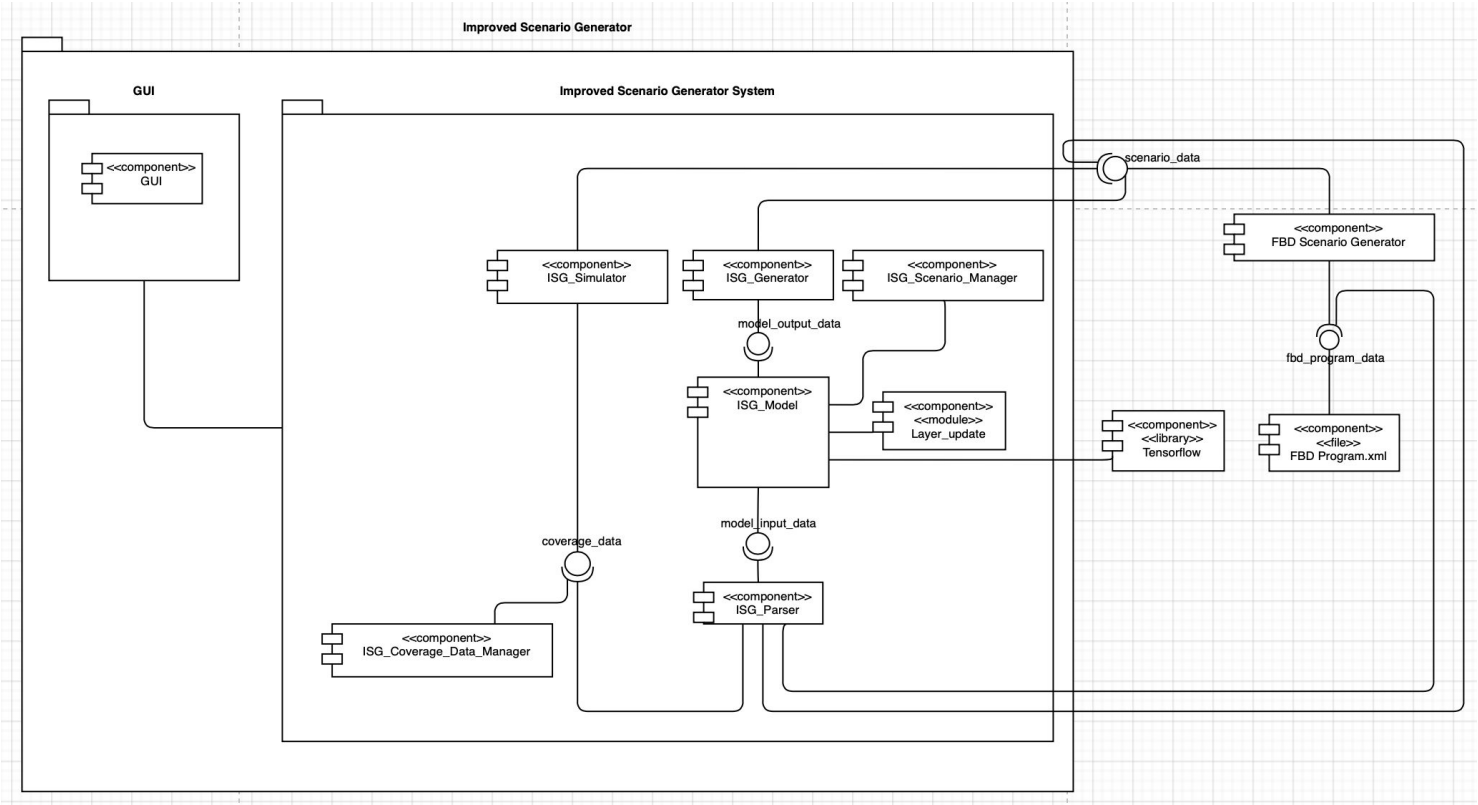
Improved
FBD Scenario Generator

1st Implementation : Prototype

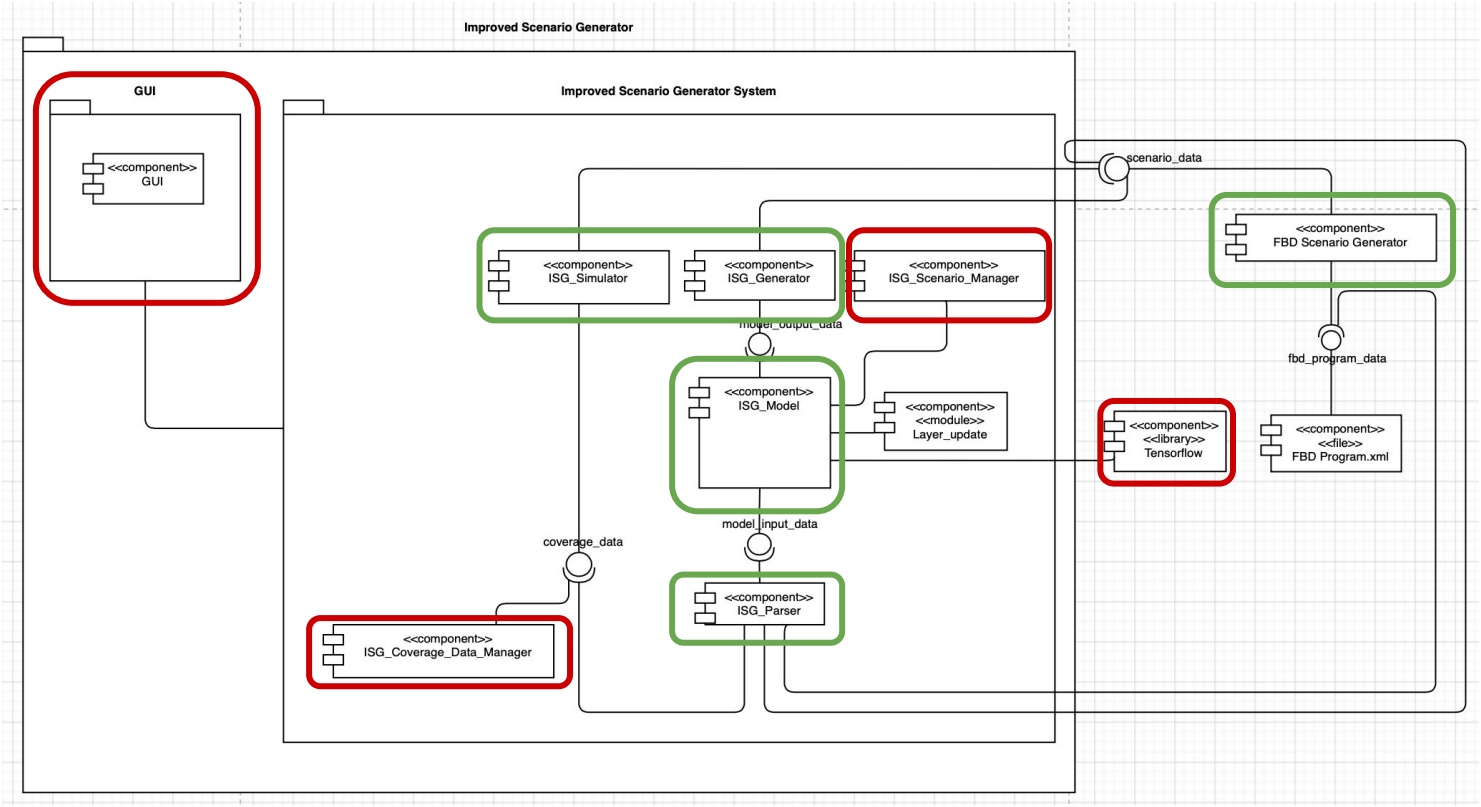
201411300 컴퓨터공학과 이정우
201312439 컴퓨터공학과 소경현
201312428 컴퓨터공학과 김정훈
201510624 컴퓨터공학과 김용현

Index

Architecture - SRA



Architecture - 1st implementation



ISG_Generater

- 시나리오 자동 생성기
- 선택된 FBD 프로그램을 parsing해서 데이터 분석하여 최초 시나리오를 자동 생성
- 기존 NuDE 코드를 분석 및 활용하여 구현
- 기존 Generator의 ADD 기능과 패턴 선택 기능은 만들고자 하는 프로그램에 필요없으므로 제거
- 로직과 UI를 분리하여 구현

