CS3213 Project – Week 5

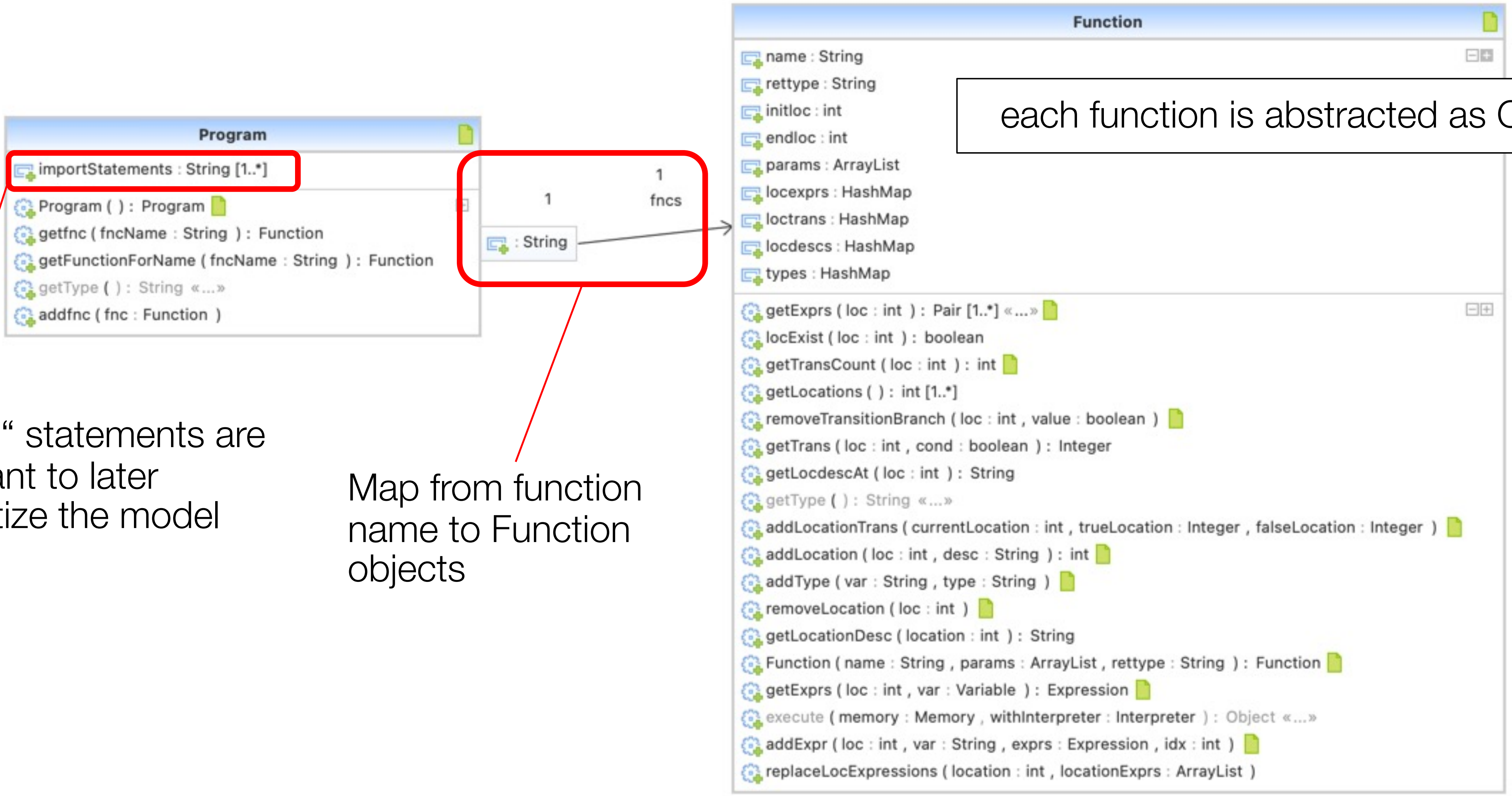
Module Design & Project Planning | 09-02-2022

- ❏ Plagiarism & Attribution
- ❏ its-core: Program model
- ❏ Short Intro to Project Planning
- ❏ Assignment 5

Plagiarism: How to attribute work?

1. Use *code comments* to highlight code which is not your contribution.
2. *Summarize* all attributions in one file in the parent folder of your repository: **`ATTRIBUTIONS.md`**
    - ❏ You need to specify **where** in the code we can find the comment for this attribution (see item 1)
    - ❏ You need to specify the **reference**: where does the code come from?
    - ❏ You need to specify **why** you need to include this code

its-core: Program model (1/3)



```
Program
importStatements : String [1..*]
Program ( ) : Program
getfnc ( fncName : String ) : Function
getFunctionForName ( fncName : String ) : Function
getType ( ) : String «...»
addfnc ( fnc : Function )
```

```
: String
1
1
fncs
```

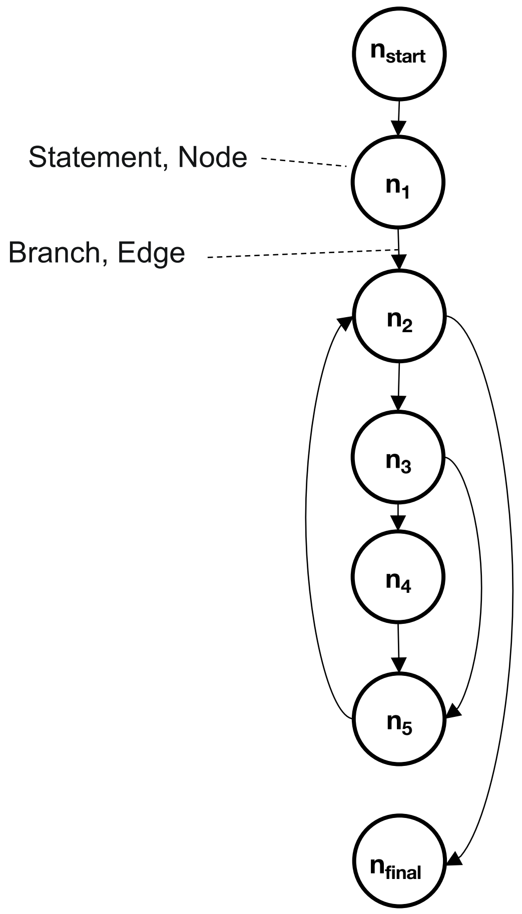
```
Function
name : String
rettype : String
initloc : int
endloc : int
params : ArrayList
locexprs : HashMap
loctrans : HashMap
locdescs : HashMap
types : HashMap
getExprs ( loc : int ) : Pair [1..*] «...»
locExist ( loc : int ) : boolean
getTransCount ( loc : int ) : int
getLocations ( ) : int [1..*]
removeTransitionBranch ( loc : int , value : boolean )
getTrans ( loc : int , cond : boolean ) : Integer
getLocdescAt ( loc : int ) : String
getType ( ) : String «...»
addLocationTrans ( currentLocation : int , trueLocation : Integer , falseLocation : Integer )
addLocation ( loc : int , desc : String ) : int
addType ( var : String , type : String )
removeLocation ( loc : int )
getLocationDesc ( location : int ) : String
Function ( name : String , params : ArrayList , rettype : String ) : Function
getExprs ( loc : int , var : Variable ) : Expression
execute ( memory : Memory , withInterpreter : Interpreter ) : Object «...»
addExpr ( loc : int , var : String , exprs : Expression , idx : int )
replaceLocExpressions ( location : int , locationExprs : ArrayList )
```

