

COMPUTATIONAL CLUSTER

Authors:

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

Warsaw University of Technology
Software Engineering I

Table of Contents

| | |
|---|----|
| Introduction | 2 |
| Description | 3 |
| Architecture..... | 5 |
| Class diagrams..... | 5 |
| State diagrams | 6 |
| Event flow diagrams..... | 13 |
| Activity diagrams..... | 14 |
| Sequence diagrams | 15 |
| Communication protocol desing..... | 16 |
| Input data format specification..... | 17 |
| Special system states description | 18 |
| Example class problem | 19 |
| | |
| fig. 1 | 6 |
| fig. 2 | 8 |
| fig. 3 | 9 |
| fig. 4 | 10 |
| fig. 5 | 11 |
| fig. 6 | 12 |

Introduccion

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 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.

- b. 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.
- c. 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)
- 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

Architecture

Class diagrams

Architecture of Computational Cluster

Class Diagram

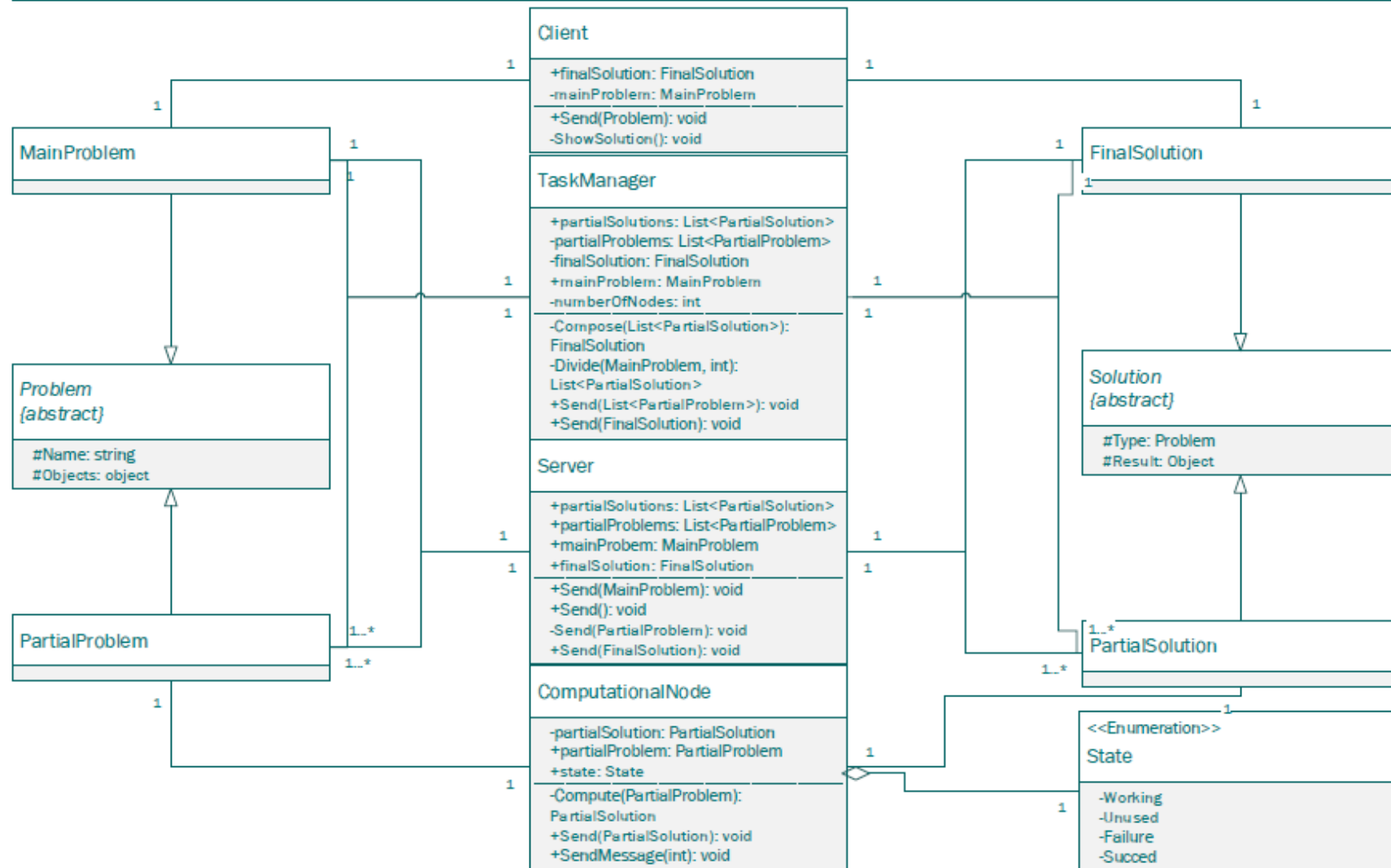


fig. 1

State diagrams

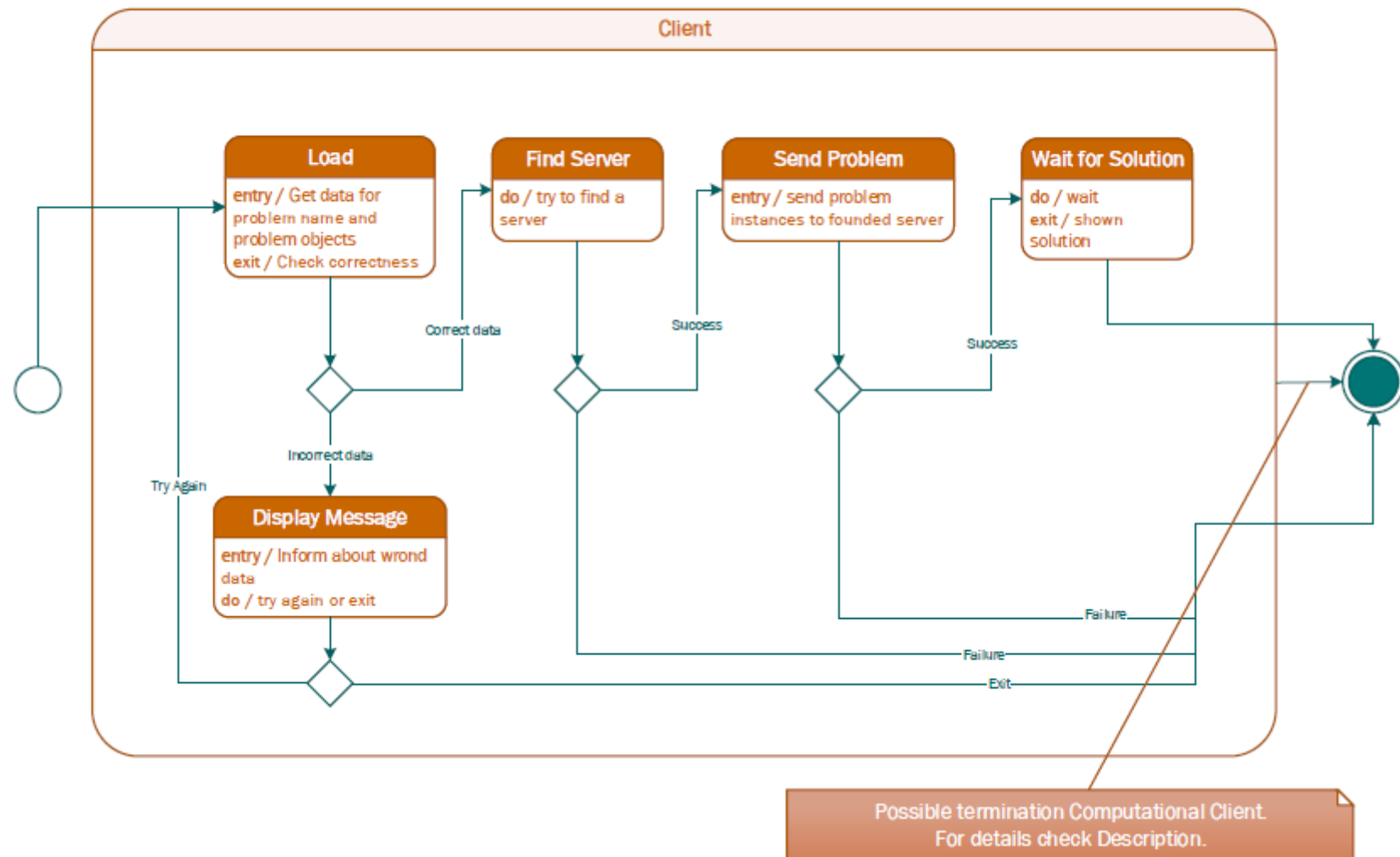


fig. 2

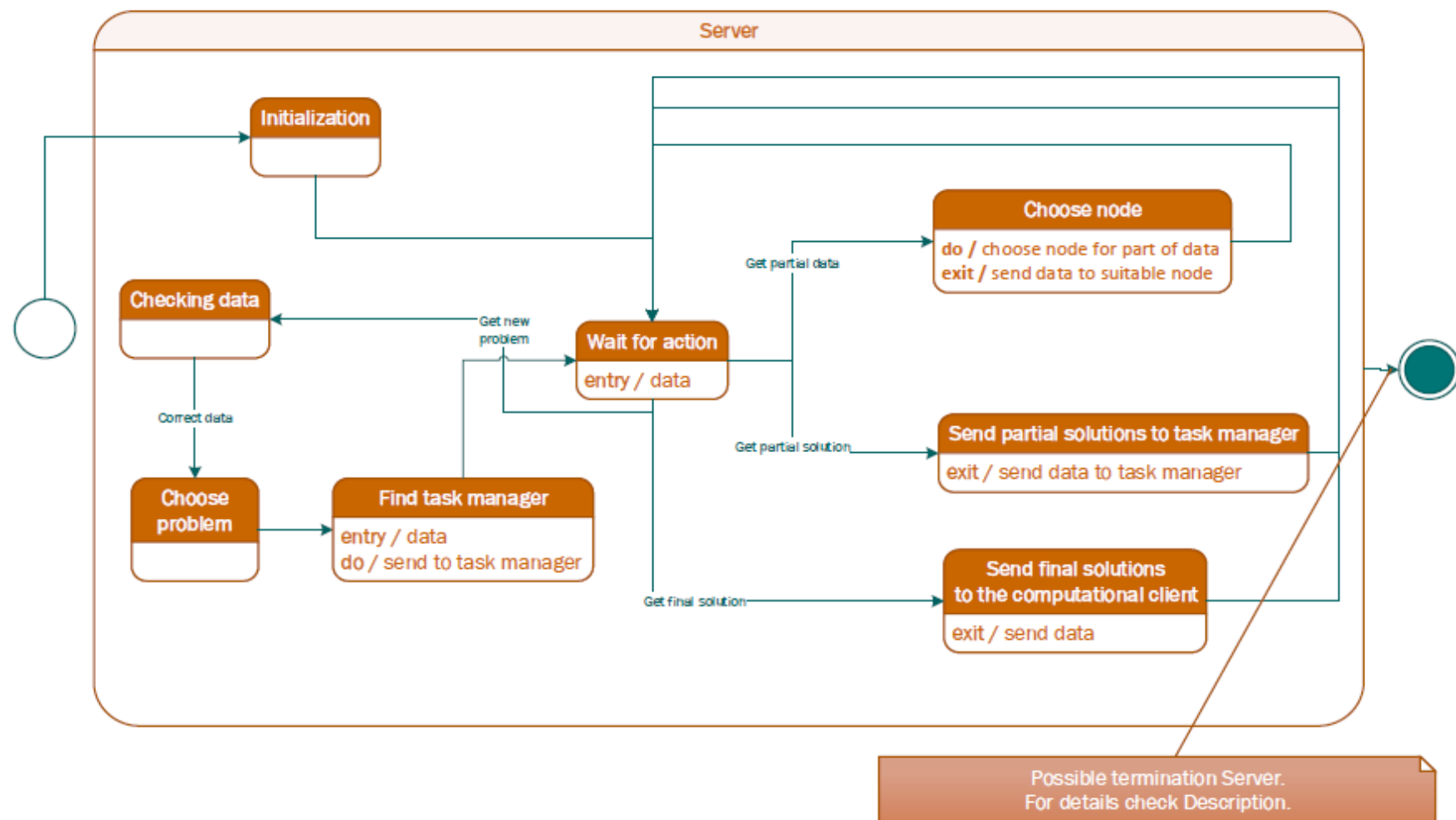


fig. 3

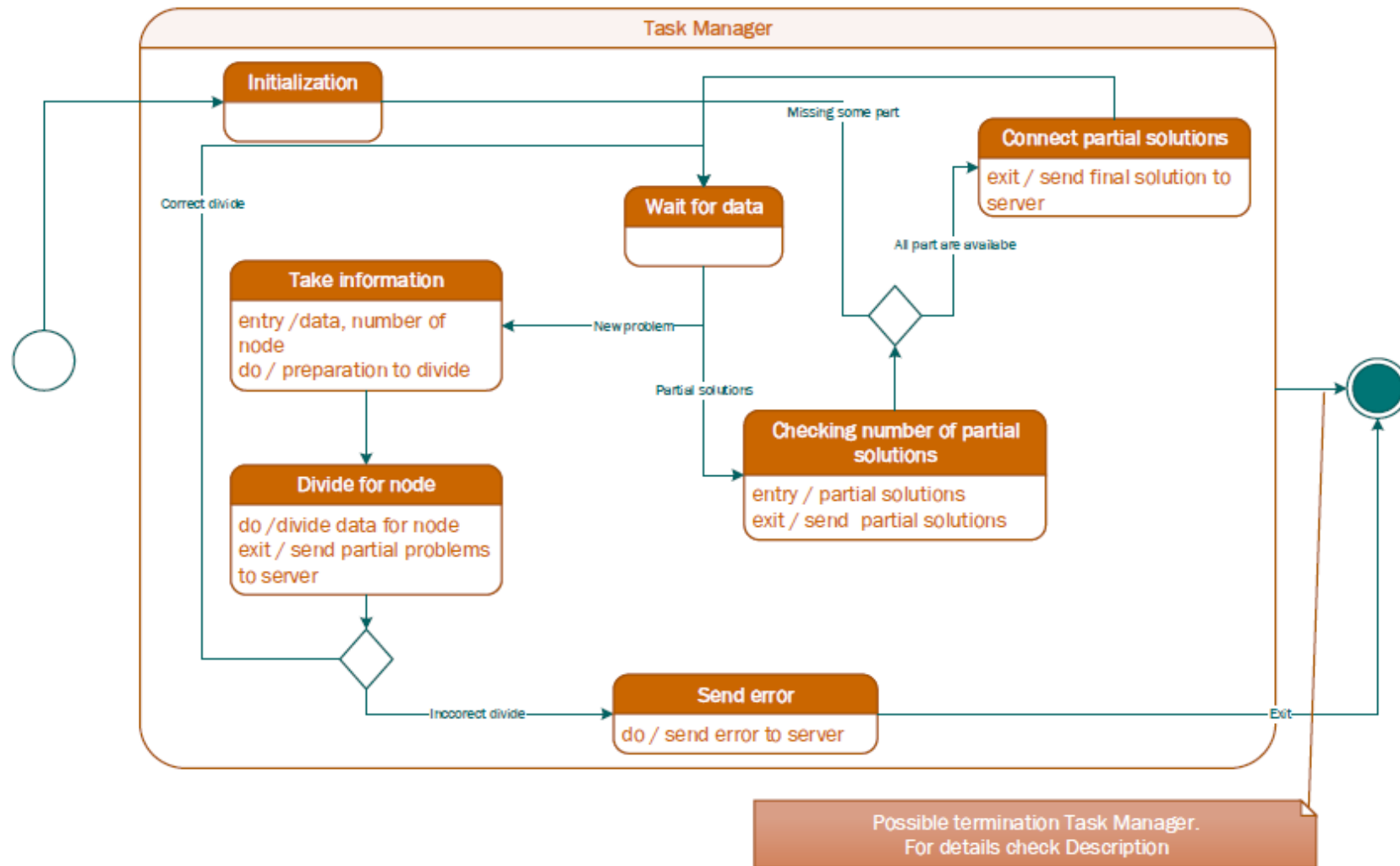


fig. 4

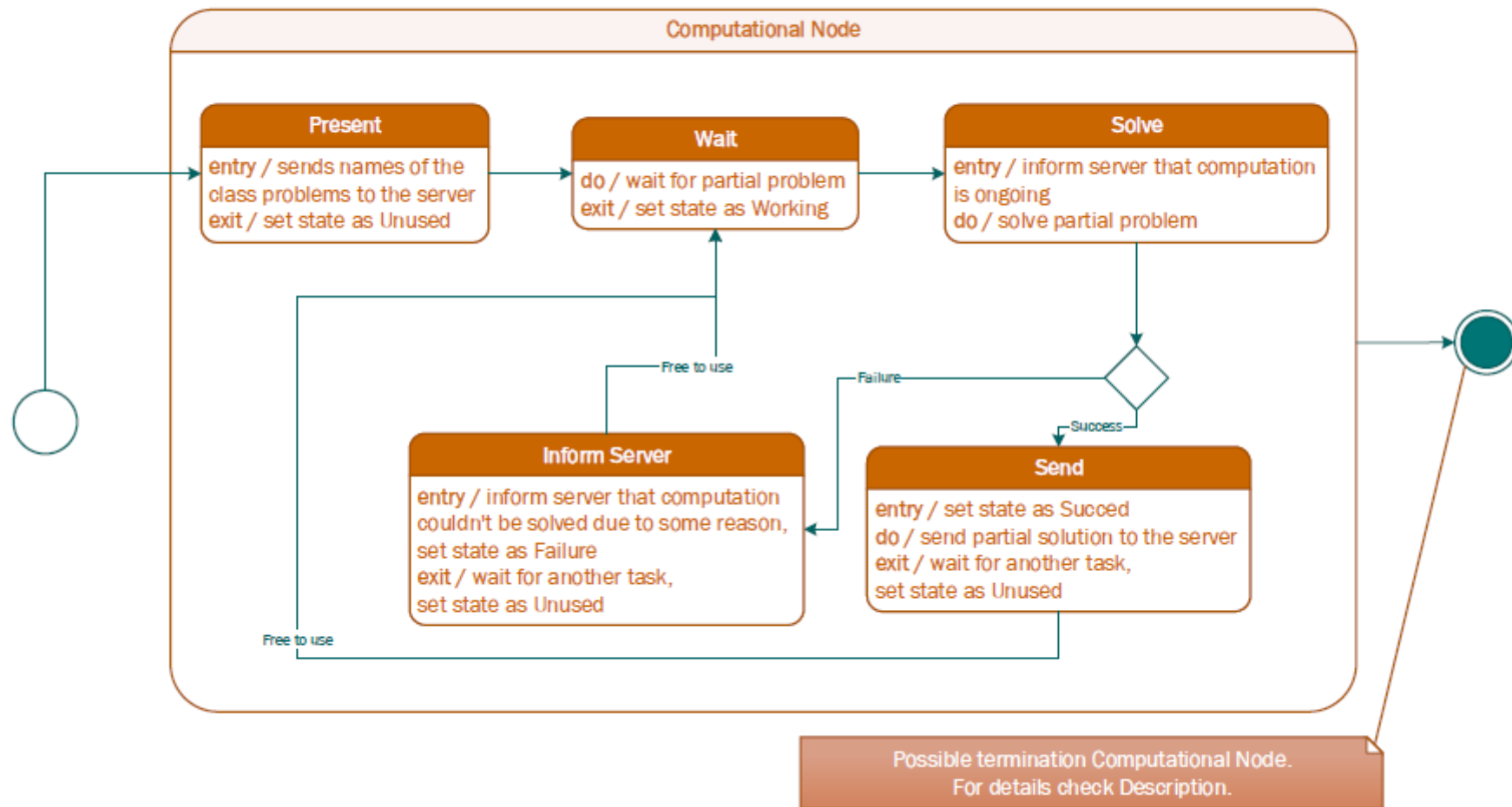


