

COMPUTATIONAL CLUSTER

Authors:

- Michał Padzik
- Michał Mierzyński
- Łukasz Napora
- Kamil Żak - [Contact](#)

Warsaw University of Technology
Software Engineering I

Description

1. Class Diagram is composed of several classes:

- a. Client – Using this class is initial step of computation. This class can solve problem that is defined in *mainProblem* variable. At the end, it present solution on screen, so it is last step too.
- b. Server – Independent module that serves to data transfer. It is a bridge connecting the modules at different points in the network. Provides a special log mechanism in order to prevent data loss.
- c. Task Manager – Main core of calculations distraction. Split problem into smaller ones and makes a solution from partial solutions.
- d. Computational Node – Part of cluster, gets a partial problem and turns it into partial solution.
- e. Problem – Abstract base class of computation unit. Its instance (concrete type at concrete moment of computation) is in circulation in working cluster.
- f. Partial Solution, Final Solution, Partial Problem, Main Problem – all of them inherit from Problem class and serve to present actual state of information flow. Each of them have some variable that define their type e.g. *string Name*; .

2. State Diagrams present a work of particular class from Class Diagram. Due to stability, architecture takes into account critical use-case's. All modules could be terminated due to lack of electricity, ignorance of users or unhandled exceptions (random errors), but all of this cases are not destroying for computation. Modules provides system of making a logs and restoring lost or crashed part of computation and resume solving the problem.

- a. The Client State Diagram shows the following state of the client. In the beginning, the client is initialized, then checks for wrong data. If data are correct execute next step, otherwise displays message error and stop working. The next step is finding a server. Again in case of error stop execution. The same thing is with sending data to founded server – if correct client waits for solution, if not – stop executing. Complete error messaging are taken into consideration (not shown on diagrams).
- b. The Server State Diagram shows the following state of the server. In the beginning, the server is initialized, if initialization is correct it starts waiting for action. Having received some data, it checks what type of data it received, and passes to the conditions associated with the data. Server can receive the following types of data: new problem, partial solutions, partial data and final solution.
- c. The Task Manager State Diagram shows the following state of the task manager. In the beginning, the task manager is initialized, if initialization is correct it starts waiting for data. Having received some data, it checks what type of data it received, and passes to the conditions associated with the data. Task manager can receive the following types of data: new problem and partial solutions.

- d. The Computational Node State Diagram shows the following state of the Computational Node. In the beginning, the Computational Node is initialized and it presents itself to the server (in order to inform what problem it solves). Then waits for partial problem (state field – Unused) and if data has come, solve the problem (state field – Working). In case of failure it informs the server about failure and again wait for problem to compute. If solving was ended with Success, sends partial solution to the server (state field – Succeed), and wait for other data (state field – Unused).
 - e. The Problem Instance State Diagram shows the following state of the Problem Instance. In the beginning, the Problem Instance is initialized by Client. Then is sent (by Client too) through Server to Task Manager. In the next step is sent to Computational Nodes (through Server) as a partial problems (list of them). Further partials are sent to Task Manager (when composed into Final Solution) and sent to Client in order to display on screen.
3. Event Flow Diagram shows the following events of Computational Cluster.
- a. Choose problem – The client chooses a problem which should be solved.
 - b. Problem available – The server approves the problem if it is available otherwise back to Choose problem and asks for choosing a new problem.
 - c. Put data – The client enters data to system.
 - d. Correct data – If the data are appropriate for the problem server starts converting it, otherwise asks client to send new data.
 - e. Find problem – The server looks for suitable task manager and sends data to it.
 - f. Divide for node – The task manager divides received data and sends partial data to the server.
 - g. Choose node – The server chooses available node to solve partial problem.
 - h. Node work – The node solves the partial problem it received from the server.
 - i. Get partial solutions – The server receives solved partial problem and sends it to the task manager.
 - j. Connect partials solutions – The task manager adds all partial problems and creates final result, which is sent to the server.
 - k. Show result – The client receives the final result and prints it.
4. Sequence diagrams describes basic flow of the communication between important parts of the Computation Cluster. They include handling errors in connection, lacks of power and other unexpected situations. All the requests are described in details in XML schema files attached to documentation.