each function is abstracted as CFG

„Import“ statements are important to later concretize the model

Map from function name to Function objects

Control Flow Graph (CFG) Example



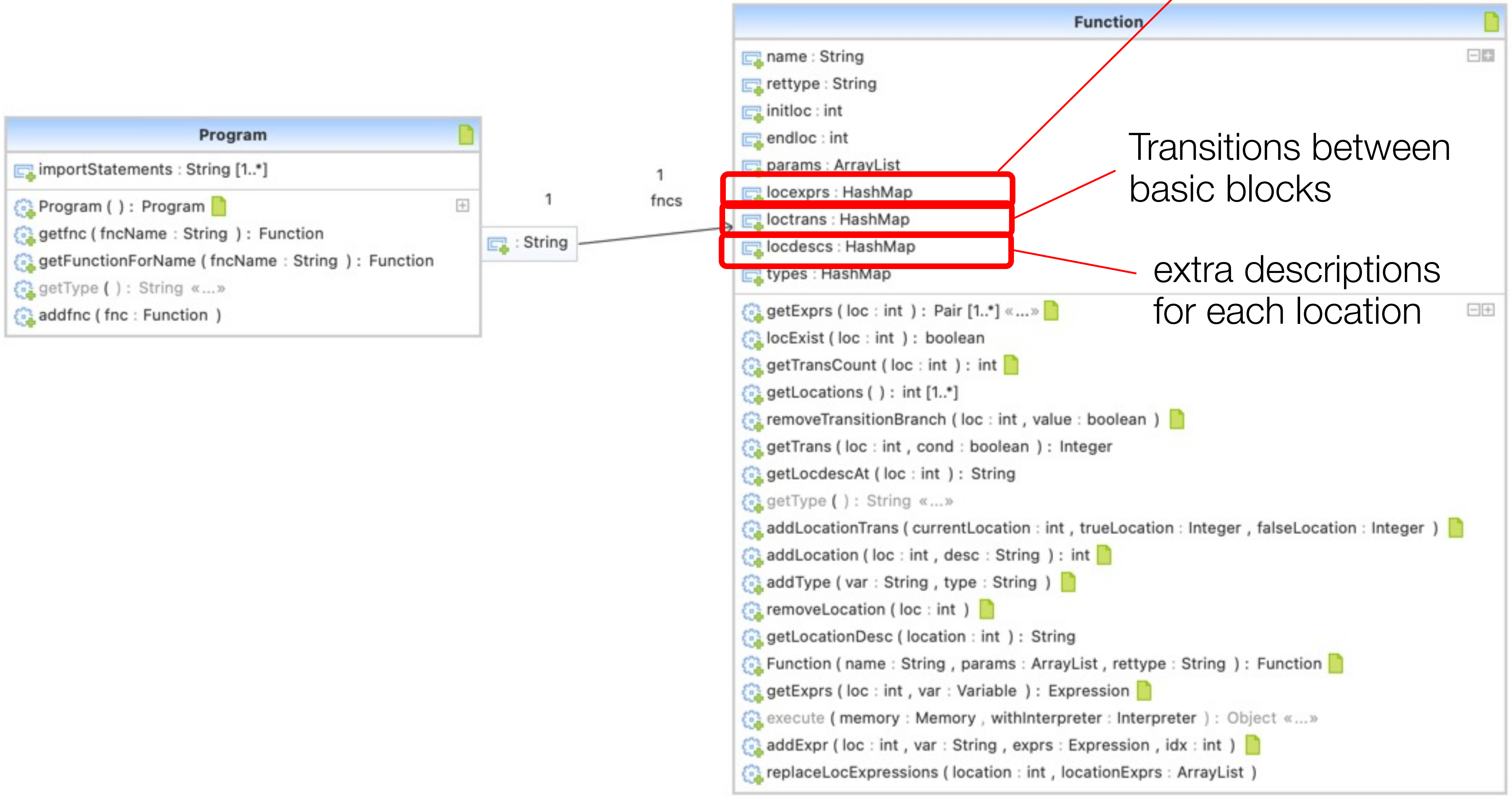
```
void CountChar (int &ACount, int &TotalCount)
{

  char c;
  cin >> c;


  while ((c >= 'A') && (c <= 'Z') && (TotalCount < INT_MAX))
  {


    TotalCount = TotalCount + 1;
    if ((c == 'A'))
    {

      ACount = ACount + 1;

    }

    cin >> c;
  }


}
```

its-core: Program model (2/3)