ISG_Generater

ISG_Generater

```
if (k % 10 == 0)
    bw.write( str: "\n");
for (int h = 0; h < value.size(); h++) {

    if (rate.get(h) == 0) {
        value.set(h, (value.get(h)));
        value3.add(value.get(h));
    } else {
        int a = random.nextInt( bound: 10);
        int b = random.nextInt(rate.get(h));
        if (a < 5)
            b = b * -1;

        if (bool.get(h)) {
            if (a < 9) {
                value.set(h, 0);
                value3.add(0);
            } else {
                value.set(h, 1);
                value3.add(1);
            }
        } else {
            if (value.get(h) + b < 0)
                value.set(h, 0);
            else
                value.set(h, (value.get(h) + b));
            value3.add(value.get(h) + b);
        }
    }
}
```

- 기존 NuDE 프로그램에서 Default Setting 알고리즘을 차용
- ADD기능에서 사용되는 Scenario_num 변수나 패턴 선택에 사용되는 pattern_setting 부분 제거

ISG_Generater

Frame_ISG_Generater

```
for (int i = 0; i < inputVars.getLength(); i++) {
    String Variable_name = inputVars.item(i).getAttributes().getNamedItem("name").getNodeValue();

    for (int j = 0; j < inputVars.item(i).getChildNodes().getLength(); j++) {
        if (inputVars.item(i).getChildNodes().item(j).getNodeName().equals("type")) {
            for (int k = 0; k < inputVars.item(i).getChildNodes().item(j).getChildNodes().getLength(); k++) {
                if (inputVars.item(i).getChildNodes().item(j).getChildNodes().item(k).getNodeType() == 1) {
                    if (inputVars.item(i).getChildNodes().item(j).getChildNodes().item(k).getNodeName().equals("BOOL")){
                        isbool.put(Variable_name, true);
                        // 함수를 만들자 --> initial_value 와 rate를 초기화 해주는 함수를 넣자. isRandom을 인자로
                        initial_value = Integer.toString(random.nextInt( bound: 2));
                        rate = Integer.toString(random.nextInt( bound: 2));
                    }
                    else {
                        isbool.put(Variable_name, false);
                        // 함수를 만들자 --> initial_value 와 rate를 초기화 해주는 함수를 넣자. isRandom을 인자로
                        initial_value = Integer.toString(random.nextInt( bound: 5000));
                        rate = Integer.toString(random.nextInt( bound: 100));
                    }
                }
            }
        }
    }
```

- initial_value와 rate를 랜덤생성
- Data type을 제작하려는 프로그램에 맞게 변경

ISG_Generator

```
[ijeong-uui-MacBook-Pro:src twblne$ java Frame_ISG_Generator
 Frame_Scenario_Generator.Frame_Scenario_Generator()
 selected pou name = FIX_FALLING
```

AI_E	MDL_E	OB_INIT_STA	PV_OUT
1	1	0	2507
1	1	0	2497
1	1	0	2514
1	1	0	2511
1	1	0	2445
1	1	0	2490
1	1	0	2448
1	1	0	2375
1	1	0	2449
1	1	0	2415
end			
AI_E	MDL_E	OB_INIT_STA	PV_OUT
1	1	0	2507
1	1	0	2572
1	1	0	2592
1	1	0	2647
1	1	0	2663
1	1	0	2681
1	1	0	2645
1	1	0	2569
1	1	0	2541
1	1	0	2459

- Scenario를 제작하려는 프로그램에 맞게 생성
- BOOL과 INT를 구분하여 랜덤하게 생성

ISG_Generator

Problem

1. GUI 구현
   a. SRA에서의 Prototype에 맞춰 구현 필요
   b. Coverage 등 표현하고자 하는 부분들이 각 Class간 연동이 필요

2. Scenario 파일 형식 디테일 수정
   a. ISG_Model, ISG_Simulator에 맞춰 Scenario file의 형식 수정 필요

3. NuDE 프로그램에 대한 이해
   a. 필요한 상당부분을 이해했지만 simulation 부분과 동시에 이해할 필요가 있음

ISG_Simulation

- Coverage 계산 및 Scenario 연산
- ISG_Generation을 통해 생성된 시나리오를 통해 Coverage 및 Scenario 결과 연산
- 기존 FBD_Simulation을 분석 및 활용하여 구현
- 기존 Simulation의 GUI 기능은 개선을 위해 배제
- 로직을 통해 File text형식의 Output 추출