fig. 5

PI = Problem Instance

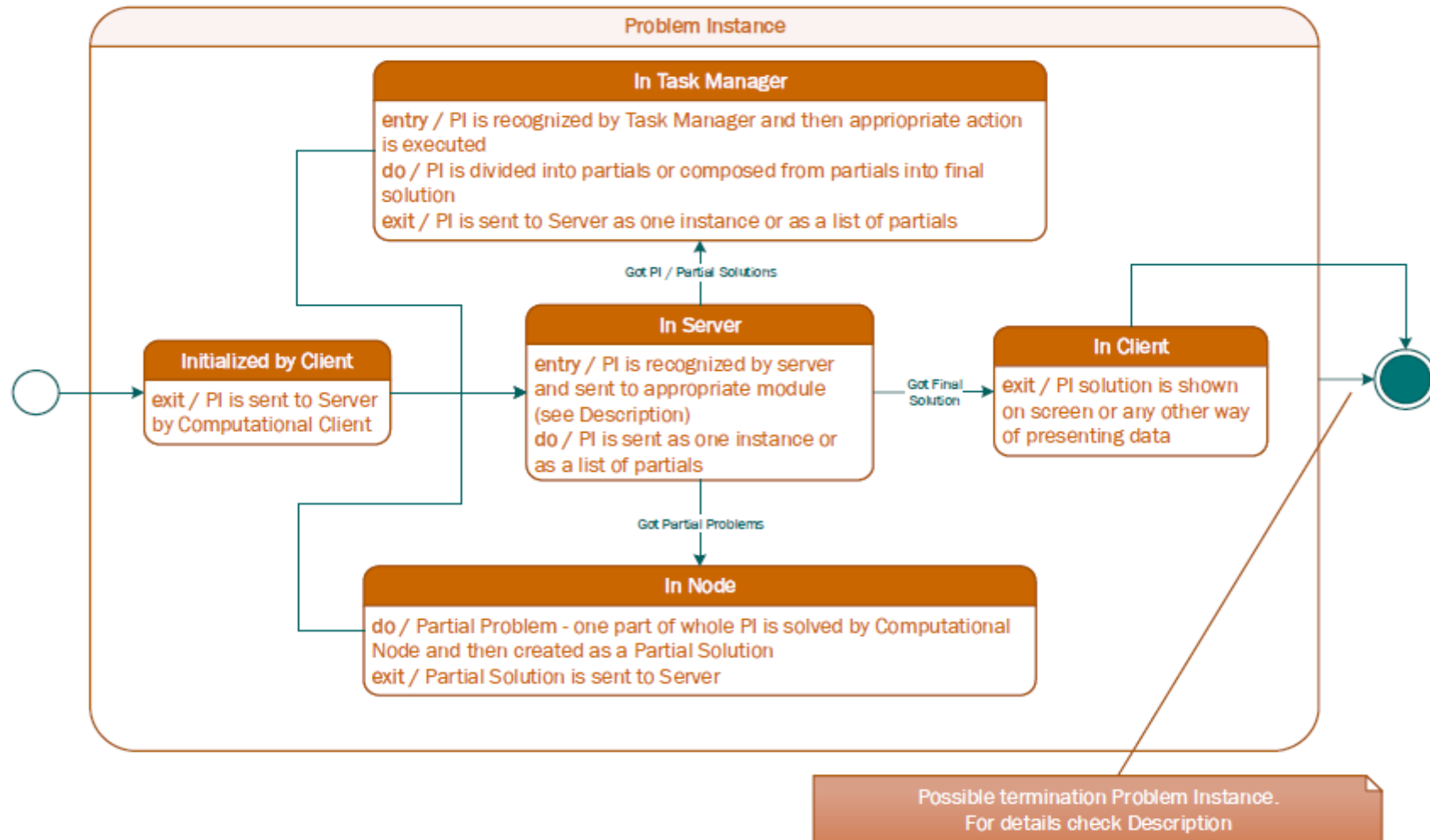


fig. 6

Event flow diagrams

Activity diagrams

Sequence diagrams

Communication protocol desing

Input data format specification

Special system states description

Example class problem