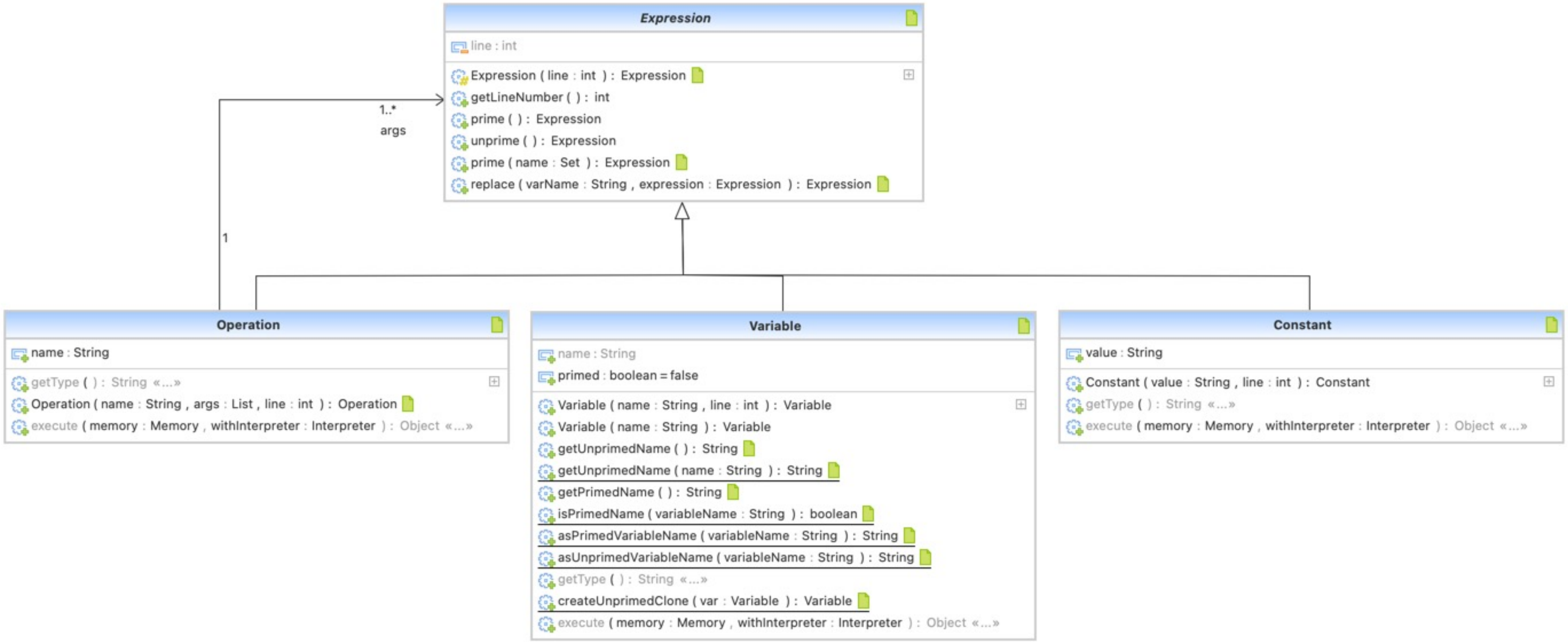
List of statements at specific location

Transitions between basic blocks

extra descriptions for each location


Program
importStatements : String [1..*]
Program ( ) : Program
getfnc (fncName : String) : Function
getFunctionForName (fncName : String) : Function
getType ( ) : String «...»
addfnc (fnc : Function)
1
: String
1
fncs
Function
name : String
rettype : String
initloc : int
endloc : int
params : ArrayList
locexprs : HashMap
loctrans : HashMap
locdescs : HashMap
types : HashMap
getExprs (loc : int) : Pair [1..*] «...»
locExist (loc : int) : boolean
getTransCount (loc : int) : int
getLocations ( ) : int [1..*]
removeTransitionBranch (loc : int , value : boolean)
getTrans (loc : int , cond : boolean) : Integer
getLocdescAt (loc : int) : String
getType ( ) : String «...»
addLocationTrans (currentLocation : int , trueLocation : Integer , falseLocation : Integer)
addLocation (loc : int , desc : String) : int
addType (var : String , type : String)
removeLocation (loc : int)
getLocationDesc (location : int) : String
Function (name : String , params : ArrayList , rettype : String) : Function
getExprs (loc : int , var : Variable) : Expression
execute (memory : Memory , withInterpreter : Interpreter) : Object «...»
addExpr (loc : int , var : String , exprs : Expression , idx : int)
replaceLocExpressions (location : int , locationExprs : ArrayList)

its-core: Program model (3/3)


Expression
line : int
Expression (line : int) : Expression
getLineNumber ( ) : int
prime ( ) : Expression
unprime ( ) : Expression
prime (name : Set) : Expression
replace (varName : String , expression : Expression) : Expression
1..*
args
1
Operation
name : String
getType ( ) : String «...»
Operation (name : String , args : List , line : int) : Operation
execute (memory : Memory , withInterpreter : Interpreter) : Object «...»
Variable
name : String
primed : boolean = false
Variable (name : String , line : int) : Variable
Variable (name : String) : Variable
getUnprimedName ( ) : String
getUnprimedName (name : String) : String
getPrimedName ( ) : String
isPrimedName (variableName : String) : boolean
asPrimedVariableName (variableName : String) : String
asUnprimedVariableName (variableName : String) : String
getType ( ) : String «...»
createUnprimedClone (var : Variable) : Variable
execute (memory : Memory , withInterpreter : Interpreter) : Object «...»
Constant
value : String
Constant (value : String , line : int) : Constant
getType ( ) : String «...»
execute (memory : Memory , withInterpreter : Interpreter) : Object «...»

its-core: Program model (Example 1/2)

```
#include <stdio.h>
int main() {
  int a=0,b=0;
  b=1+a;
  return 0;
}
```

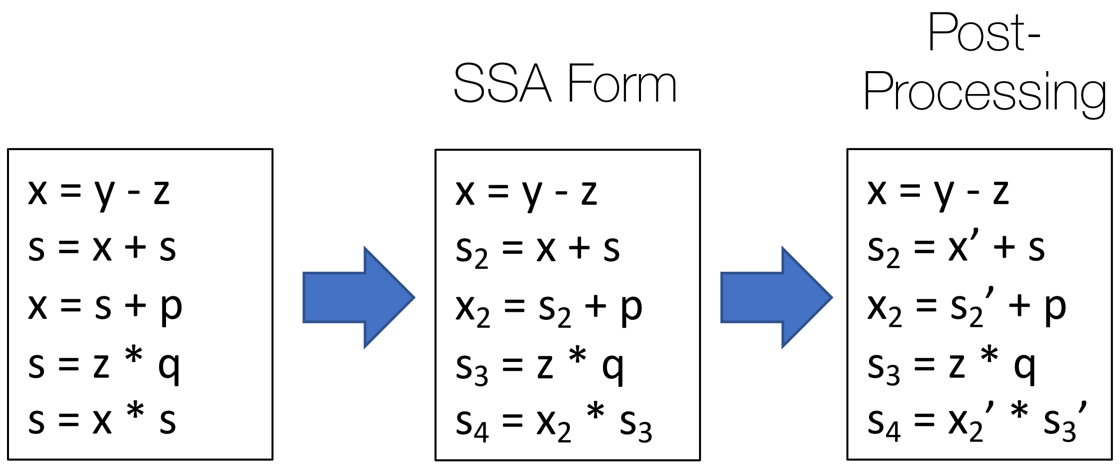
```
fun main () : int
----------------------------------------
initloc : 1
Loc 1 (at the beginning of the function 'main')
----------------------------------------
  a := 0
  b := +(1, a')
  $ret := 0
----------------------------------------
  True -> null   False -> null
```

Static Single Assignment (SSA)

- ❑ requires that each variable be assigned exactly once
- ❑ makes use-def chains explicit
  - ❑ helps to simplify optimizations
  - ❑ helps to formulate local repair (comparison with reference solution)
- ❑ enforced on a basic block level

unprimed: before assignment
primed: after assignment

For example:



SSA Form

Post-Processing

$x = y - z$
$s = x + s$
$x = s + p$
$s = z * q$
$s = x * s$

$x = y - z$
$s_2 = x + s$
$x_2 = s_2 + p$
$s_3 = z * q$
$s_4 = x_2 * s_3$

$x = y - z$
$s_2 = x' + s$
$x_2 = s_2' + p$
$s_3 = z * q$
$s_4 = x_2' * s_3'$

If-Then-Else (ITE)

- ❑ simplifies model by merging branches if possible
- ❑ `sg.edu.nus.se.its.util.Constants.CONDITIONAL_OPERATOR`

For example:

