CS3213 Project – Week 5

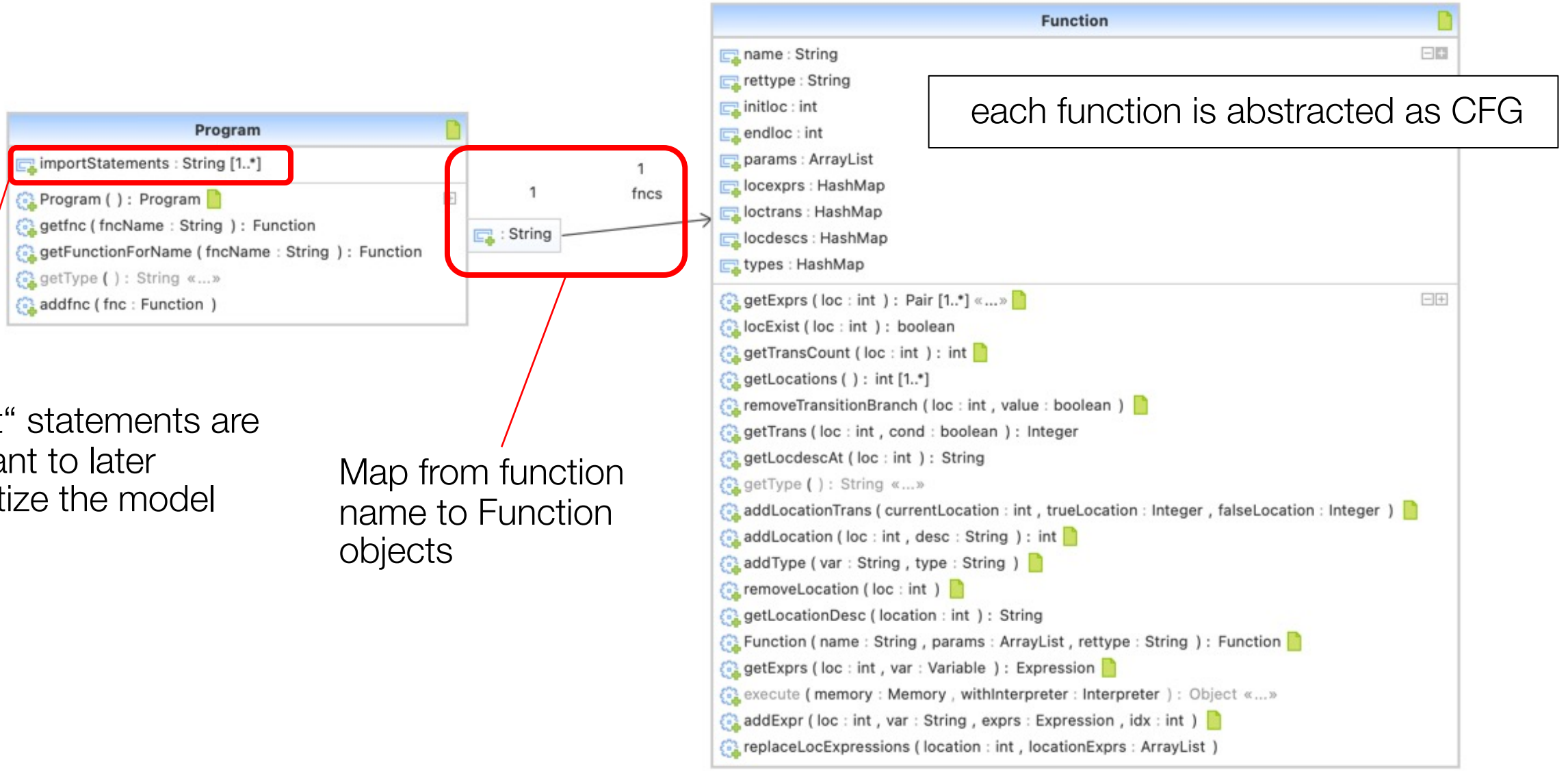
Module Design & Project Planning | 09-02-2022

- ❑ Plagiarism & Attribution
- ❑ its-core: Program model
- ❑ Short Intro to Project Planning

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)

