# Software Design Document
# Team 2.2

## Introduction

### Purpose of the System

The goal is to create two add-ons for the software VANTED (http://vanted.org) that implement fast stress minimization and a multilevel graph drawing framework respectively. The add-ons are to be developed for the class “Softwareprojekt” in the summer term 2019 at the University of Konstanz. This document lays out the requirements for the add-ons.

### Design Goals

MLF: Development of an add-on for the VANTED software that provides a multilevel framework. It shall provide the following:

- Performance that is not asymptotically slower than the ogdf::ModularMultilevelMixer implementation.
- Possibility to choose and customize a layout algorithm (via the VANTED graphical interface) that will be applied on every level of the graph.
- Provide an interface for adding other coarsening and placement methods.

SM: Development of an add-on for VANTED that adds an stress minimization layout to it. It shall provide the following:

- The add-on shall be able to work with large graphs.
- It shall not perform asymptotically slower than ogdf::StressMinimization.
- The add-on shall be thoroughly tested and benchmarked.

### Definitions, Acronyms and Abbreviations

- **VANTED** refers to “Visualisation and Analysis of Networks containing Experimental Data”.
- **Add-on** refers to an extension of the VANTED software that can be loaded by VANTED and enhances it’s functionality.
- **OGDF** refers to “Open Graph Drawing Framework” (http://www.ogdf.net/doku.php)
- **GUI** refers to “Graphical User Interface”
- **MLF** refers to “Multilevel Framework”
- **SM** refers to “Stress Minimization”
- **Large Graph** refers to graphs consisting of more than 1000 vertices.

## References

- **[Spec]** Softwareprojekt2019_Gruppe2Thema.pdf
- **[V]** VANTED source code (https://bitbucket.org/vanted-dev/vanted/src/master/)
- **[Meet]** The initial meeting on the 25 April 2019
- **[Meet2]** The supervisor meeting on the 2 May 2019
- **[Eval]** Bartel2011_Chapter_AnExperimentalEvaluationOfMult.pdf
- **[SRS]** Software Requirements Document

## Overview

This document specifies how the MLF and the SM-algorithm shall be designed and is categorized into three chapters. The first gives an overview of the contents of the SDD. The second chapter gives a description of the current architecture and the third chapter describes the proposed software architecture for the application.

# Current Architecture

In the current state none of the add-ons is implemented. VANTED already has an add-on interface that will be used.

# Proposed Software Architecture

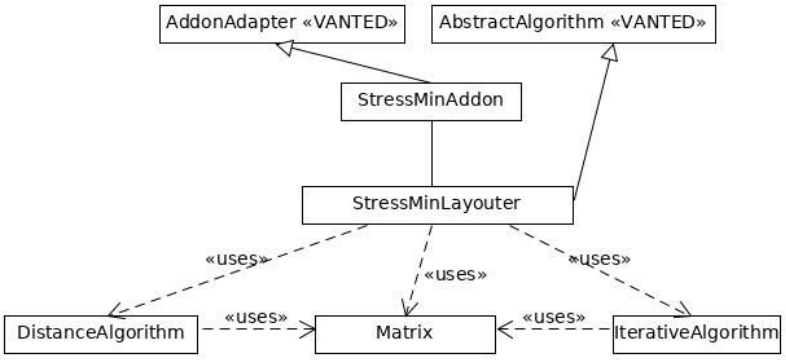
## Overview

MLF: Implementation of an add-on for VANTED that adds a multilevel framework layouter. The layouter shall be modular in that it shall be possible to run it with different mergers, placers and level layouters. This shall be achieved by providing an interface that the used mergers and placers must implement. The interaction with the layouters happens through the *Algorithm* interface that already exists within VANTED. A data structure for storing and manipulating the coarsening levels of the graph shall be implemented. It is to be used by the implementers of the merger and the placer interfaces and needs to implement the *Graph* interface so that the level layouters can be used on it.

SM: Implementation of a network layout method for undirected graphs (“Stress Minimization”). A simple iterative variant as a basic version is implemented.
Initially unweighted shortest paths between all node pairs calculated, then an initial layout is determined, and then iteratively improved until a termination criterion is reached.

# Class Diagram

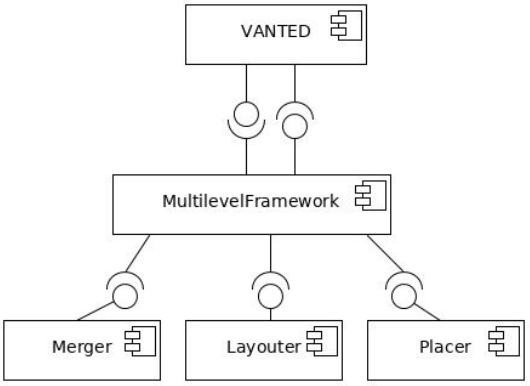
## Stress Minimization:




The main class of the SM-algorithm is the “StressMinAddon”-class. It registers the layout algorithm. The algorithm uses the graph distance matrix which gets calculated by the “DistanceAlgorithm”. Since a lot of operations are done on matrices there is a separate “Matrix” class.  The “IterativeAlgoritm” is used to calculate the next iteration step and outputs the new positions for the vertices of the graph.

