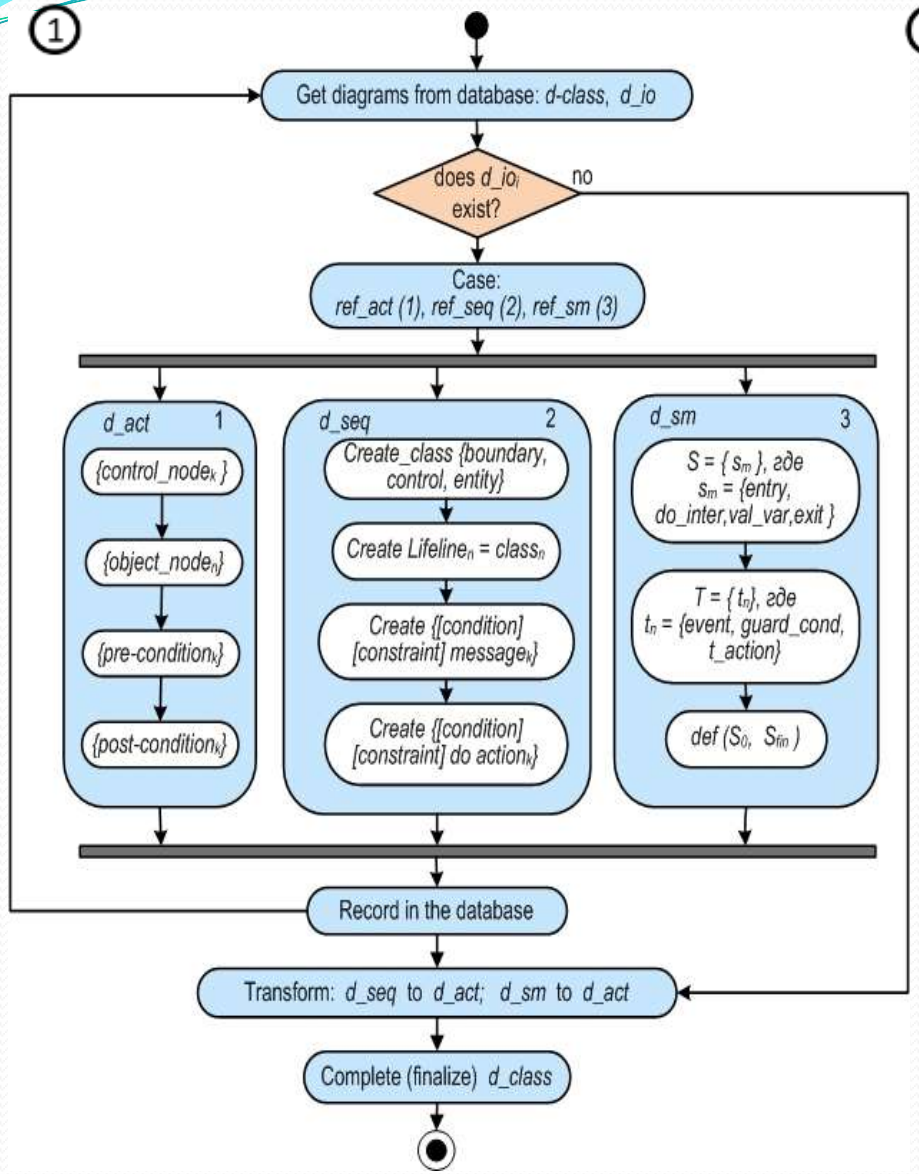
Step 1: Verification of requirements model to software