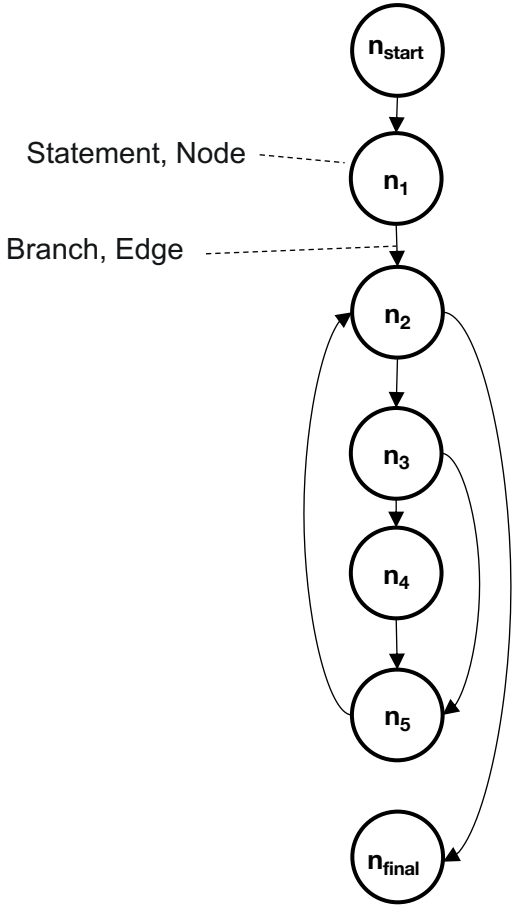


each function is abstracted as CFG

„Import“ statements are important to later concretize the model

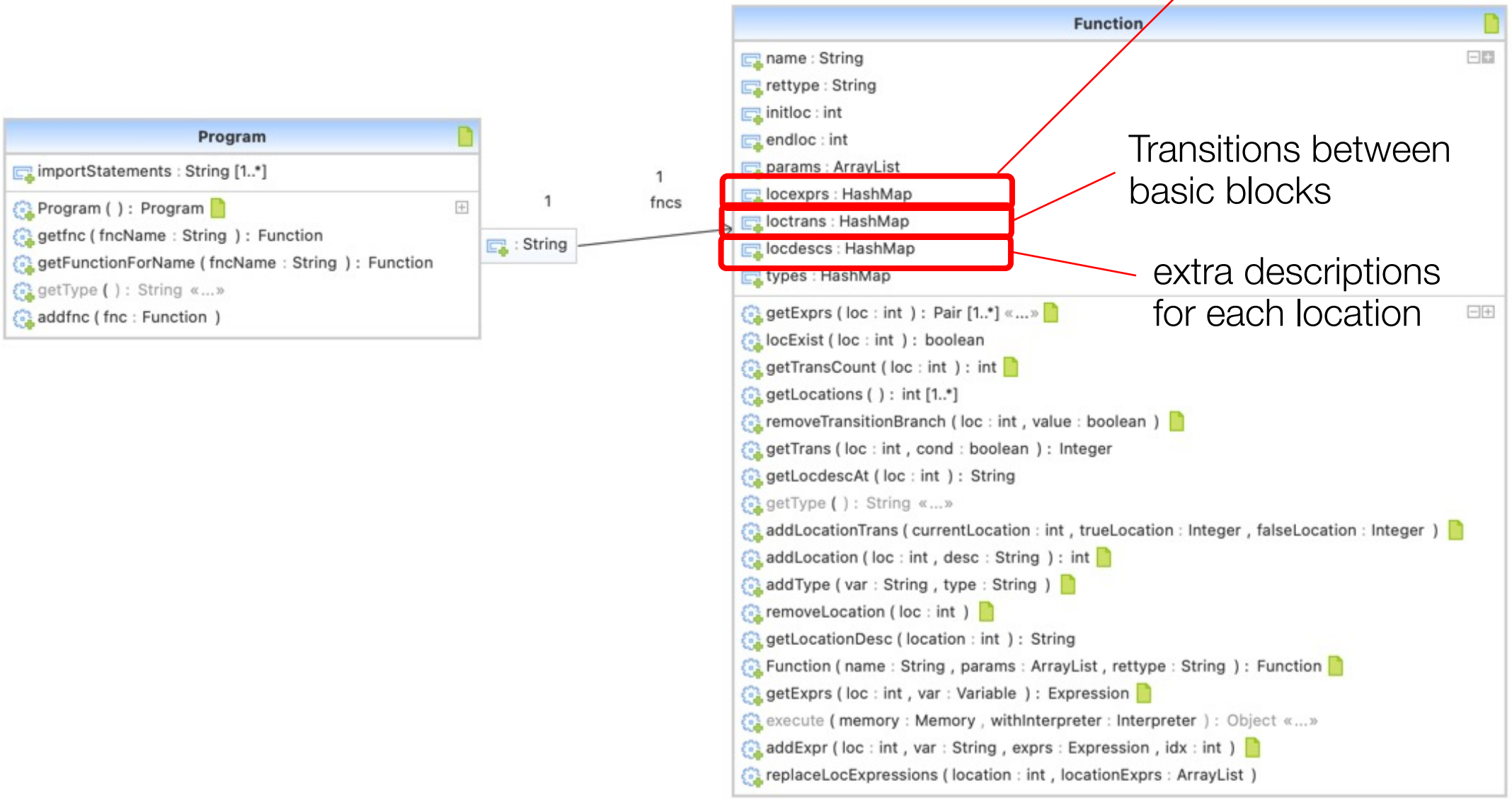
Map from function name to Function objects