ISG_Simulation

```
private void cover() {
    //
    textField_model_input = "Fix_FALLING 경로\\FIX_FALLING.xml";
    textField_Simulation_Model_Open = "Fix_FALLING 경로\\FIX_FALLING.xml";
    textField_Simulation_Scenario_Open = "만들어진 Scenario 경로\\scenario.txt";

    selectedPOUName = "FIX_FALLING";

    File selectedFile = new File(textField_Simulation_Model_Open);
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    factory.setIgnoringElementContentWhitespace(true);
    try {
        DocumentBuilder builder = factory.newDocumentBuilder();
        doc = builder.parse(selectedFile);
        doc.getDocumentElement().normalize();
    } catch (ParserConfigurationException | SAXException | IOException e1) {
        e1.printStackTrace();
    }
      CreatPOUList(selectedFile);


    Map<String, DATA> Scenario = null;
    try {
        File scenario = new File(textField_Simulation_Scenario_Open);

        Comparision_FBD_Input fbd_Editor_for_Comparision = new Comparision_FBD_Input(scenario);
        Scenario = fbd_Editor_for_Comparision.getdata();

    } catch (IOException e1) {
        e1.printStackTrace();
    }

    File savedFile = new File("결과를 생성할 경로\\result.txt");
    FBDCover_Scenario_Result_Generation(Scenario, savedFile);

    Scenario.clear();
}
```

- Model과 Scenario의 경로를 통해 파일을 읽어온다.

- Scenario와 모델을 이용해 FBDCover_Scenario_Result_Generation에서 Simulation과 Coverage 계산을 진행

ISG_Simulation

```
// Coverage
set_elements_for_coverage(scenario);

for (String tag2 : scenario.keySet()) {
    int total_cycle = scenario.get(tag2).getList().get(scenario.get(tag2).getContext_list().get(0)).size();
    test_name = tag2;
    DATA data = new DATA(test_name);
    result.put(test_name, data);

    for (String tag : class_outputs.keySet()) {
        List<String> value = new ArrayList<String>();
        result.get(test_name).getList().put(tag, value);
        result.get(test_name).getContext_list().add(tag);
        class_outputs.get(tag).setCurrent_value(class_outputs.get(tag).getinitial_value());
    }
    for (String tag : class_feedbacks.keySet()) {
        class_feedbacks.get(tag).setCurrent_value(class_feedbacks.get(tag).getinitial_value());
    }

    for (int i = 0; i < total_cycle; i++) {
        cycle = i;

        Next_Calc();

        // output //
        for (String tag : class_outputs.keySet()) {
            result.get(test_name).getList().get(tag).add(class_outputs.get(tag).getCurrent_value());
            class_outputs_toggle.get(test_name).get(tag)
                    .setCurrent_value(class_outputs.get(tag).getCurrent_value());
        }
        // feedback //
        for (String tag : class_feedbacks.keySet()) {
            class_feedbacks.get(tag).setCurrent_value(class_outputs.get(tag).getCurrent_value());
        }
    }
```

- Scenario를 통해 Coverage를 위한 element 저장
- 처음 for문을 통해 Toggle Coverage 및 MCDC Coverage를 위한 작업 진행
- 다음 for문을 통해 FBD Scenario에 대한 연산을 진행

ISG_Simulation

```
for (String tag : class_blocks_mcdc.get(tag2).keySet()) {
    if (class_blocks_mcdc.get(tag2).get(tag).gettype().equals("BOOL")) {
        // 처음엔 Int값
        percentage = percentage + class_blocks_mcdc.get(tag2).get(tag).getCoveragePercentage();
        // MCDC 블록의 개수 return 받아서 total_percentage에 계속 저장
        total_percentage = total_percentage + class_blocks_mcdc.get(tag2).get(tag).getnumofMCDC();
    } else {
        // type이 INT인 부분은 MCDC에서 체크를 안하므로 numofNonMCDCBlock 수를 증가
        numofNonMCDCBlock++;
    }
}

System.err.println("-- BLOCK MC/DC COVERAGE --");
//커버된 MCDC 커버리지 블락락
int temp_percentage = 0;
if (percentage != 0) {
    System.err.println("[" + percentage + "]" + " / " + "[" + total_percentage + "]" + " = "
            + percentage * 100 / total_percentage + " % " + "(Non MC/DC block = " + numofNonMCDCBlock
            + ")");
    temp_percentage = percentage * 100 / total_percentage;
} else {
    System.err.println("[" + percentage + "]" + " / " + "[" + total_percentage + "]" + " = " + "0" + " % "
            + "(int = " + numofNonMCDCBlock + ")");
    temp_percentage = 0;
}

System.err.println("UnCoverd MC/DC - Block");
for (String tag : class_blocks_mcdc.get(tag2).keySet()) {
    String temp = "";
    if (class_blocks_mcdc.get(tag2).get(tag).gettype().equals("BOOL")) {
        temp = class_blocks_mcdc.get(tag2).get(tag).getUnCover();
    }
    if (!temp.equals("")) {
        System.err.println(temp);
    }
}
```