Architecture

Class diagrams

Architecture of Computational Cluster

Class Diagram

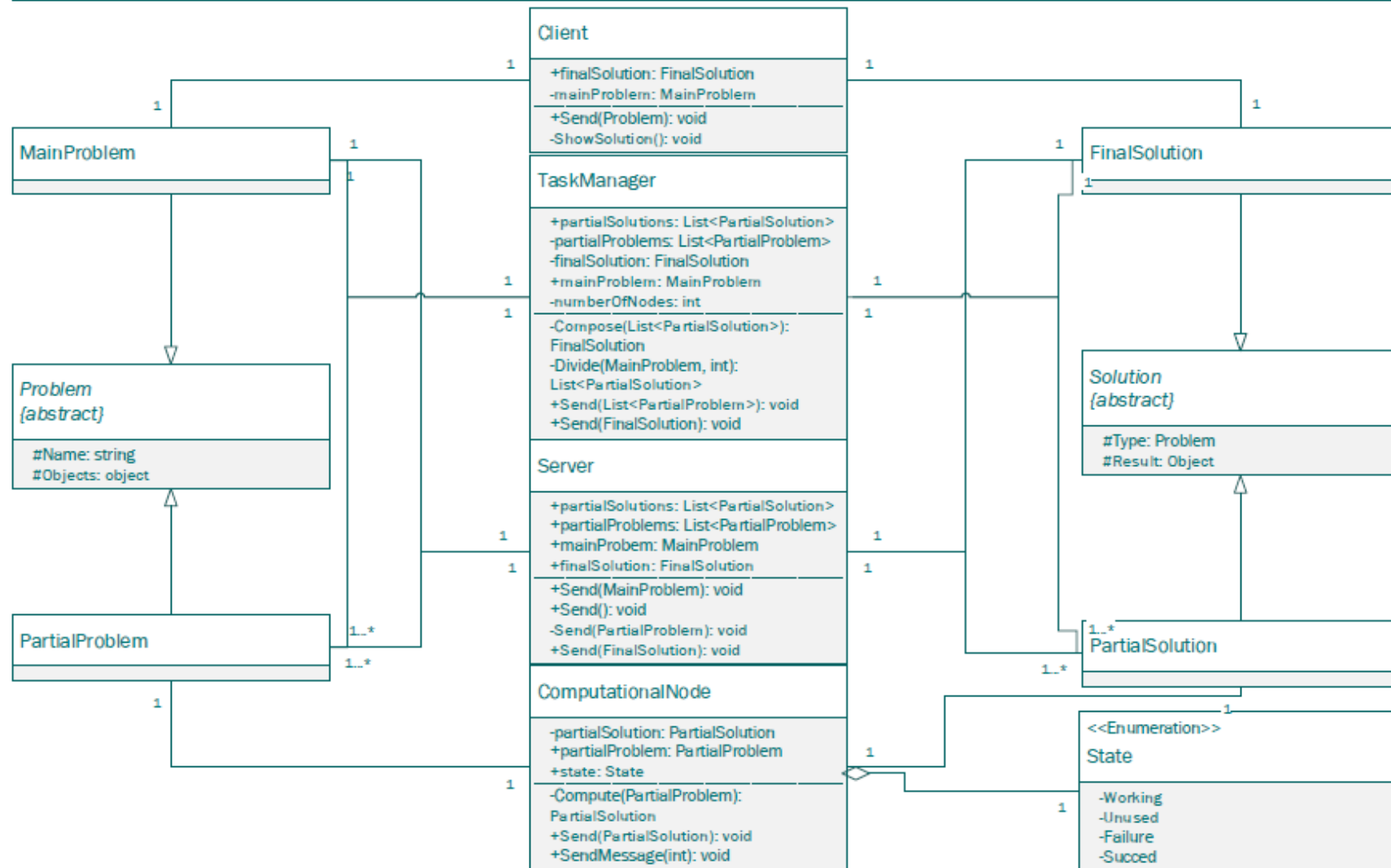