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)



its-core: Program model (3/3)

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

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

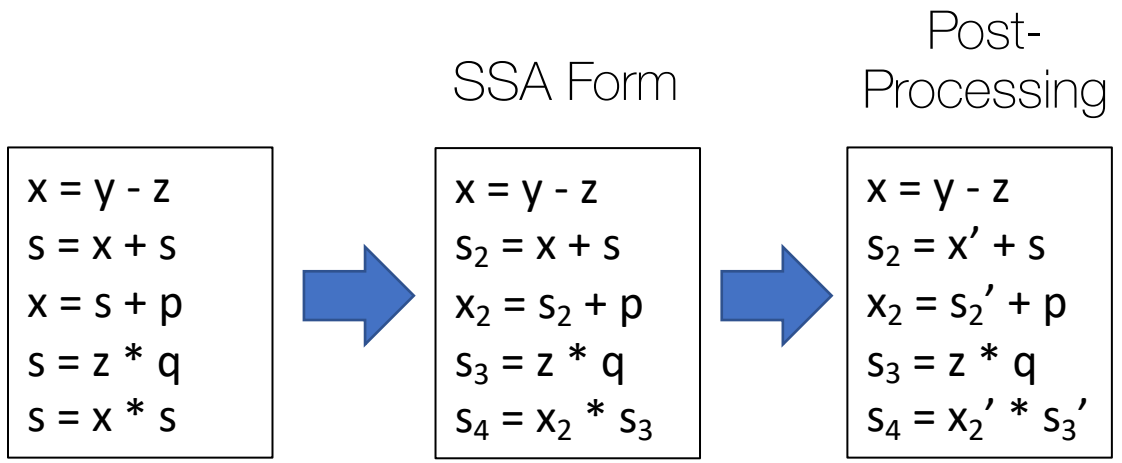
```
fun main () : int
----------------------------------------
initloc : 1
Loc 1 (at the beginning of the function 'main')
----------------------------------------
  a := 0
  b := +(1, a')
  $ret := 0
----------------------------------------
  True -> null   False -> null
```

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:



If-Then-Else (ITE)

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

For example:

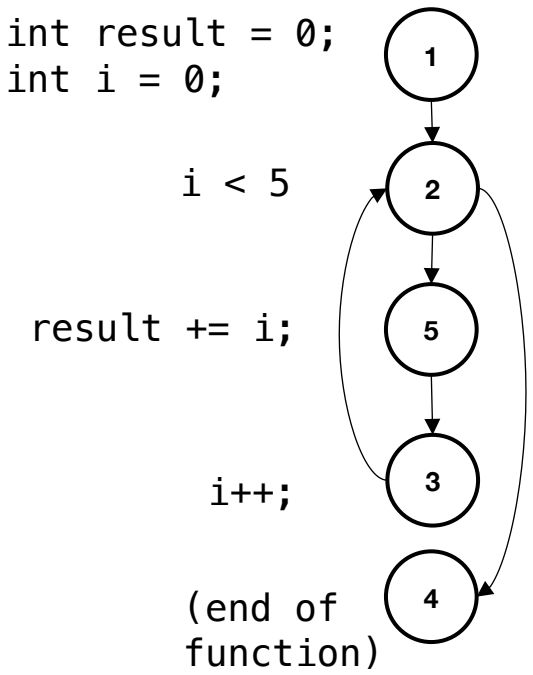
```
#include <stdio.h>
int main() {
  int a=0,b=0,c=0;
  b=1+a;
  if (b > 1) {
    c = 3;
  } else {
    c = 5;
  }
  return 0;
}
```

```
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;
    }
}
```