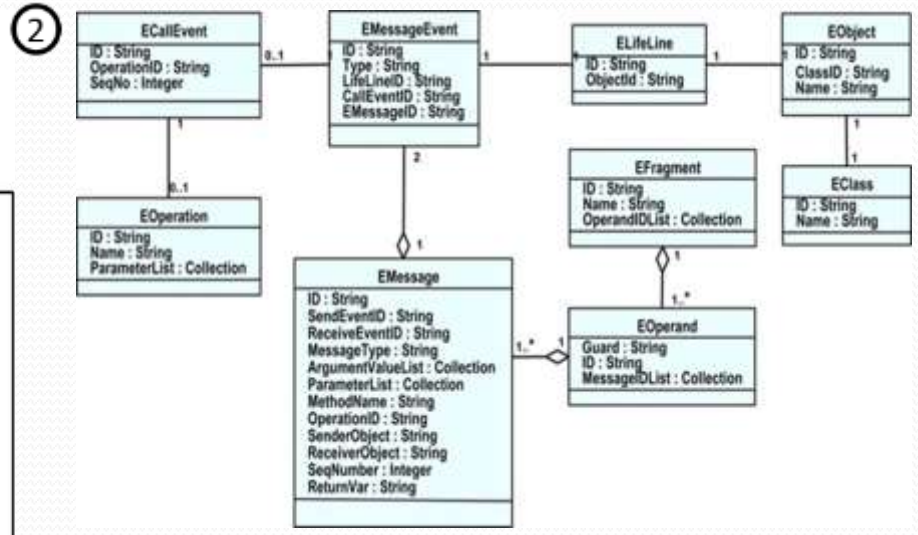


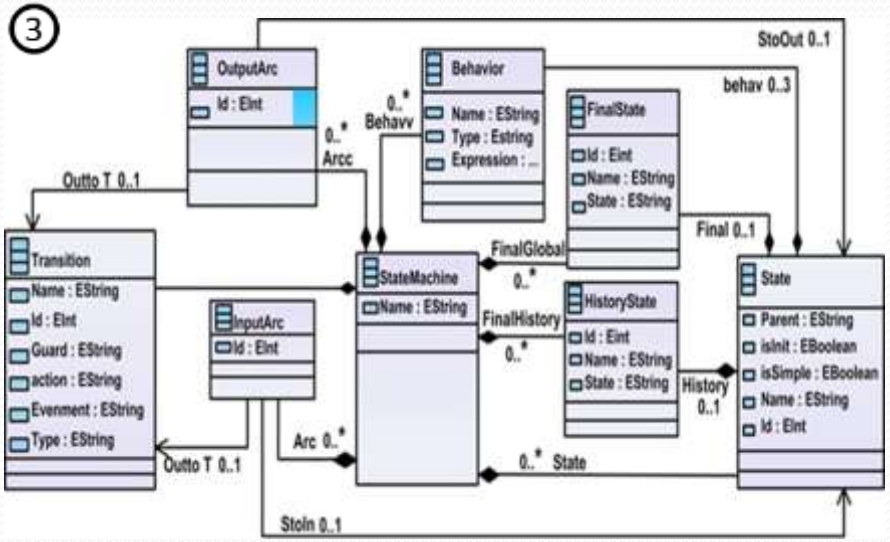
Step 1: Virtual machine fUML