# Subsystem Decomposition

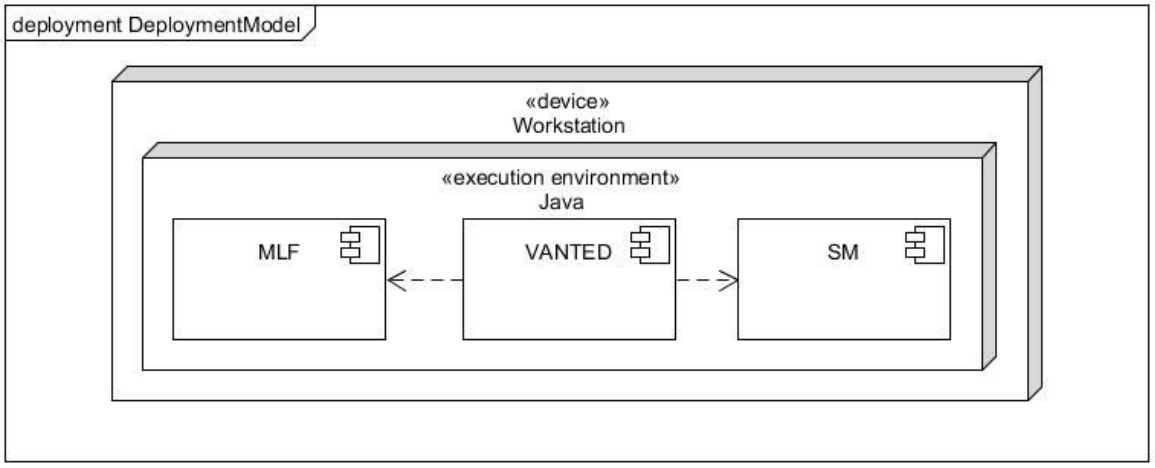
## Multilevel Framework




The MultilevelFramework component uses three other components. The Merger is used to coarsen the graph, the Layouter is used to layout the individual levels and the Placer arranges the “re-inserted” vertices of lower levels of coarsening.

# Hardware/Software Mapping

In this section, it is described how the subsystems are assigned to the related hardware.




Both add-ons run on the Java Virtual Machine as extensions for the VANTED software. They provide the framework and stress minimization functionality respectively to VANTED. These functions can be called from VANTED. Thus the user interface is also provided by VANTED.

## Persistent Data Management

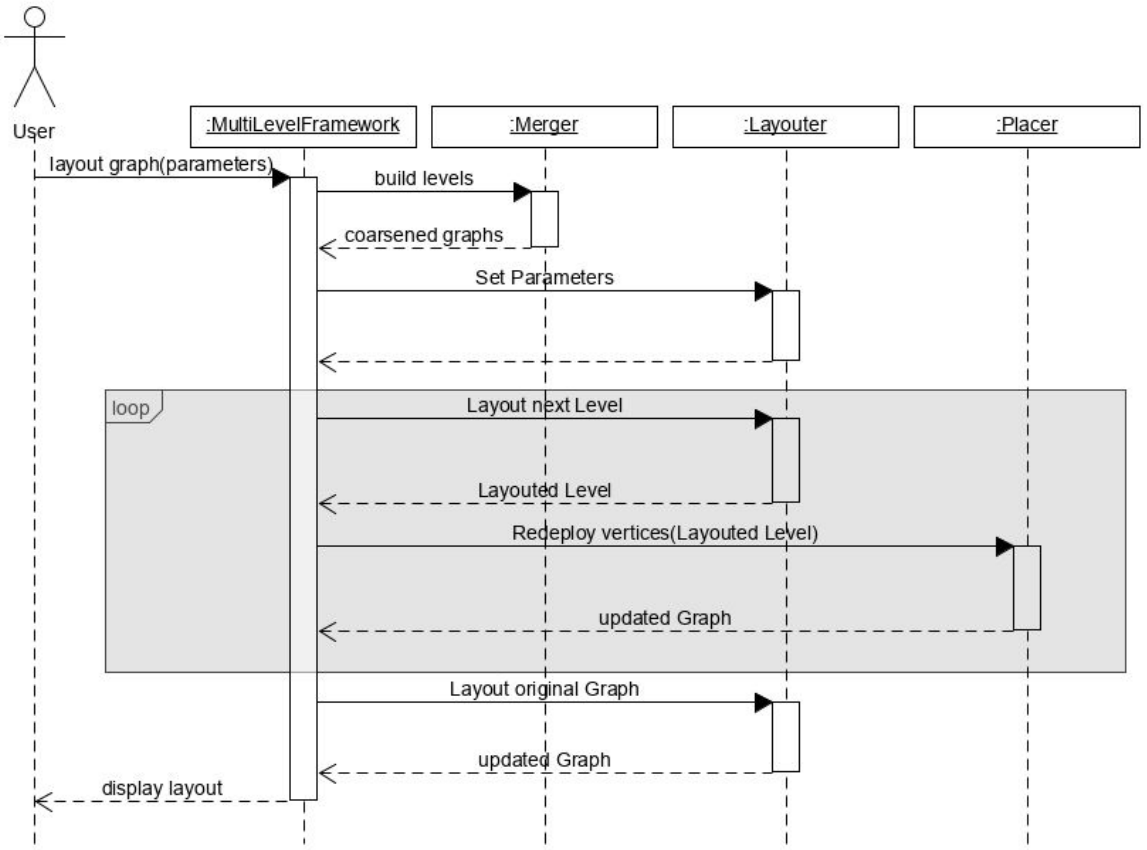
Data Management is controlled by VANTED, raw graph data and layouted graphs can be saved and opened as for example CSV or PNML format.