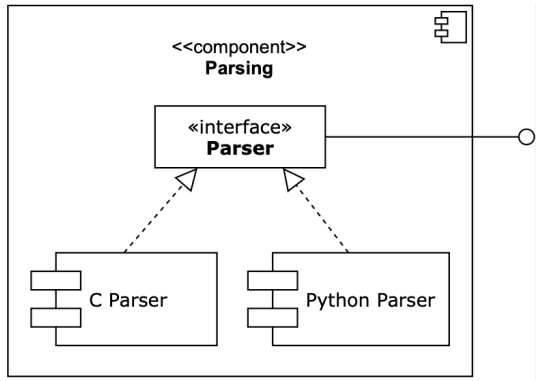



```
fun main () : int
----------------------------------------
initloc : 1
Loc 1 (at the beginning of the function 'main')
----------------------------------------
  result := 0
  i := 0
----------------------------------------
  True -> 2   False -> null
Loc 2 (the condition of the 'for' loop at line 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

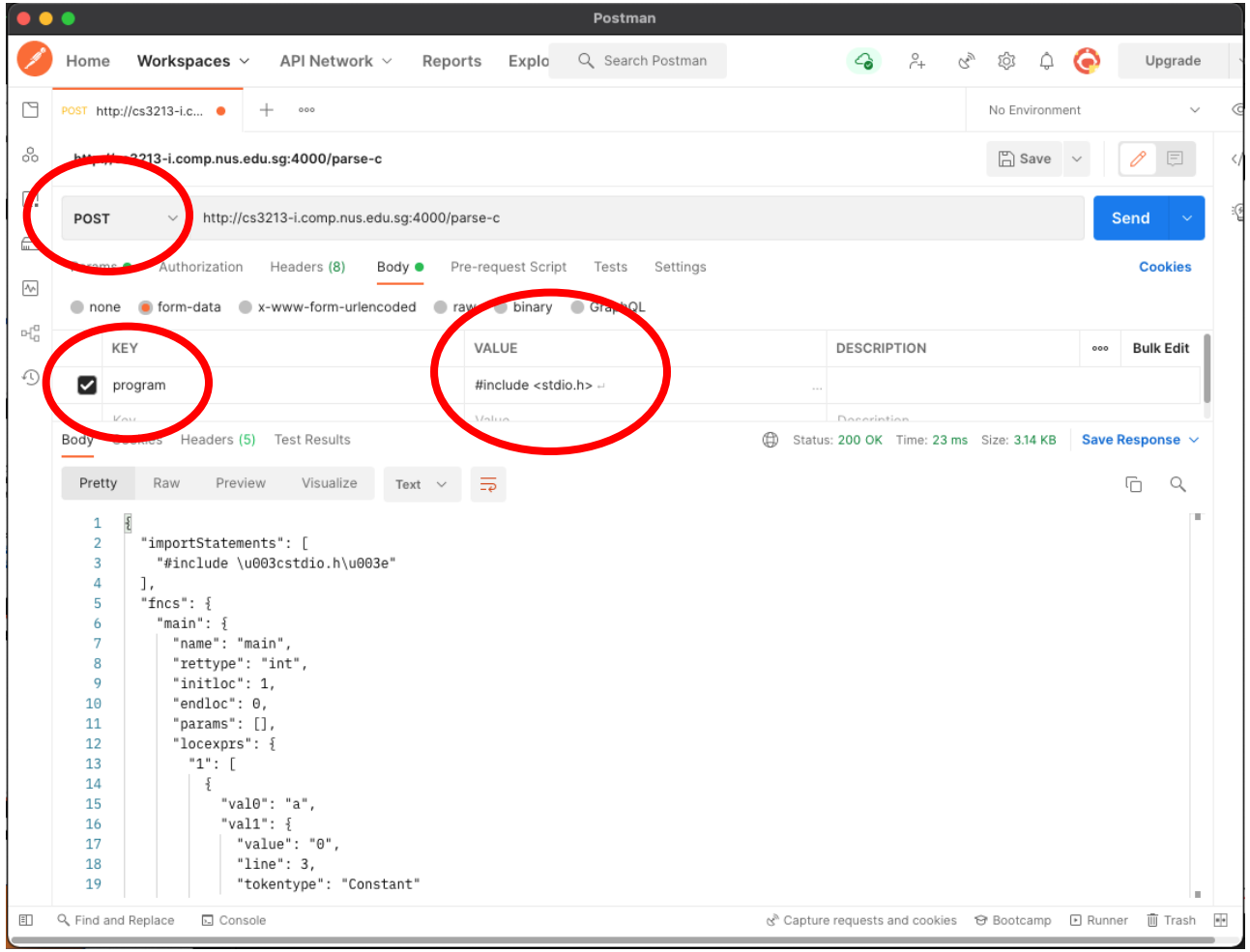


