

Ikarus::KirchhoffLoveShell
::calculateVectorImpl

Ikarus::LinearElastic
::calculateVectorImpl

Ikarus::NonLinearElastic
::calculateVectorImpl

Ikarus::Traction::calculate
VectorImpl

```
graph LR; A[Ikarus::KirchhoffLoveShell::calculateVectorImpl] --> D[Ikarus::Traction::calculateVectorImpl]; B[Ikarus::LinearElastic::calculateVectorImpl] --> D; C[Ikarus::NonLinearElastic::calculateVectorImpl] --> D;
```

The diagram illustrates a design pattern where three different classes (KirchhoffLoveShell, LinearElastic, and NonLinearElastic) implement a common interface or base class method (calculateVectorImpl). These implementations are then used by a Traction class, which has its own calculateVectorImpl method. The Traction class's method likely delegates the calculation to the appropriate implementation based on the context or state.