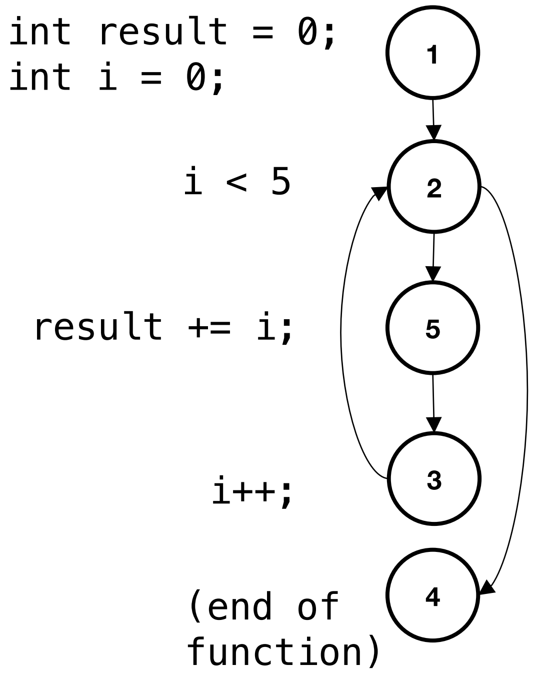
```
#include <stdio.h>
int main() {
  int a=0,b=0,c=0;
  b=1+a;
  if (b > 1) {
    c = 3;
  } else {
    c = 5;
  }
  return 0;
}
```

```
fun main () : int
----------------------------------------
initloc : 1
Loc 1 (at the beginning of the function
'main')
----------------------------------------
  a := 0
  b := +(1, a')
  c := ite(>(b', 1), 3, 5)
  $ret := 0
----------------------------------------
  True -> null   False -> null
```

its-core: Program model

(Example 2/2)

```
int main() {
    int result = 0;
    for (int i = 0; i < 5; i++) {
        result += i;
    }
}
```

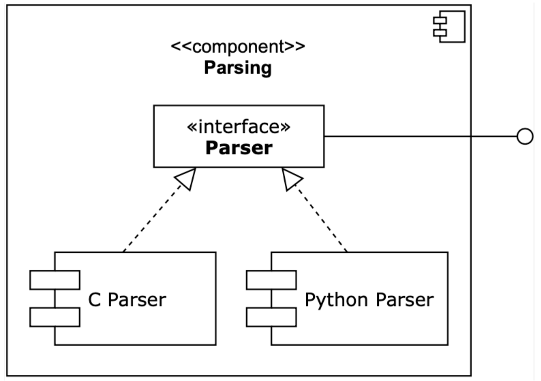


```
fun main () : int
----------------------------------------
initloc : 1
Loc 1 (at the beginning of the function 'main')
----------------------------------------
  result := 0
  i := 0
----------------------------------------
  True -> 2   False -> null
Loc 2 (the condition of the 'for' loop at line 3)
----------------------------------------
  $cond := <(i, 5)
----------------------------------------
  True -> 5   False -> 4
Loc 3 (update of the 'for' loop at line 3)
----------------------------------------
  i := +(i, 1)
----------------------------------------
  True -> 2   False -> null
Loc 4 (*after* the 'for' loop starting at line 3)
----------------------------------------
----------------------------------------
  True -> null   False -> null
Loc 5 (inside the body of the 'for' loop beginning at line 3)
----------------------------------------
  result := +(result, i)
----------------------------------------
  True -> 3   False -> null
```

its-core: Program model (Current Limitations)

- ❏ current assumption: program is compilable
- ❏ not supported yet: pointer and multi-dimensional arrays

Parser API

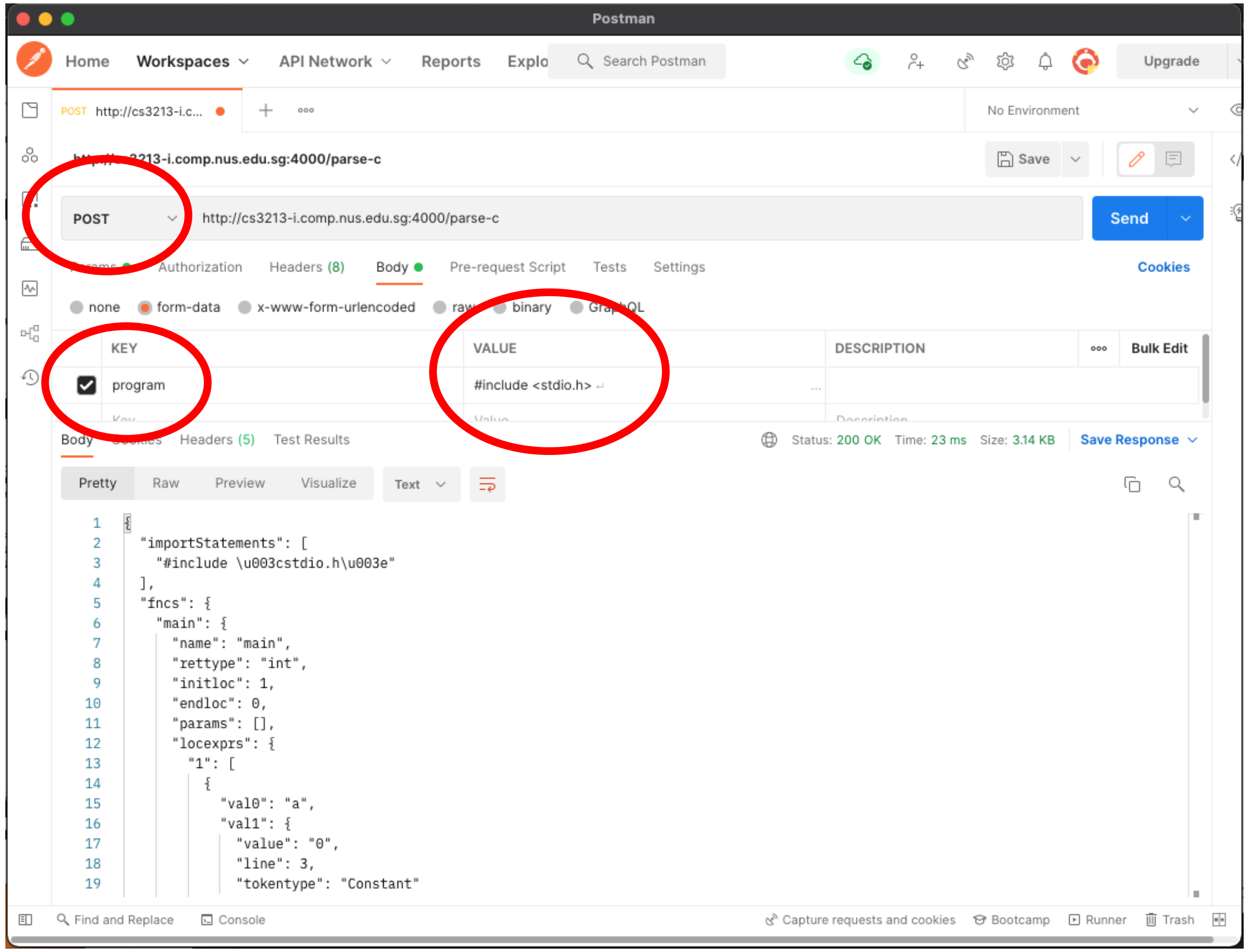


- ❑ Input: program in `.c` or `.py` source file
- ❑ Output: internal program object in `json` format
- ❑ Purpose: prepare test inputs for your test cases / evaluation