- Input: program in `.c` or `.py` source file
- Output: internal program object in `json` format
- Purpose: prepare test inputs for your test cases / evaluation

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)



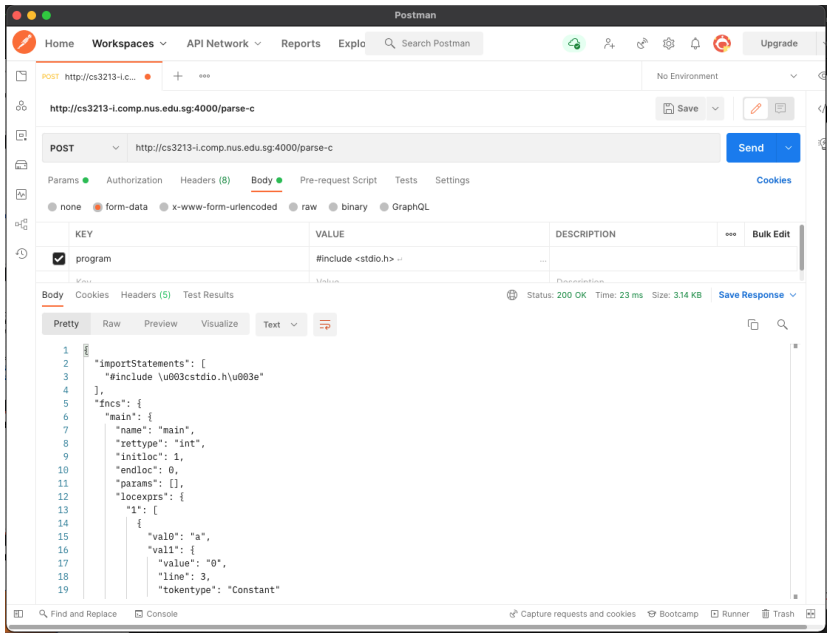
- For example, you can use the tool **Postman**[1] to send `POST` requests to our server
- `POST` body should have the *key* „`program`“ and as *value* the source code

[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