fig. 1

State diagrams

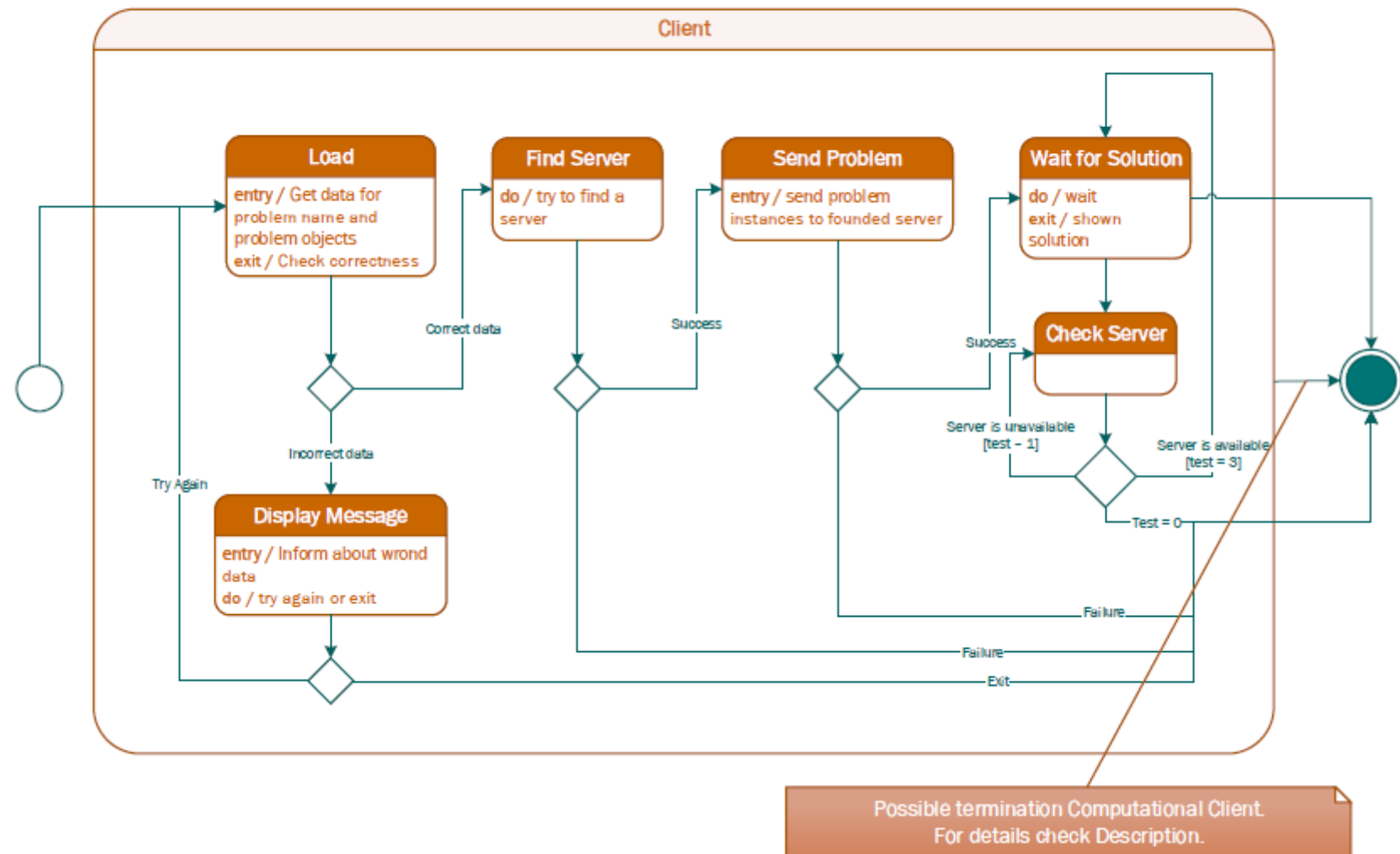


fig. 2

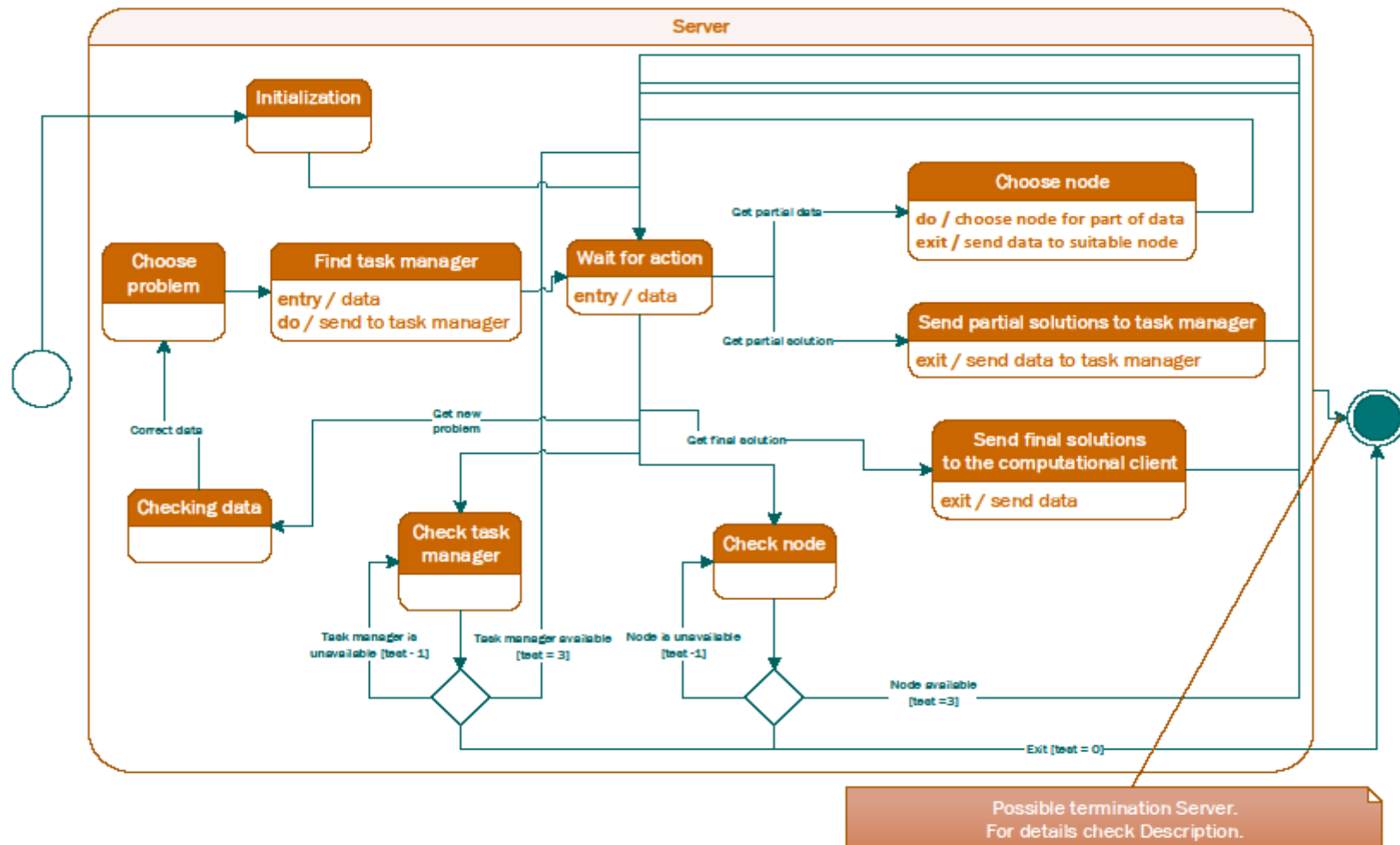