Deployed as POST service, accessible within the SoC VPN:

- ❑ http://cs3213-i.comp.nus.edu.sg:4000/parse-c
- ❑ http://cs3213-i.comp.nus.edu.sg:4000/parse-python

How to use: Parser API (1/2)



POST http://cs3213-i.c...

http://cs3213-i.comp.nus.edu.sg:4000/parse-c

POST http://cs3213-i.comp.nus.edu.sg:4000/parse-c

KEY	VALUE	DESCRIPTION
✓ program	#include <stdio.h> ↵	

Status: 200 OK Time: 23 ms Size: 3.14 KB

```
{
  "importStatements": [
    "#include \u003cstdio.h\u003e"
  ],
  "fncs": {
    "main": {
      "name": "main",
      "rettype": "int",
      "initloc": 1,
      "endloc": 0,
      "params": [],
      "locexprs": {
        "1": [
          {
            "val0": "a",
            "val1": {
              "value": "0",
              "line": 3,
              "tokentype": "Constant"
```

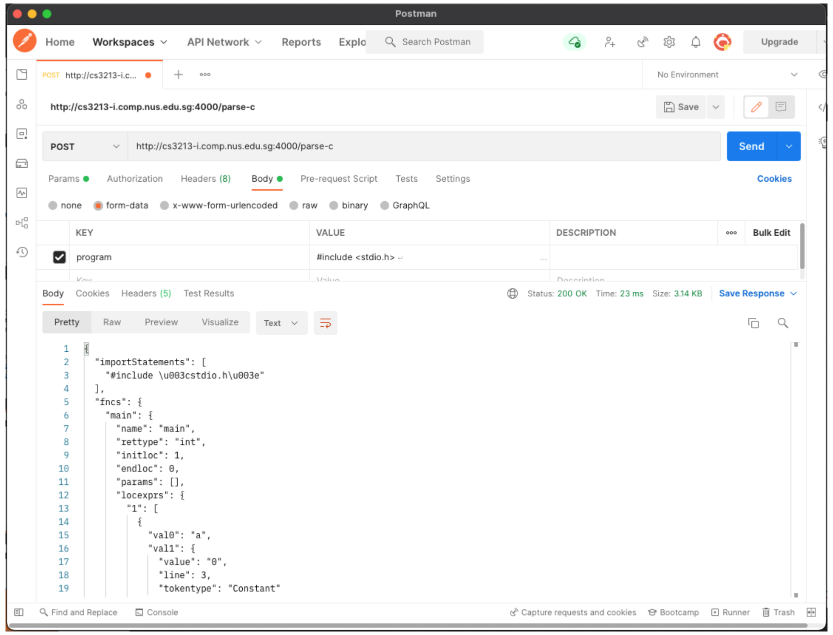
- ❑ For example, you can use the tool **Postman**[1] to send `POST` requests to our server
- ❑ `POST` body should have the *key* „`program`“ and as *value* the source code

[1] https://www.postman.com (you can use the free version)

How to use: Parser API (2/2)

```c
#include <stdio.h>
int main() {
  int a=0,b=0;
  b=1+a;
  return 0;
}
```

http://cs3213-i.comp.nus.edu.sg:4000/parse-c



POST http://cs3213-i.c...

http://cs3213-i.comp.nus.edu.sg:4000/parse-c

POST http://cs3213-i.comp.nus.edu.sg:4000/parse-c

Body: none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION
✓	program	#include <stdio.h> ↵	

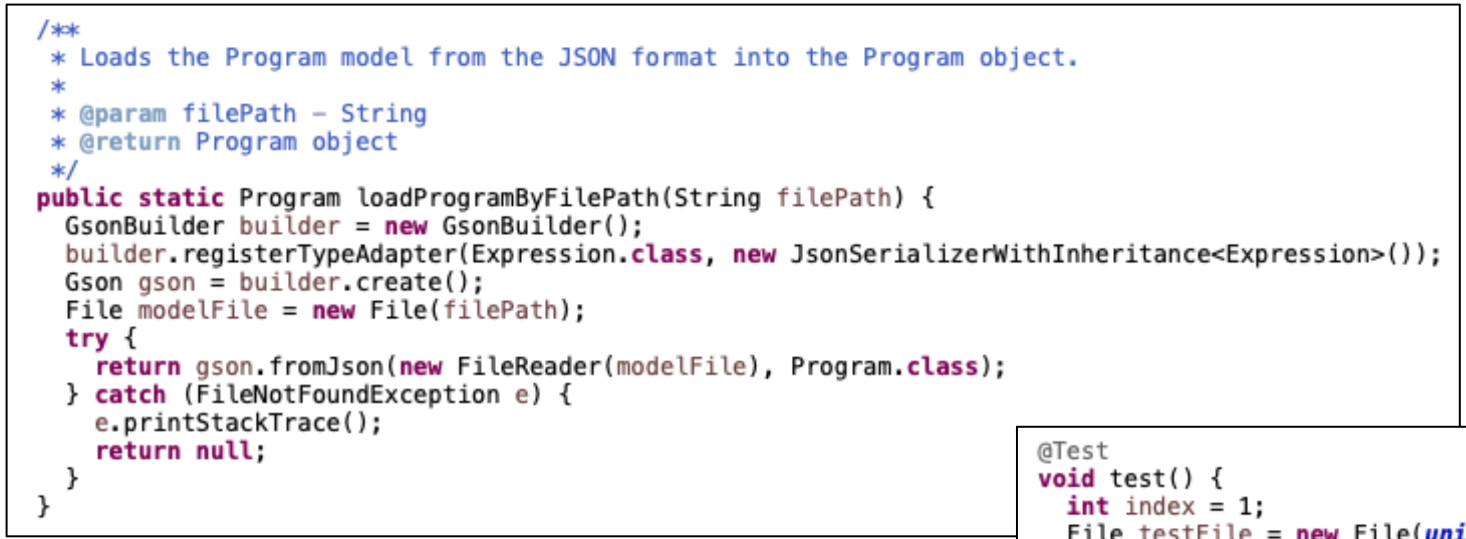
Status: 200 OK Time: 23 ms Size: 3.14 KB

```json
{
  "importStatements": [
    "#include \u003cstdio.h\u003e"
  ],
  "fncs": {
    "main": {
      "name": "main",
      "rettype": "int",
      "initloc": 1,
      "endloc": 0,
      "params": [],
      "locexprs": {
        "1": [
          {
            "val0": "a",
            "val1": {
              "value": "0",
              "line": 3,
              "tokentype": "Constant"
```