## Global Software Control

The software gets controlled by the VANTED-GUI. The add-ons provide a set of parameters to VANTED via an interface. VANTED then displays these parameters (depending on the type of data) as e.g. sliders to the users. They can make some adjustments. When the framework or the algorithm is run they can read the possibly changed parameters and adjust their behaviour accordingly. As a part of VANTED a user can stop any running algorithm or the framework at any time. If this happens the stress minimization shall return the layout of the graph at the current iteration step whereas the multilevel framework shall return the graph to its original state.

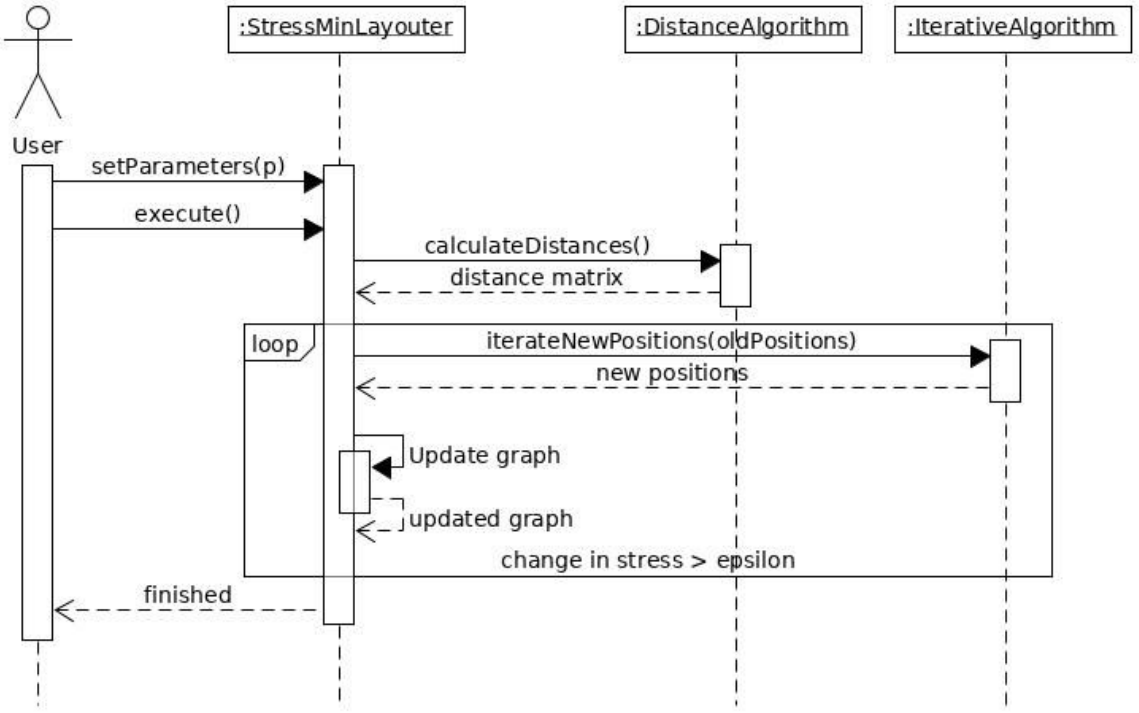
## Interaction of subsystems and initiation of requests

### Multilevel framework




The user can set up parameters for the MLF and the layouter and start the layouting process. The MLF then proceeds to to build the coarsening levels. The parameters that the user set up for the layouter are passed to it. Then the layout of the levels is carried out in a loop for the different levels, starting with the coarsest one. After each level layout (except for the last one) the nodes from the previous level are placed back into the graph using the placer. At last, the layouter is run on the original graph.
The user can also decide to abort the multilevel framework by clicking the respective button in VANTED (not shown in the sequence diagram). This will restore the original graph layout.