fig. 3

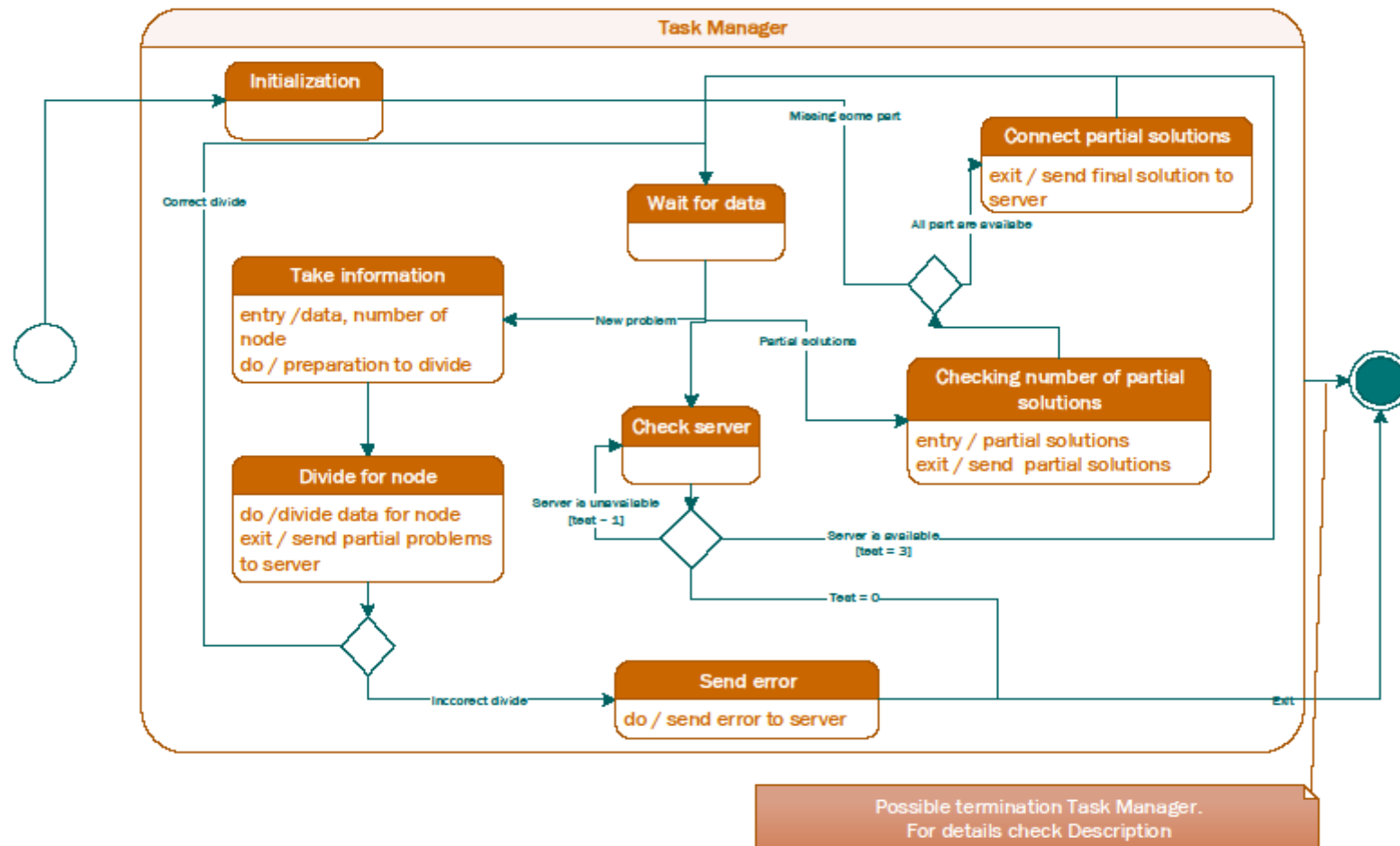


fig. 4

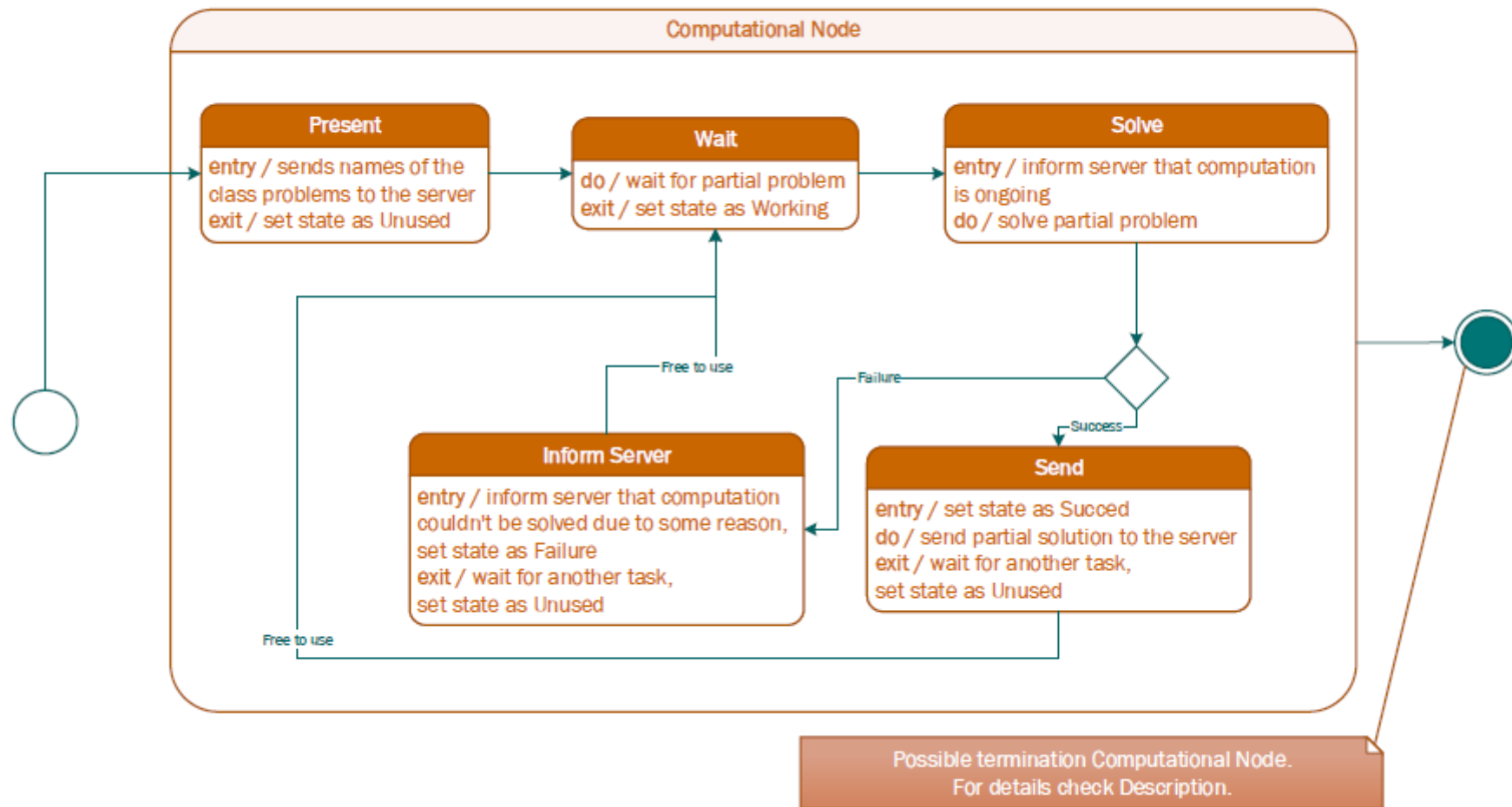