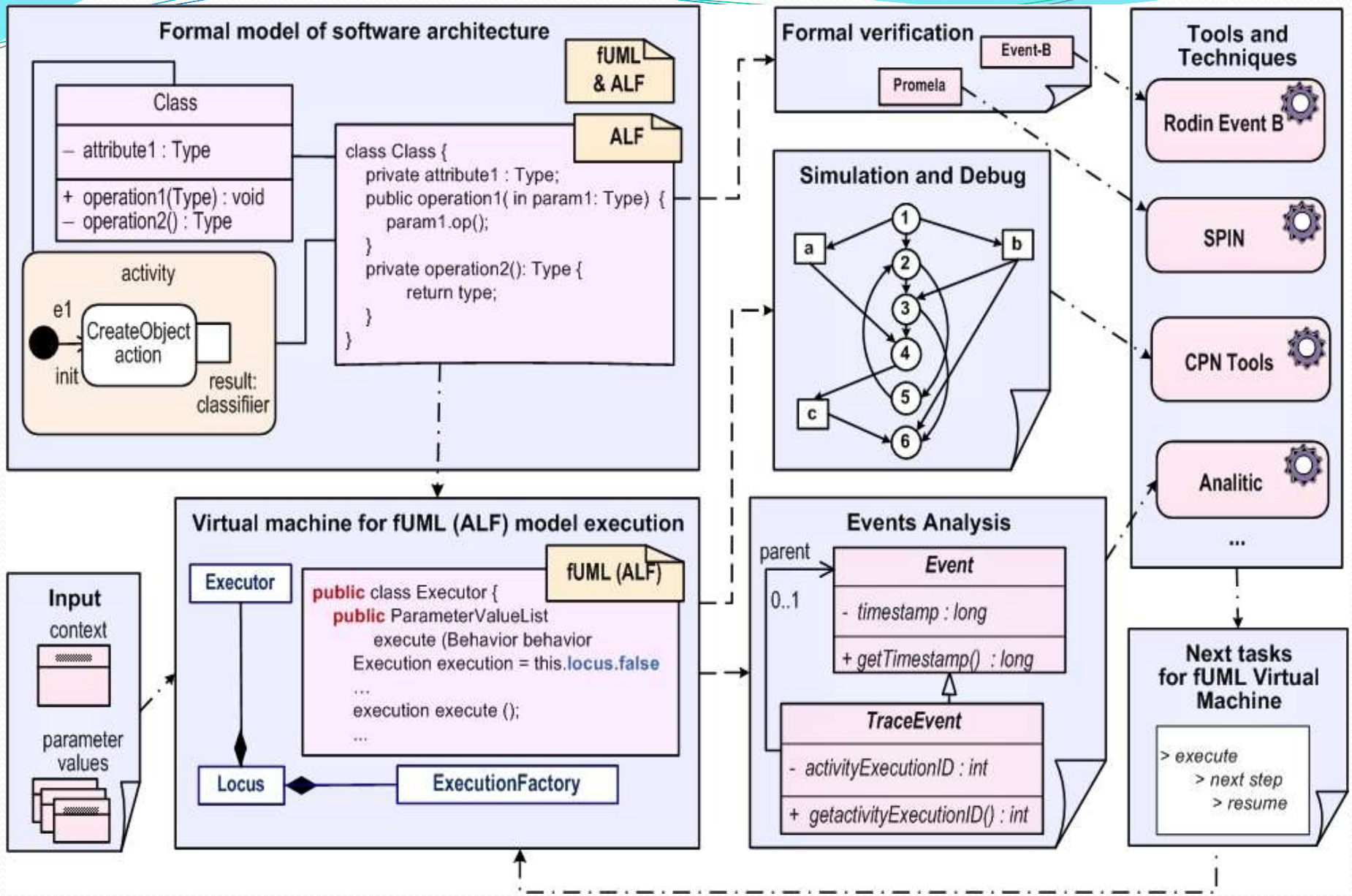
Step 2: develop the model of software architecture



