私はソースコードの良し悪しや可読性に興味があります．大学のプログラミングの課題で，バグの除去に困っていた友人のソースコードを見たことがあります．その友人のソースコードは同じ課題に対するものにもかかわらず，実装方法が私のソースコードと全く異なっていました．その際，私と友人の実装方法のどちらが優れているのか，そして，その実装方法の差の原因は何だったのか，私にはわからず，疑問に思いました．そのため，どういったソースコードが良い，または悪いとされるのか，また，ソースコードの評価に差が生まれる原因が何であるか，研究を通して知りたいと考えています．

そこで，私は情報科学研究科に進学し，ソースコードを用いたプログラマーの練度を判定する方法について研究したいと考えています．プログラマーの練度によってプログラムの結果に差はないとしても，ソースコードには可読性をはじめとして様々なところに差が生まれ，良し悪しの評価が変わります．しかし，そういった差が生まれる原因は人による判定では発見しにくいものです．そこで，ソースコードを様々な観点から数値化できるメトリクスを用いて，機械学習によるプログラマーの練度の判定を行いたいと考えています．また，この判定を行うと練度の差によってソースコード中に変化が生じている点が明確になるため，練度の低いプログラマーが作る，悪いソースコードに対して改善点も提示したいと考えています．そうすることで，改善点が提示されたプログラマーはそのソースコードを改善することができるだけでなく，これから書くソースコードにここで提示された改善点を活かせるのではないかと考えています．

先行研究ではプログラミング上級者と初級者のソースコードにおける特徴の違いを数値化して機械学習を用いて分析することで，異なるソースコードに対してプログラマーが初級者，上級者のどちらであるか判定を行い，初級者であれば上級者との違いから考えられる修正方法を提示していました[1]．しかし，プログラマーは初級者と上級者だけではなく，どちらでもない中級者が存在します．先行研究の手法では中級者と考えられるソースコードに判定を行うと，必ず初級者または上級者だと判定されてしまいます．また，[1]では初級者と判定されたものに対して，上級者との違いをもとに修正方法を提示していますが，この場合，一度にたくさんの修正方法が提示されてしまいます．初級者はこのように修正方法が提示されると，どの修正から行えばよいのか当惑してしまうことが考えられます．そこで中級者も判定結果に加えると，中級者は初級者でも上級者でもない中級者だと判定されます．また，初級者と判定されたものには上級者との違いから特に重要だと考えられる修正方法，そして中級者と判定されたものには上級者との違いから考えられる修正方法を提示すれば，一度に提示される修正方法が少なくなるため，どこから修正していけばよいのかわかりやすくなり，プログラマーは着実に成長することができるのではないかと考えています．

[1]は上級者，初級者を決めるために先行研究[2]で研究された，初級者・上級者間で違いのある特徴量をベクトルに変形して利用し，機械学習を行いました．そこで，中級者を新たに設定するアプローチとして，それぞれの特徴量が初級者と上級者の中間値をとったものを中級者とする手法，そして新たに中級者のソースコードから特徴量を入手し，学習する手

法を考えました.

これらの手法により初級者，上級者に加え，中級者も判定します．そして初級者に対しては上級者との違いから特に重要だと考えられる修正方法，中級者に対しては上級者との違いから考えられる修正方法を提示し，プログラマーの成長を促せると考えています.

[1]槇原 啓介，ソースコード特徴量を用いた機械学習によるソースコードの良否の判定
[2]堤 祥吾，プログラミングコンテスト初級者・上級者間におけるソースコード特徴量の比較