fig. 5

PI = Problem Instance

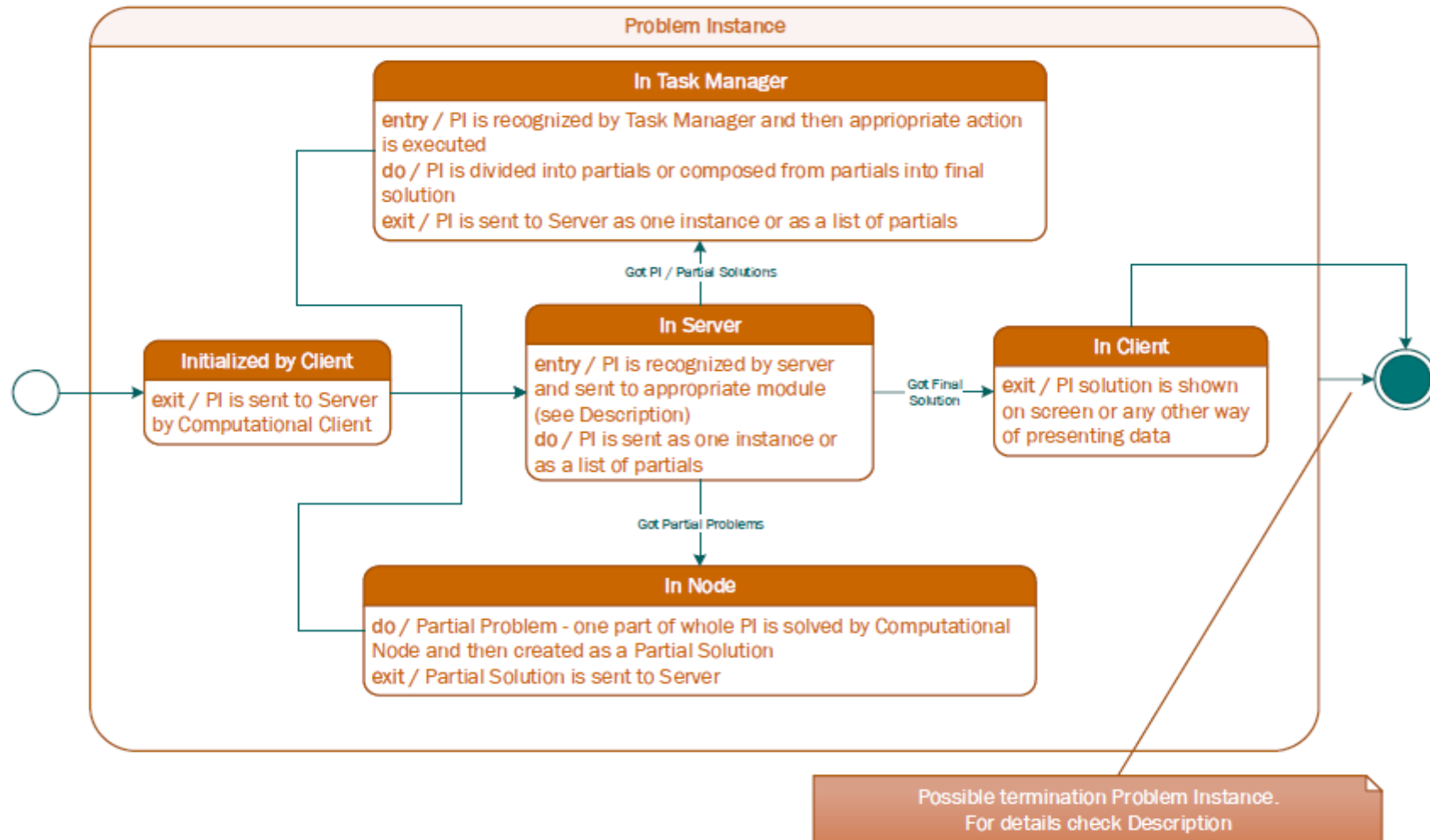


fig. 6

Event-flow diagram