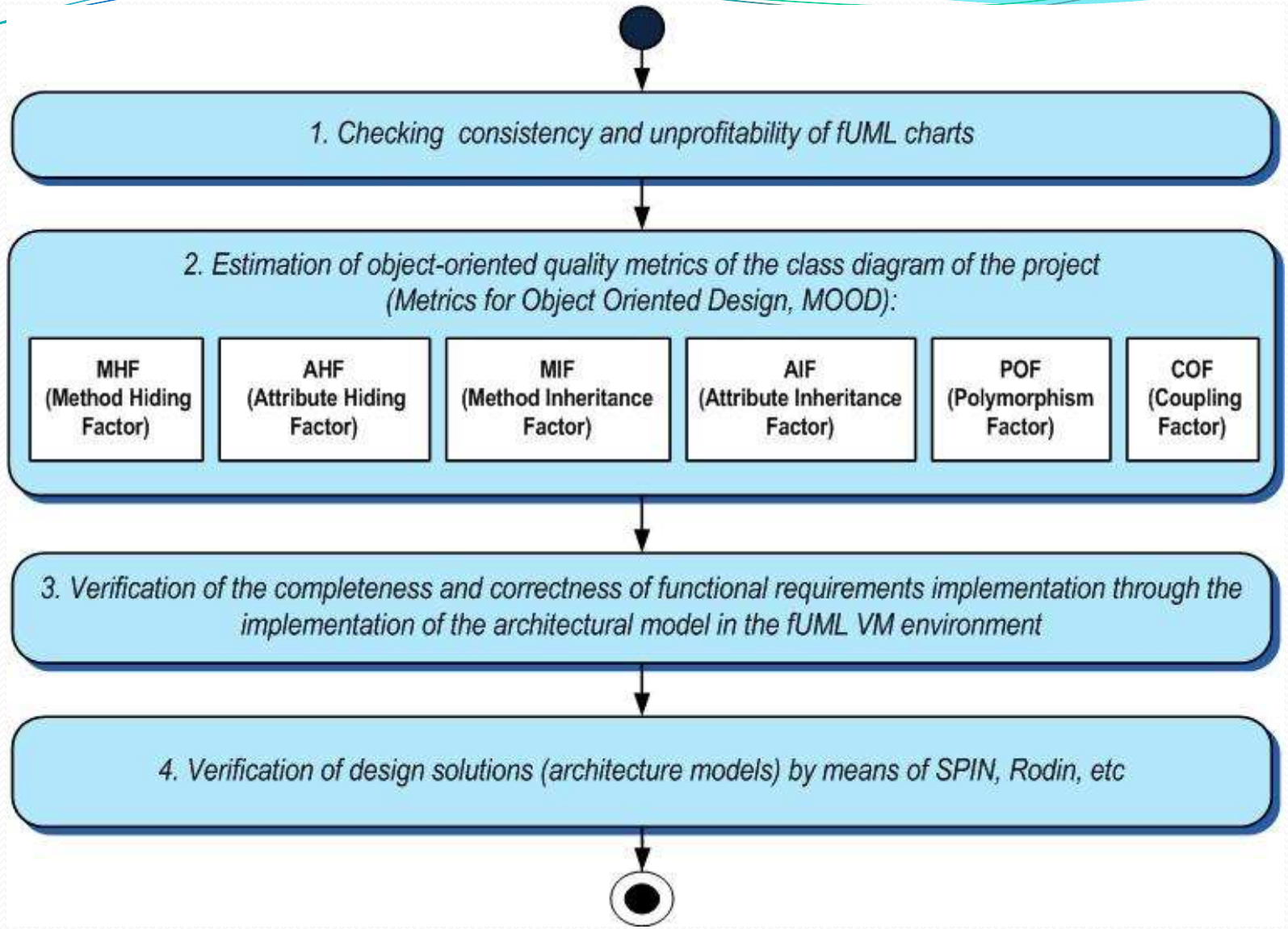




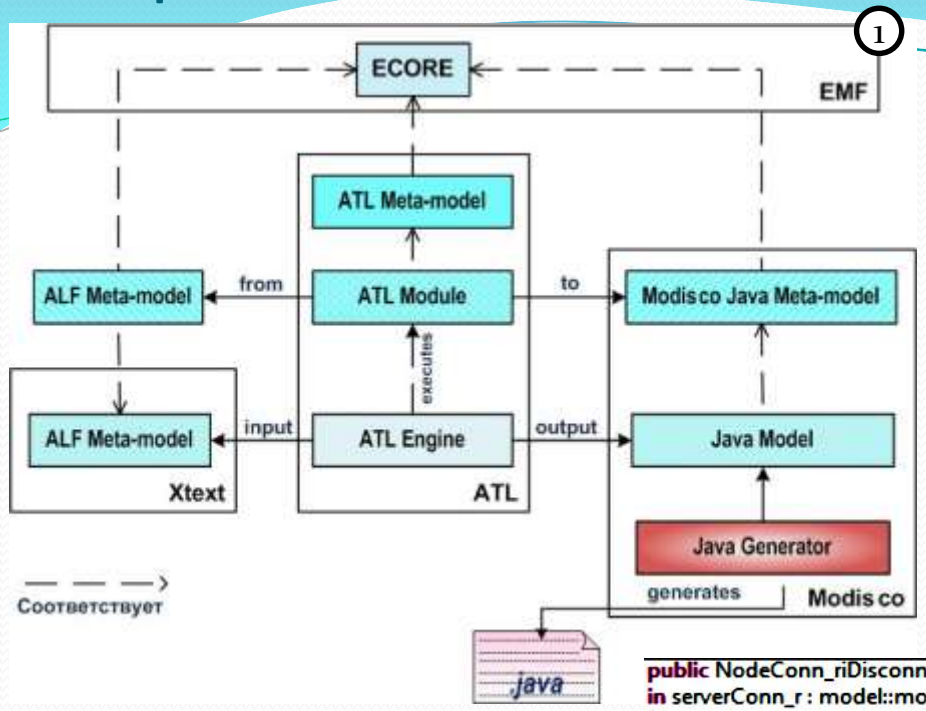
Step 2: Verification the model of software architecture