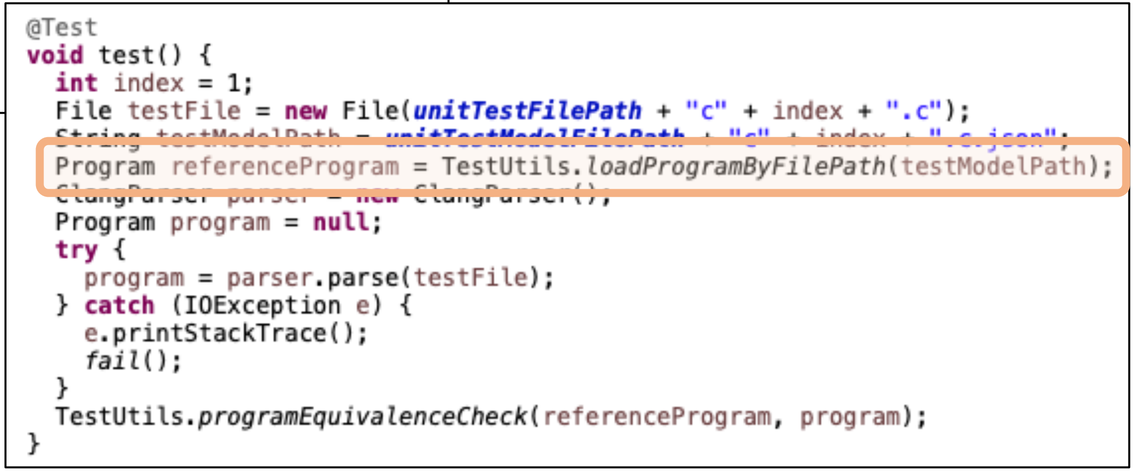
```json
{
  "importStatements": [
    "#include \u003cstdio.h\u003e"
  ],
  "fncs": {
    "main": {
      "name": "main",
      "rettype": "int",
      "initloc": 1,
      "endloc": 0,
      "params": [],
      "locexprs": {
        "1": [
          {
            "val0": "a",
            "val1": {
              "value": "0",
              "line": 3,
              "tokentype": "Constant"
            },
            "valueArray": [
              "a",
              {
                "value": "0",
                "line": 3
              }
            ],
            "valueList": [
              "a",
              {
                "value": "0",
                "line": 3
              }
            ]
          },
          ...
```

How to import program as `.json`

→ `sg.edu.nus.se.its.util.TestUtils`



```
/**
 * Loads the Program model from the JSON format into the Program object.
 *
 * @param filePath - String
 * @return Program object
 */
public static Program loadProgramByFilePath(String filePath) {
  GsonBuilder builder = new GsonBuilder();
  builder.registerTypeAdapter(Expression.class, new JsonSerializerWithInheritance<Expression>());
  Gson gson = builder.create();
  File modelFile = new File(filePath);
  try {
    return gson.fromJson(new FileReader(modelFile), Program.class);
  } catch (FileNotFoundException e) {
    e.printStackTrace();
    return null;
  }
}
```



```
@Test
void test() {
  int index = 1;
  File testFile = new File(unitTestFilePath + "c" + index + ".c");
  String testModelPath = unitTestModelFilePath + "c" + index + ".c.json";
  Program referenceProgram = TestUtils.loadProgramByFilePath(testModelPath);
  ClangParser parser = new ClangParser();
  Program program = null;
  try {
    program = parser.parse(testFile);
  } catch (IOException e) {
    e.printStackTrace();
    fail();
  }
  TestUtils.programEquivalenceCheck(referenceProgram, program);
}
```

→ `sg.edu.nus.se.its.parser.BasicTest`

Any remaining question about the `Program` model or the API?

Project Management Tasks

- ❑ Product Quotation
- ❑ Project and Time Planning
- ❑ Project Cost Calculation
- ❑ Project Supervision and Review
- ❑ Selection/Hiring, Assessment, and Leading of Team Members
- ❑ Presentation and Creation of Reports
- ❑ Securing good surrounding conditions

Project Planning - Aspects

Task Planning	Work Packages
Time Planning	Milestones, Releases
Resource Planning	Effort, Staff, Budget