Event-flow diagram

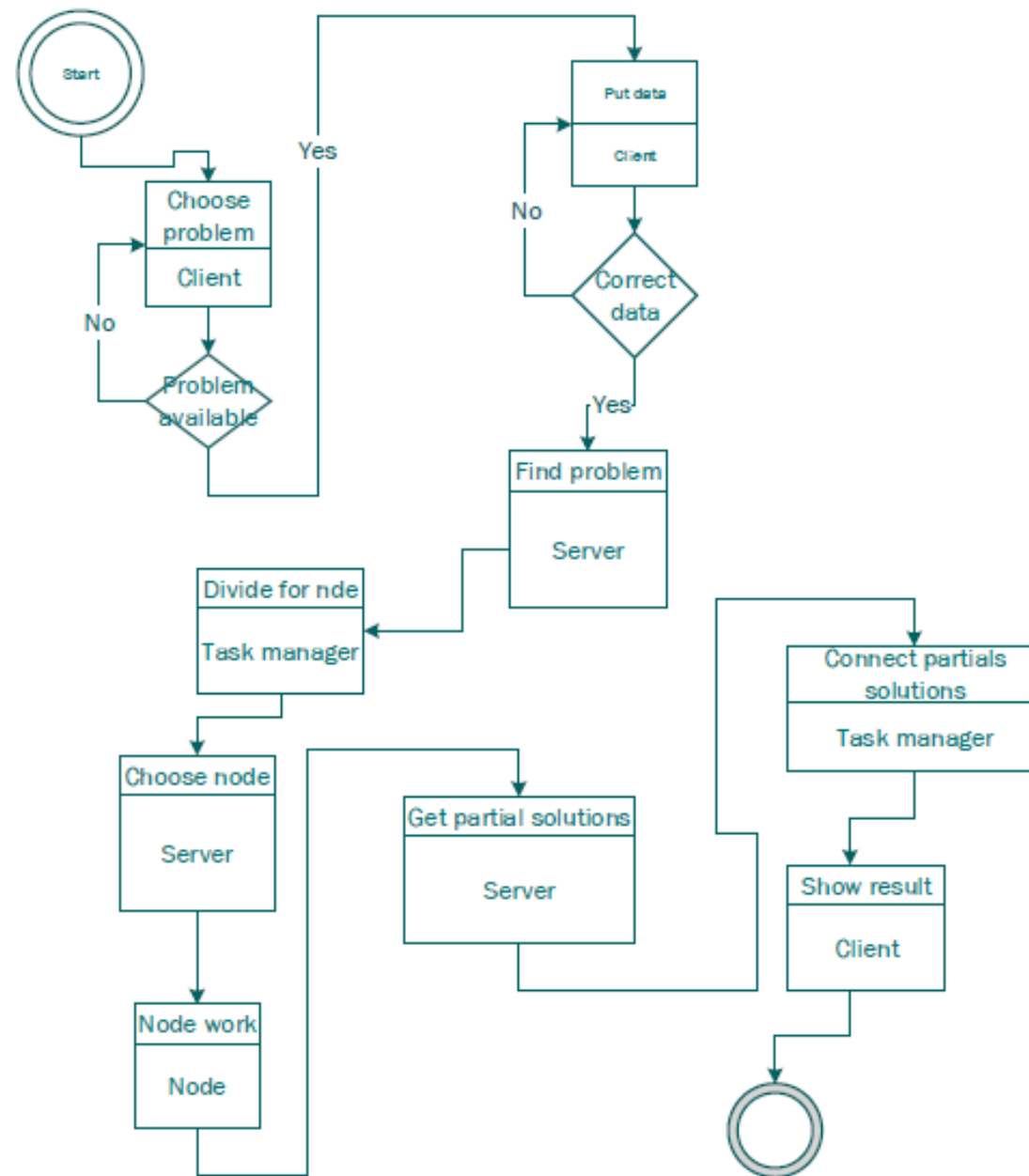


fig. 7

Sequence diagrams

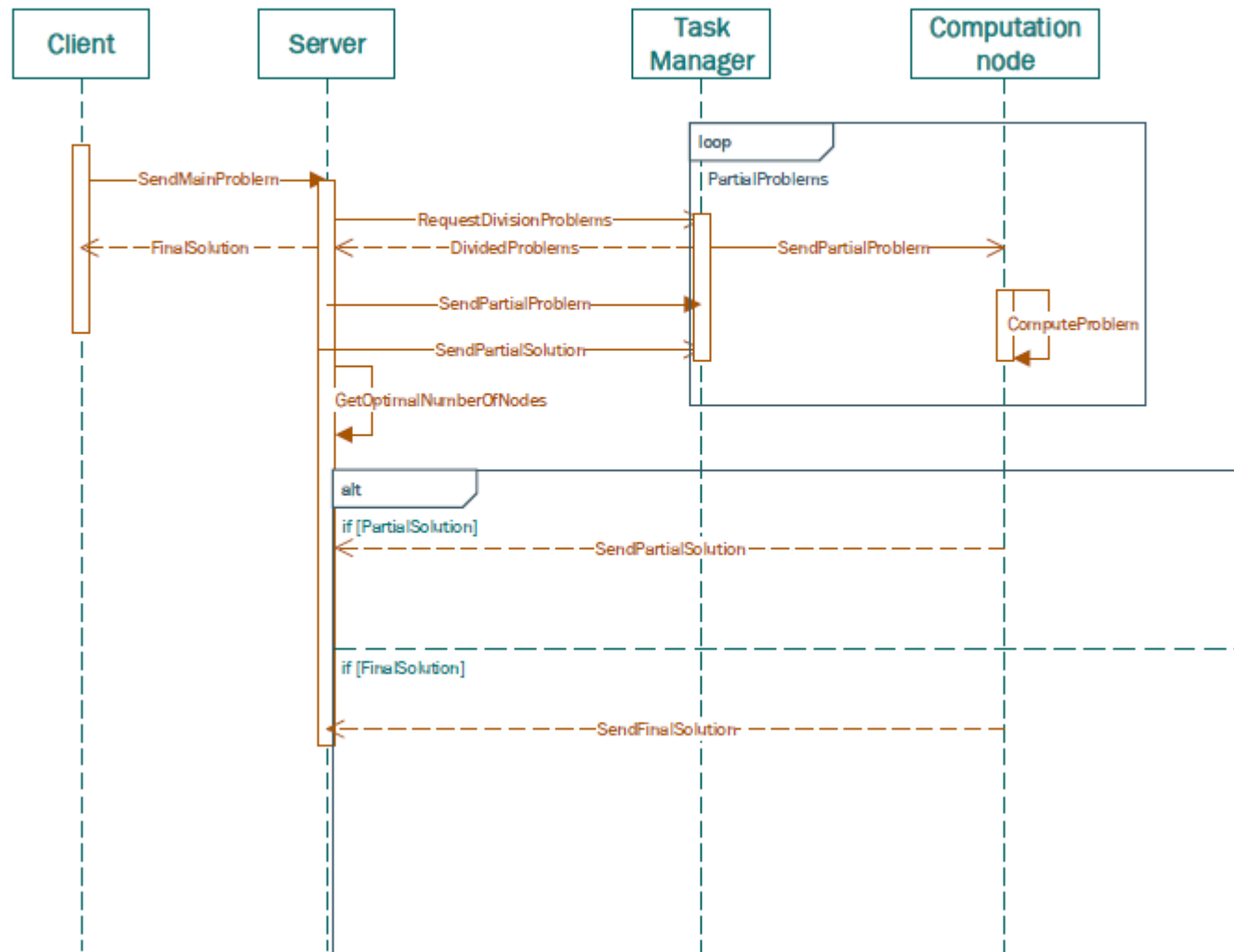