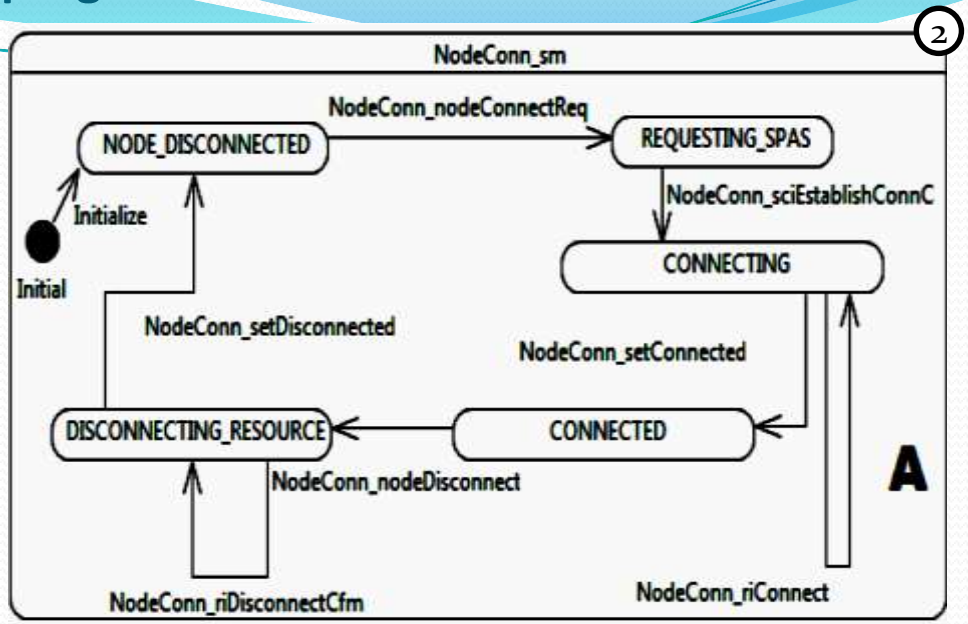


Step 2. Validation and verification of design solutions



Step 3: Generation and verification of program code

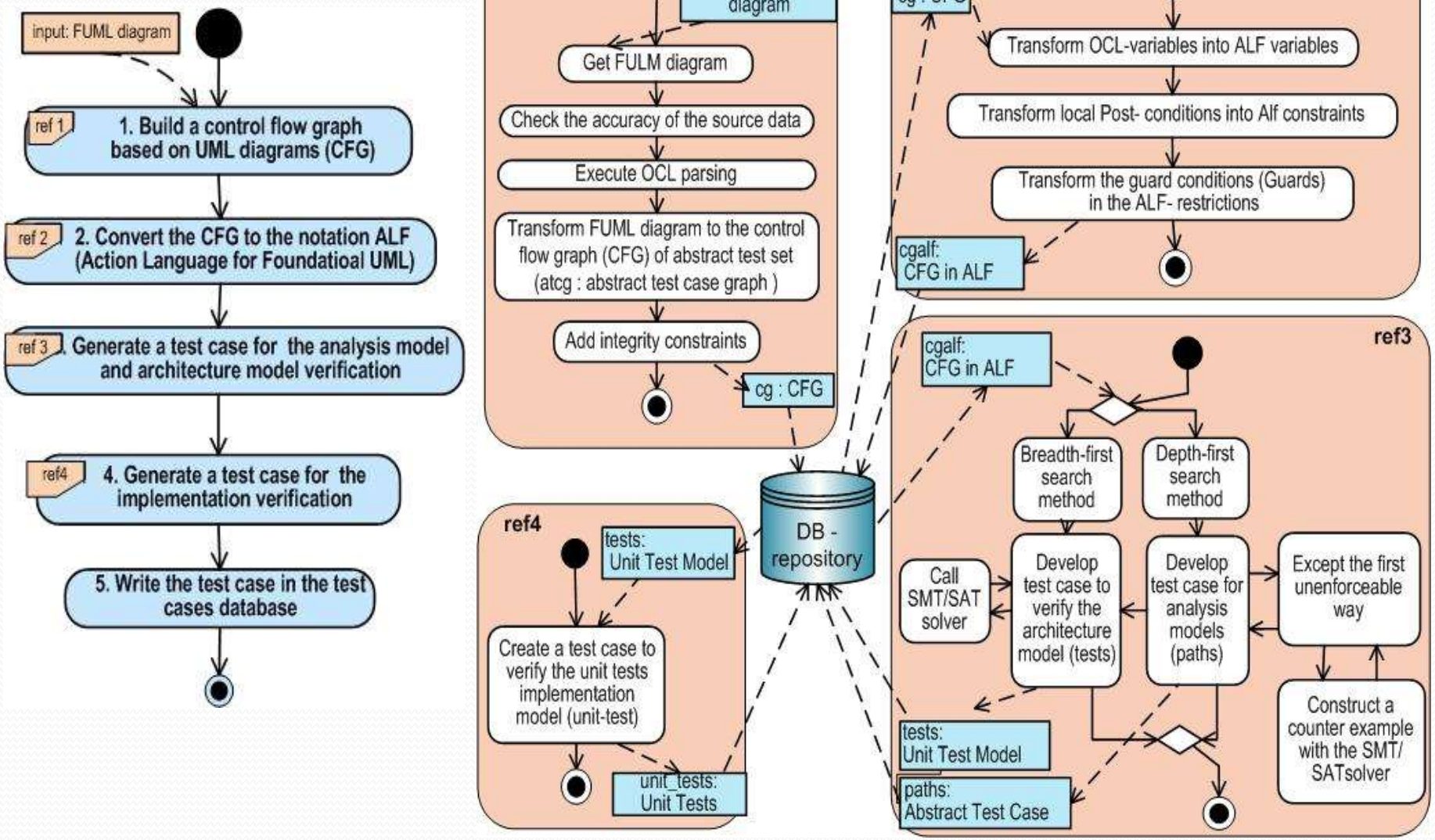




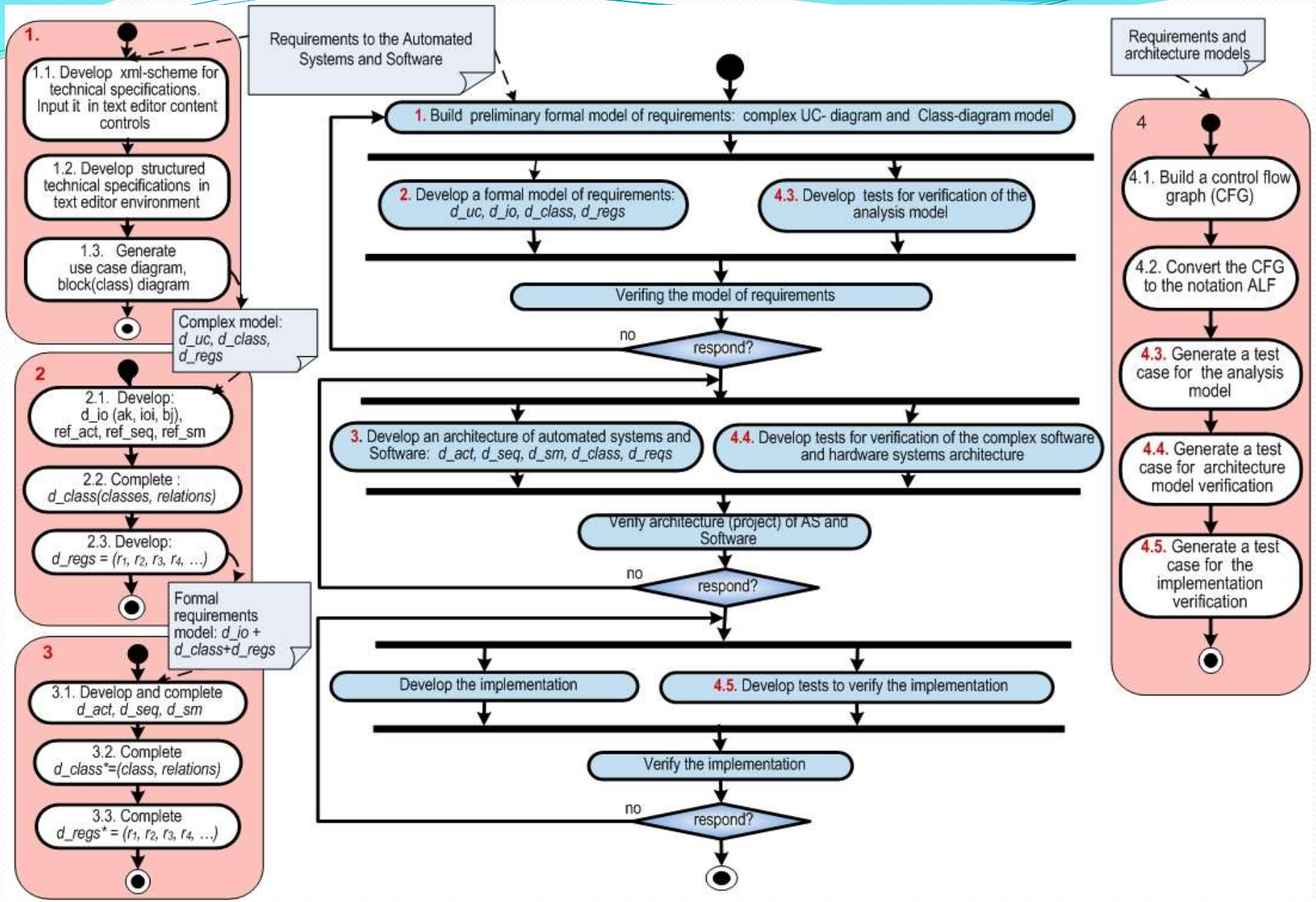
3

```
public NodeConn_riDisconnectCfm ( in portId : model::modelComponentView::DataTypes::Cello_Port,
in serverConn_r : model::modelComponentView::DataTypes::Cello_RiServerConn)
{
/*@inline('C++')
   "ENTER( \"NodeConn_riDisconnectCfm. \" );"
*/

if(serverConn_r.serverConnId == serverConnId && connHalf[serverConn_r.connHalf].portId == portId)
{
   connHalf[serverConn_r.connHalf].respondState =
        model::modelComponentView::DataTypes::NCCdataTypes::RespondState[RI_RESPONDED_CFM];

   model::modelComponentView::DataTypes::U8 secondHalf =