```
{
  "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`

```java
/**
 * 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;
  }
}
```

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

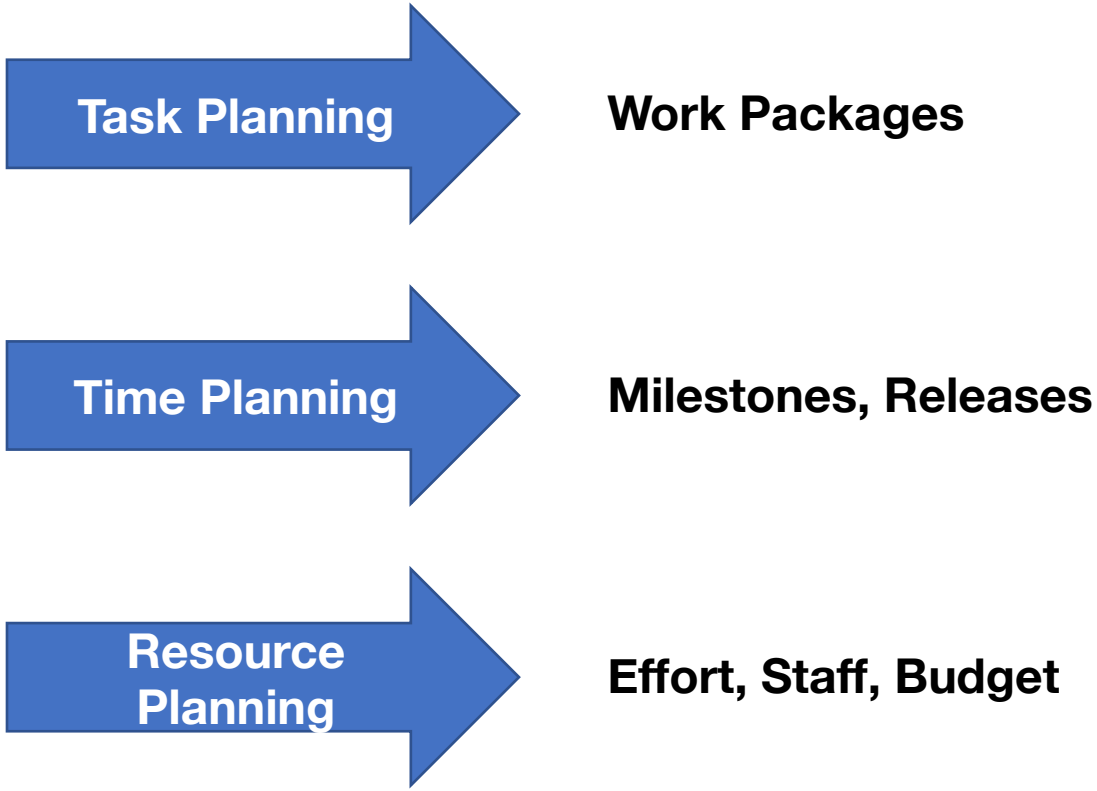
→ `sg.edu.nus.se.its.parser.BasicTest`