fig. 8

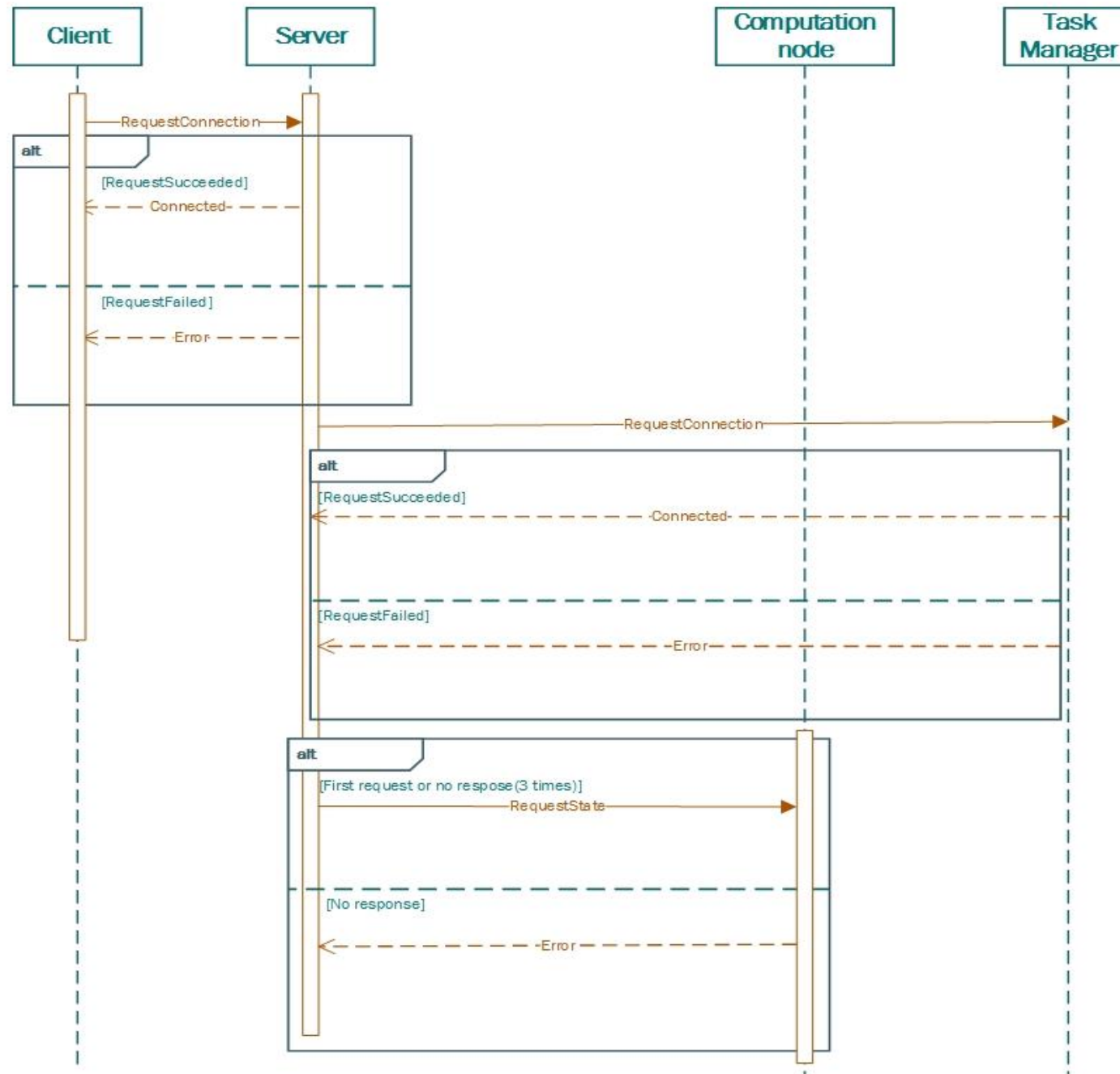


fig. 9

Others

Following things are enclosure in schema xml:

- Handling errors and responses
- Example description of TSP problem
- Communication protocol desing