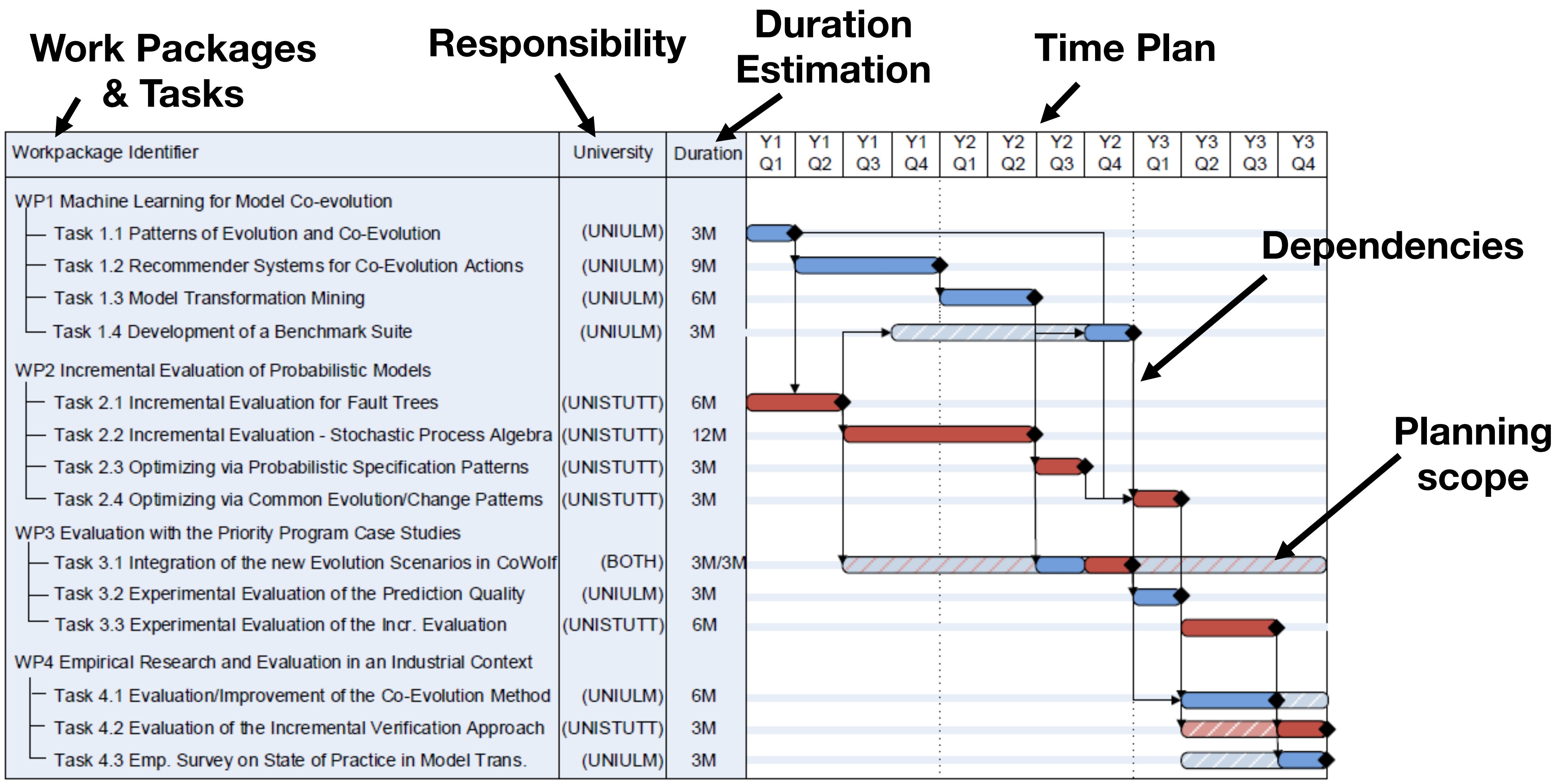
Work Packages

- **Work Package** = result & partial results
  + cost estimation
  + (after completion) real cost
- a **task** is suitable as work package if:
  - it can be done without further coordination constraint / dependency,
  - the progress and the end can be determined in an objective fashion,
  - there are events that impact the start and the end, and
  - the cost and the deadlines can be estimated.

Sample Layout

Work Package ID: a100.5	**Project:** C Parser **Phase:** Implementation	
Task:	Description Results Steps Critical Resources	
Cost:	**Plan** 3 PD (=24 hours)	**Real**
Dates: Stub xyz Module cyz …	 10/02/2022 17/02/2022 …	
Created by: YN **Authorized by:** ZF	04/12/2021 06/12/2021	

Gantt-Charts (Example)

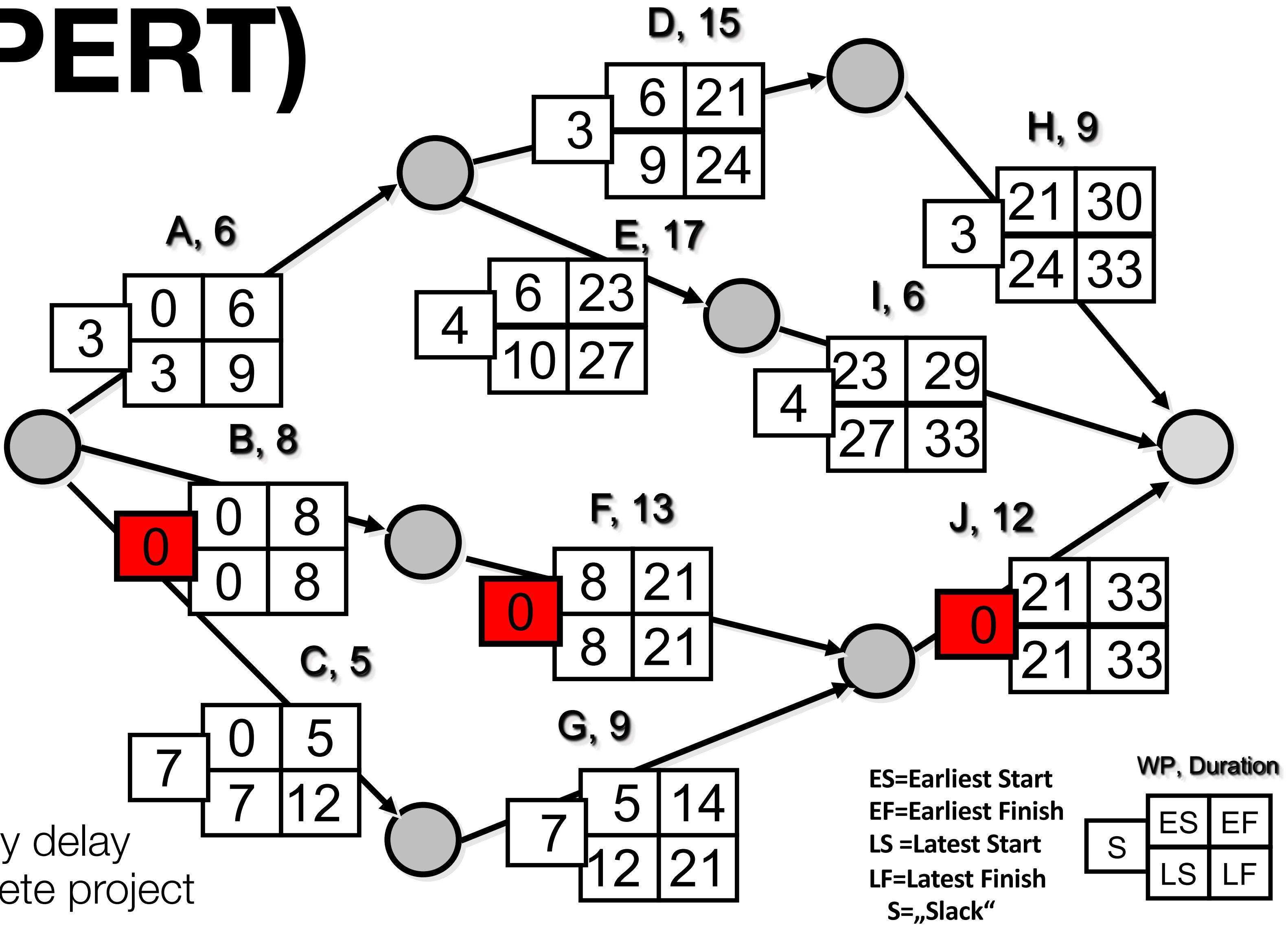


Workpackage Identifier	University	Duration
WP1 Machine Learning for Model Co-evolution		
Task 1.1 Patterns of Evolution and Co-Evolution	(UNIULM)	3M
Task 1.2 Recommender Systems for Co-Evolution Actions	(UNIULM)	9M
Task 1.3 Model Transformation Mining	(UNIULM)	6M
Task 1.4 Development of a Benchmark Suite	(UNIULM)	3M
WP2 Incremental Evaluation of Probabilistic Models		
Task 2.1 Incremental Evaluation for Fault Trees	(UNISTUTT)	6M
Task 2.2 Incremental Evaluation - Stochastic Process Algebra	(UNISTUTT)	12M
Task 2.3 Optimizing via Probabilistic Specification Patterns	(UNISTUTT)	3M
Task 2.4 Optimizing via Common Evolution/Change Patterns	(UNISTUTT)	3M
WP3 Evaluation with the Priority Program Case Studies		
Task 3.1 Integration of the new Evolution Scenarios in CoWolf	(BOTH)	3M/3M
Task 3.2 Experimental Evaluation of the Prediction Quality	(UNIULM)	3M
Task 3.3 Experimental Evaluation of the Incr. Evaluation	(UNISTUTT)	6M
WP4 Empirical Research and Evaluation in an Industrial Context		
Task 4.1 Evaluation/Improvement of the Co-Evolution Method	(UNIULM)	6M
Task 4.2 Evaluation of the Incremental Verification Approach	(UNISTUTT)	3M
Task 4.3 Emp. Survey on State of Practice in Model Trans.	(UNIULM)	3M

(Example taken from a Research Project)