Stress minimization

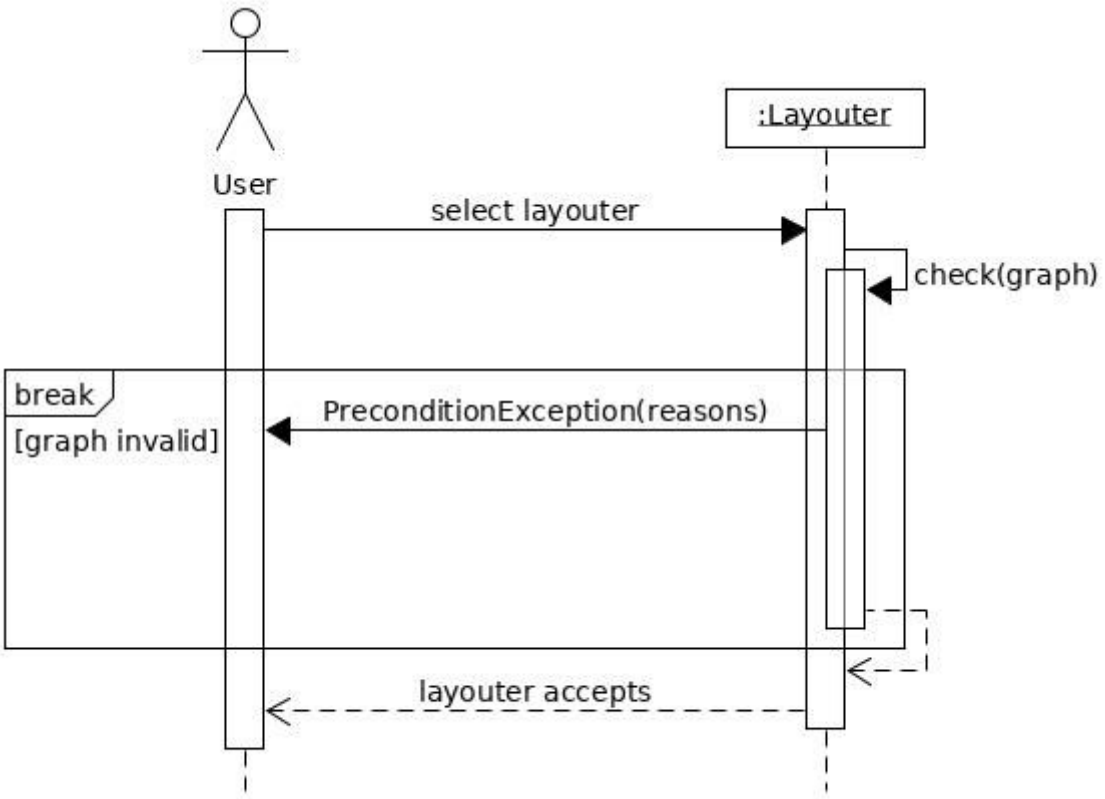



The diagram describes the normal use procedure:
First the user uses the VANTED GUI to select the algorithm and change its parameters. After being started by the user the algorithm first calls the “Distance” algorithm class to calculate the shortest distance between all vertices.Then it tries to move the vertices to improve the layout by calculating the new positions using the “IterativeAlgorithm” and updates the graph accordingly. This procedure is repeated until the change in the stress function before the vertices were moved and after it are smaller than a user given threshold called “ε”. If this threshold is reached the algorithm stops and delivers the result to the user. If the layouter is stopped by the user, it just returns without resetting the graph. Thus its possible to stop the layouter if the result is good enough and get an immediate result.

## Boundary conditions

Start-up and shutdown are managed by the VANTED software and a standard use case with Start-Up and Shut-Down for both add-ons is described in the sequence diagrams above But there are some possible errors. These cases are described below.

## Invalid Graph




The user selects the stress minimization algorithm or the multilevel framework (in the diagram denoted as the generic class “Layouter”)  in the layout list. In this case VANTED calls the “check” method of the layouter. The layouter checks whether the current graph meets the required preconditions i.e. is valid . In our case that means e.g. whether the graph is undirected. If the graph is not valid a “PreconditionException” is thrown and the user is not able to execute the algorithm or set its parameters until these preconditions are met. If they are met, the method returns silently and the user can proceed.