- MCDC Coverage 계산
- 커버가 가능한 block들의 개수를 percentage 전체 MCDC의 블록 개수를 Total_percentage에 저장
- 이후 percentage * 100 / total_percentage 를 통해 MCDC Coverage 계산

ISG_Simulation

Problem

1. OutPut 형식
   a. Model에서 사용하기 위한 Output 형식 지정
   b. Toggle Coverage 및 MCDC Coverage의 Result를 저장할 필요가 있음

2. Simulation 프로그램에 대한 이해
   a. 필요한 상당부분을 이해했지만 Generation부분과 Model Input을 위해 전체적인 코드를 해석할 필요가 있음

ISG_Model

Current

- 유전 알고리즘 사용

- mutation 생성에 2가지 전략 사용
  1. crossover된 시나리오 중 중앙 시나리오 PV_OUT 값에 rate보다 큰 값 더함
  2. crossover된 시나리오의 무작위로 선별된 PV_OUT 2개 값을 무작위로 변경

- generator의 결과로 생성된 시나리오를 읽어들여 새로운 시나리오를 생성

- 생성한 시나리오를 simulator를 위한 형식에 맞춰 새로운 시나리오 파일 생성

- Python 3.8로 유전 알고리즘을 구현

ISG_Model

```python
# make mutaion
def mutate(crossover, mutation_range = 100):
    # init centeroid
    centeroid = int(len(crossover[0])/2)

    # mutate centroid
    for idx in range(len(crossover)):
        crossover[idx].iloc[centeroid].loc['PV_OUT'] + mutation_range

    return crossover
```

[mutation 방법 1]

```python
# avoid overfitting Or underfitting
def mutation(pool_selected):
    mut_temp = pool_selected
    for i in range(len(pool_selected)):
        count = 1

        mut_temp[i].loc[5000+i, 'PV_OUT'] = np.random.randint(10000, 15000)
        mut_temp[i].loc[5000+i+1, 'PV_OUT'] = np.random.randint(10000, 15000)
        mut_temp[i] = mut_temp[i].drop(mut_temp[i].index[0:2])
        mut_temp[i] = mut_temp[i].fillna(0)

    return mut_temp
```

[mutation 방법 2]

ISG_Model

Problem

1. 시나리오 생성에 활용할 딥러닝 알고리즘 부재
   a. 기존 계획(딥러닝을 활용)과 부합하지 않음
   b. 다른 부분(simulator, generator)와 호환되지 않아 성능 저하 가능성 존재

2. 유전 알고리즘을 Python으로 구현
   a. simulator, generator와 호환성 문제

3. 유전 알고리즘 성능 개선 방안
   a. crossover
   b. mutation

Future plan - ISG_Generater

1. GUI에 대한 이해 및 로직 수정 → Problem 1.a
2. ISG_Simulation 및 ISG_Model과 연동 후, ISG_Coverage_Data_Manager를 구현하여 GUI 연동, 만약 힘들 경우 GUI를 두 단계로 나누고 SRA 일부분 수정 → Problem 1.b, 2.a
3. Simulation과의 연계를 위해서 NuDE 프로그램을 심층 분석 → Problem 3.a

Future plan - ISG_Simulation

1. Output 형태에 대한 고민 후 형식 바꾸기 → Problem 1.a
2. Toggle과 MCDC의 Coverage에 대한 OutPut 연구 및 형태 변형→ Problem 1.b
3. Simulation과 Model의 연동을 위한 코드 연구→ Problem 2.a

Future plan - ISG_Model

1. 유전 알고리즘의 mutation 생성에 linear regression 활용 → Problem 1.a
2. 적절한 강화학습 알고리즘 활용 → Problem 1.a
3. 적절한 neural net 알고리즘을 찾지 못할 경우 유전 알고리즘을 Java로 구현 → Problem 1.b
4. Java로 유전 알고리즘 구현 → Problem 2.a
5. 다른 부분(simulator, generator)과 연결 후 실험적으로 유전 알고리즘 성능 개선 → Problem 3.a, 3.b