Program Evaluation and Review Technique (PERT)

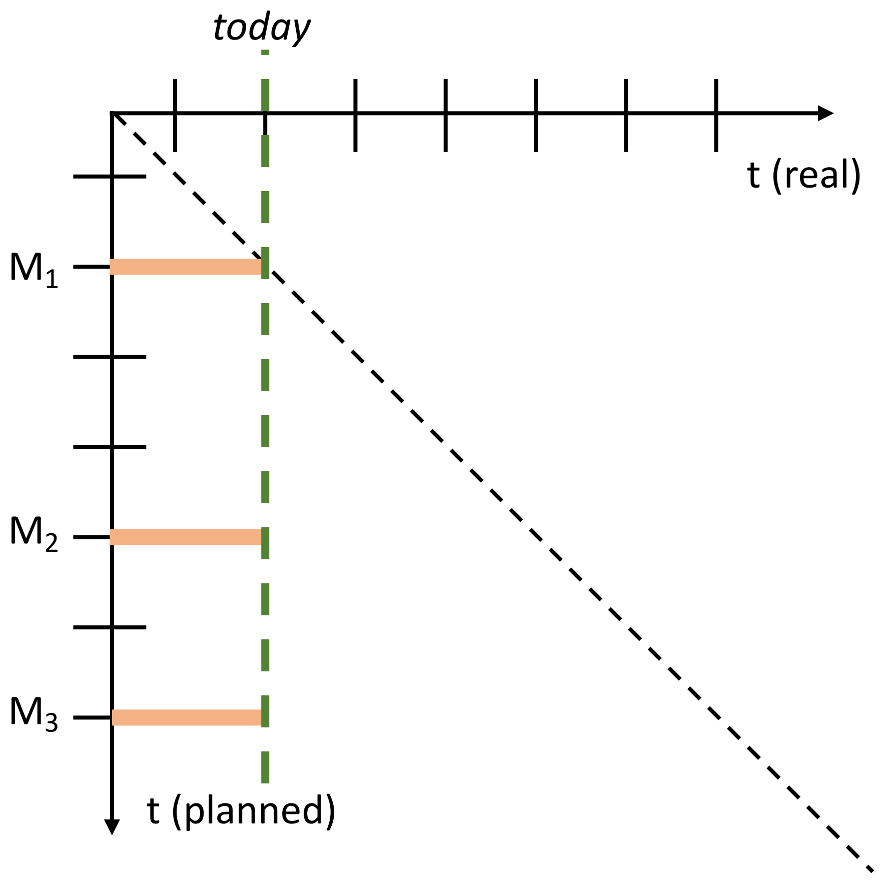
Work Package (WP)	Duration (e.g., days)	Depends on
A	6	–
B	8	–
C	5	–
D	15	A
E	17	A
F	13	B
G	9	C
H	9	D
I	6	E
J	12	F, G

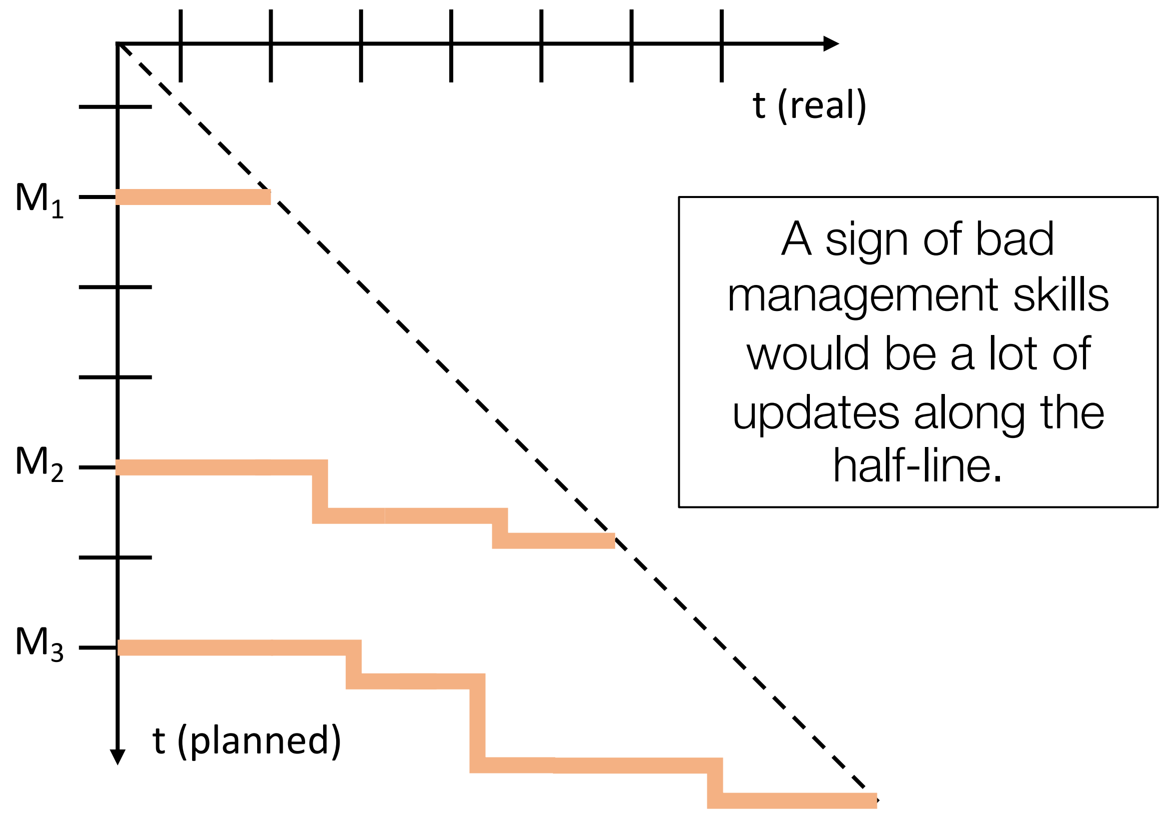


→ Identify the ***critical path***, i.e., any delay along this path will delay the complete project

Planning & Retrospective

→ **Milestone Trend Analysis (MTA)**, continuous task in project planning





A sign of bad management skills would be a lot of updates along the half-line.

Checklist Project Planning

- ❑ Select process model
- ❑ Derive project plan
- ❑ Determine and fix milestones
- ❑ Estimate Cost (i.e., time effort)
- ❑ Resource Planning
- ❑ Duration = Time Effort / Ressources
- ❑ Planning Review (e.g., PERT)
- ❑ Check Optimizations
- ❑ Reduce Risks
- ❑ Create Gantt-Chart
- ❑ Ressource Allocation
- ❑ …

Conclusion

- ❑ Use Parser API to prepare test inputs.
- ❑ Next step: exploring the **solution space → start implementation**

Next Lecture (Project-Part) – Week 6:
Implementation & Intermediate Deliverable (A6)

- ➢ Discussion Implementation (Clean Code) & Testing
- ➢ Assignment 6: Intermediate Deliverable (Content + Grading)