Model:
The following C++ program was used to run the simulations. It takes user-defined parameters, applies random environmental effects and genetic drift, and calculates the number of generations required to reach the divergence threshold. The model was run for both surface and cave-like conditions with different parameter sets.

```
#include <iostream>
#include <random>
#include <cmath>
#include <vector>
#include <numeric>

using namespace std;

struct Parameters {
    double mutationRate;
    double environmentalPressure;
    double fitness;
    double reproductiveVariability;
    int populationSize;
    double divergenceThreshold;
};

double generateGeneticDrift(mt19937& gen, const Parameters& params) {
    double stdDev = (params.mutationRate * params.fitness) / sqrt(params.populationSize) *
params.reproductiveVariability;
    normal_distribution<double> driftDist(0.0, stdDev);
    return driftDist(gen);
}

int calculateGenerations(const Parameters& params, mt19937& gen,
uniform_real_distribution<>& envDist) {
    double divergence = 0.0;
    int generations = 0;

    while (divergence < params.divergenceThreshold) {
        double geneticDrift = generateGeneticDrift(gen, params);
        double environmentalEffect = envDist(gen);
        divergence += fabs(geneticDrift + environmentalEffect);
        generations++;
    }

    return generations;
}
```

```
double runSimulations(const Parameters& params, int numTrials) {
    mt19937 gen(random_device{}());  // Use random_device for better seed
    uniform_real_distribution<> envDist(-params.environmentalPressure,
params.environmentalPressure);

    vector<int> results;
    results.reserve(numTrials);

    for (int i = 0; i < numTrials; i++) {
        results.push_back(calculateGenerations(params, gen, envDist));
    }

    double sum = accumulate(results.begin(), results.end(), 0.0);
    return sum / numTrials;
}

int main() {
    Parameters params;
    int numTrials;

    cout << "Enter mutation rate (e.g., 0.005): ";
    cin >> params.mutationRate;

    cout << "Enter environmental pressure (e.g., 0.03): ";
    cin >> params.environmentalPressure;

    cout << "Enter fitness (e.g., 1.0): ";
    cin >> params.fitness;

    cout << "Enter reproductive variability (e.g., 1.0): ";
    cin >> params.reproductiveVariability;

    cout << "Enter population size (e.g., 3000): ";
    cin >> params.populationSize;

    cout << "Enter divergence threshold (e.g., 1.0): ";
    cin >> params.divergenceThreshold;

    cout << "Enter number of simulation runs (e.g., 1000): ";
    cin >> numTrials;

    cout << "\nRunning " << numTrials << " simulations with the given parameters...\n\n";
```

```
    double averageGenerations = runSimulations(params, numTrials);

    cout << "Average generations to speciation: " << averageGenerations << endl;

    return 0;
}
```