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



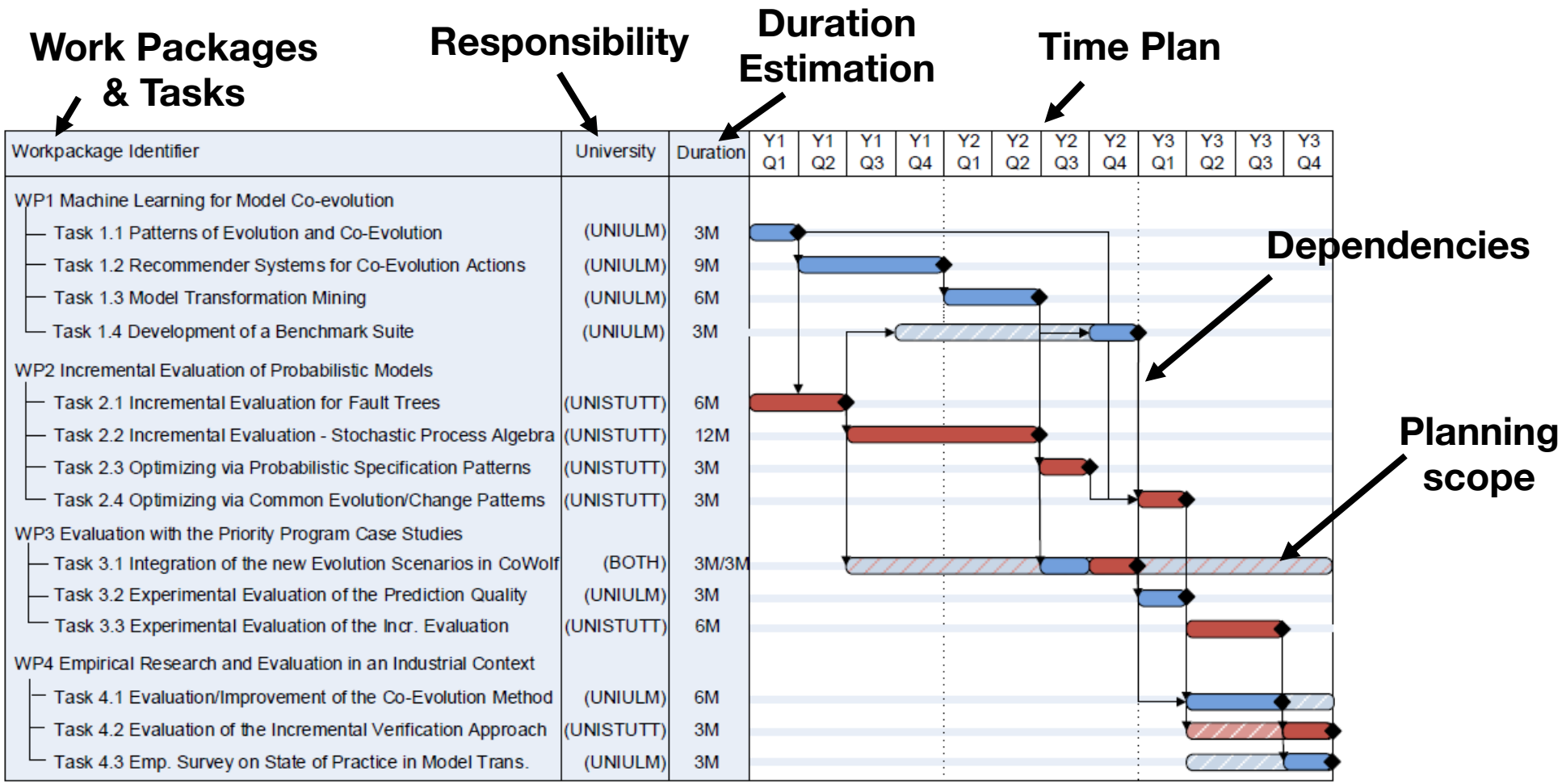
Work Packages

- **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	

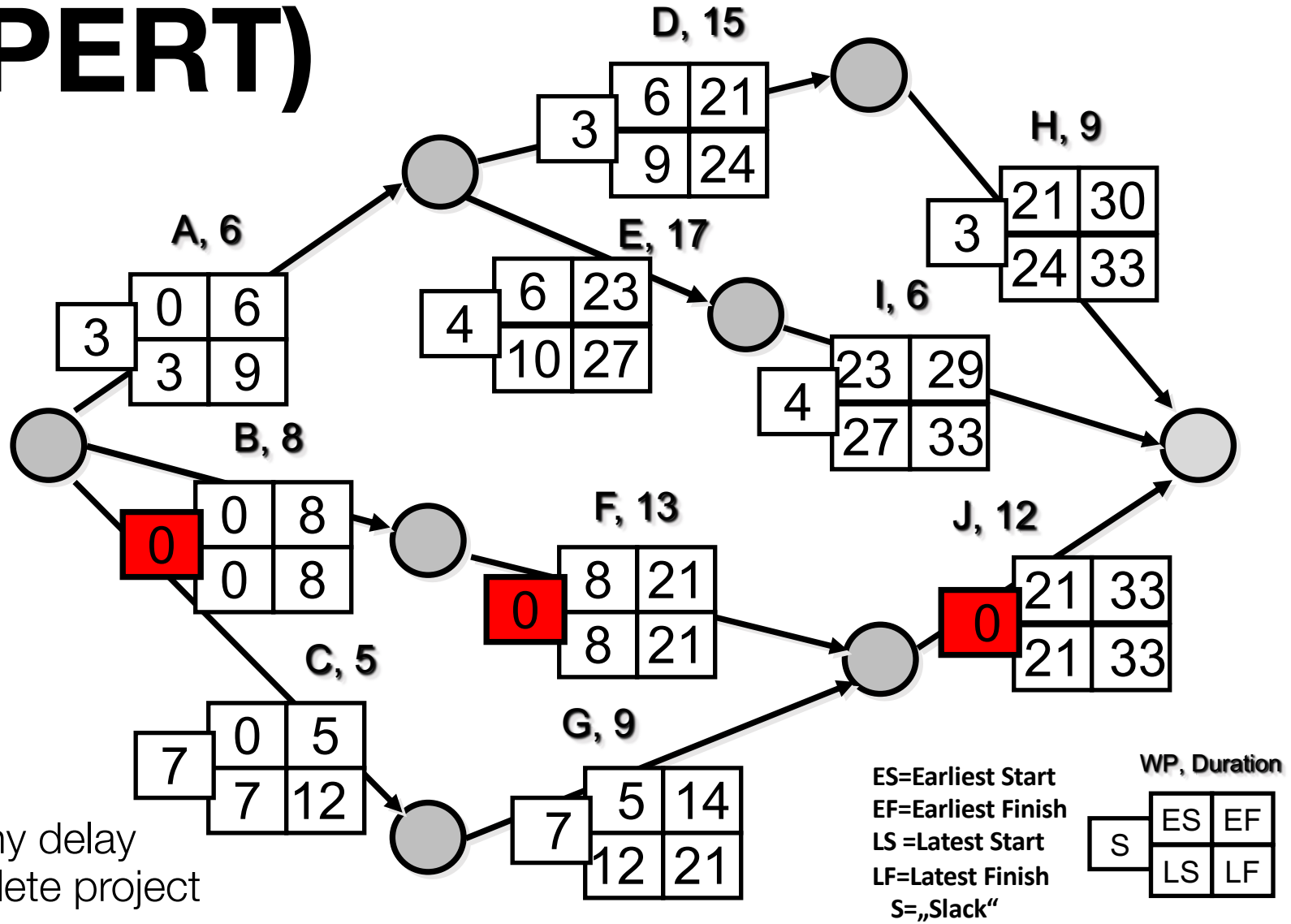
Gantt-Charts (Example)



(Example taken from a Research Project)

Program Evaluation and Review Technique (PERT)