                                          NodeConn_getSecondHalf(serverConn_r.connHalf);
   if(connHalf[secondHalf].respondState !=
  model::modelComponentView::DataTypes::NCCdataTypes::RespondState[WATING_FOR_RESPONSE])
   {
      model::modelComponentView::DataTypes::NCCdataTypes::SciState x =
                                             SciHandler_RI.SciHandler_getSciConnectionState();
      if(x == model::modelComponentView::DataTypes::NCCdataTypes::SciState[SCI_ATTACHED])
      {
         NodeConn_setDisconnectingSpasState();
         SciHandler_RI.SciHandler_sciReleaseConnReq(spasConnId, serverConnId);
         // Simulate  NodeConn_sciReleaseConnCfm();
```

Step 3: Generation and verification of program code. Methods program code verification



Implementation Stages of software-controlled process of software and hardware systems development and verification



Concise Summary

Models and algorithms are developed

1. metamodel for formalization the requirements to software: UseCase, Interaction overview, class, requirements diagrams
2. metamodel for formalized description architecture software: fUML diagrams class, activity, sequence, state machine;
3. algorithms for development of formal models of requirements and architecture software;
4. algorithms for verification of models of requirements and architecture sofware by means of their modeling and analysis in the virtual fUML machine and execution of test scenarios built with the help of SMT / SAT solvers;
5. methodological recommendations for the implementation of the software-controlled process of development and verification sofware, using the developed models and algorithms.