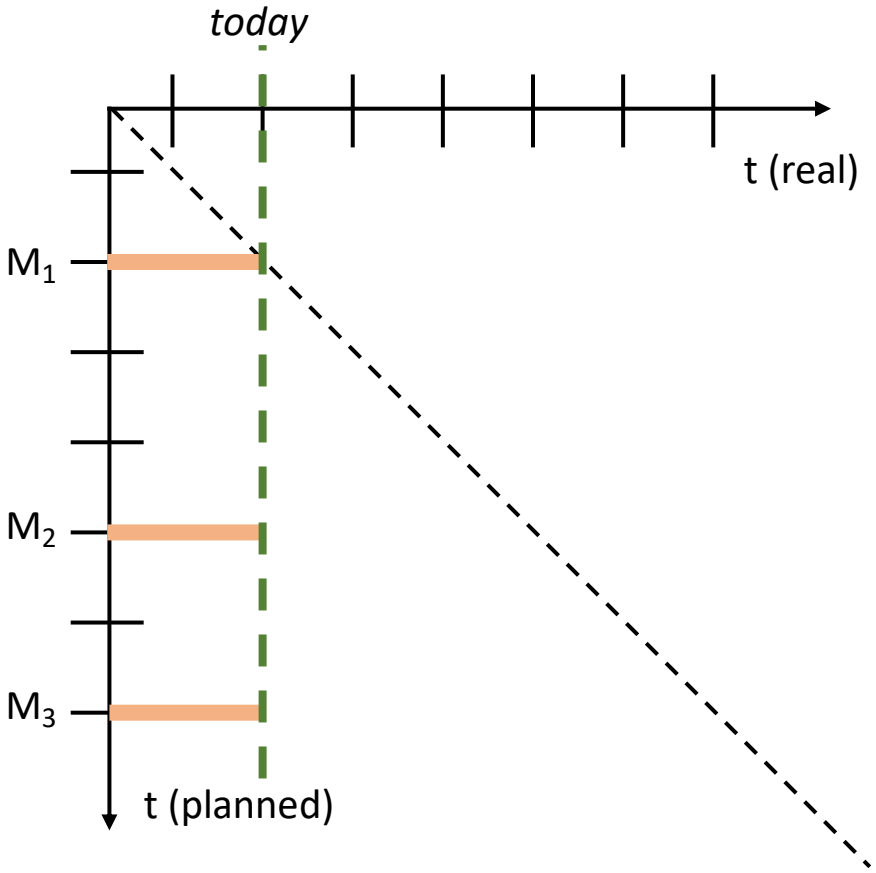
Work Package (WP)	Duration (e.g., days)	Depends on
A	6	–
B	8	–
C	5	–
D	15	A
E	17	A
F	13	B
G	9	C
H	9	D
I	6	E
J	12	F, G

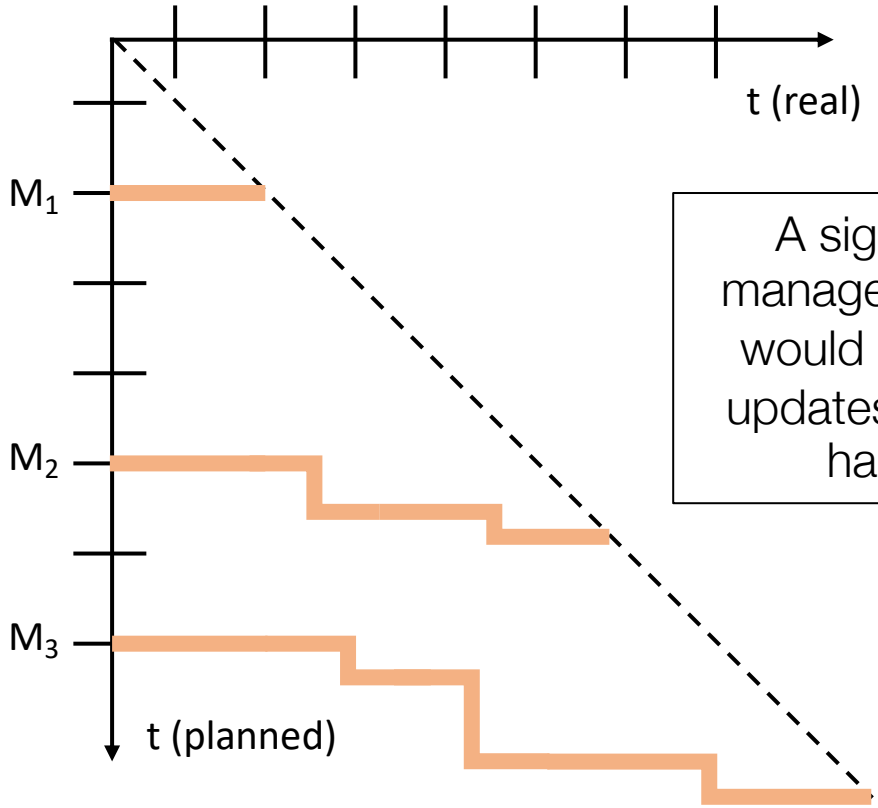


→ Identify the ***critical path***, i.e., any delay along this path will delay the complete project

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)