

CSP Project

Artificial Intelligence

Maitane Gómez Sainz 72316851 – K

Jose M^a González Gamito 45821905 - V



1. Define the problem as a Constraint Satisfaction Problem (CSP). You must define the variables, their value domains and the different constraints clearly specifying their type.

- *Variables and domain*

| Variable | Domain |
|----------|-----------------------------|
| A | Integer numbers from 1 to 9 |
| B | Integer numbers from 1 to 9 |
| C | Integer numbers from 1 to 9 |
| D | Integer numbers from 1 to 9 |
| E | Integer numbers from 1 to 9 |
| F | Integer numbers from 1 to 9 |
| G | Integer numbers from 1 to 9 |

- *Constraints*

- “Distinct” constraint

All the variables must be different from multiplier and constant. Although it is presented as a whole constraint involving all the variables, it is implemented using a constraint per each variable, so it can be defined as unary constraint.

$$(A, B, C, D, E, F, G \neq \text{Multiplier}) \ \&\& \ (A, B, C, D, E, F, G \neq \text{Constant})$$

- “MaxMinutes” constraint

The variables involved in the tens of minutes (A and E) cannot exceed the tens of the maximum minutes allowed. As in the previous case, different constraint will be made per each variable, so it is again a unary constraint

$$A \leq \frac{\text{maxMinutes}}{10} \quad E \leq \frac{\text{maxMinutes}}{10}$$

- “MaxSeconds” constraint

In spite of seconds cannot exceed the value of 59, variables involved in the tens of seconds (C and F) cannot be greater than 5. Furthermore, these constraints will be unary too because they only involved one variable each.

$$A \leq 5 \quad E \leq 5$$

○ “FinalConstraint” constraint

This constraint represents the final check of the equation's values in order to see if the assignment of the variables fulfilled the requisites of the problem.

Just because it involves all the variables in the problem, this constraint is considered as a global one.

During the checking of the constraint, some of the operations taken into account are the followings:

$$CD \times multiplier - \left[\left(\frac{CD \times multiplier}{60} \right) \times 60 \right] == GConstant$$

$$AB \times multiplier + \left(\frac{CD \times multiplier}{60} \right) == EF$$

2. According to your definition indicate what type of CSP you are facing and give the problem's formulation, either incremental or complete state formulation. Justify your answer.

We have decided to do it in an incremental way, because we start from a state where no variable has been assigned and we have to reach a complete state. The consistency of the problem will be checked in every constraint, that is why we only care about reach a complete state.

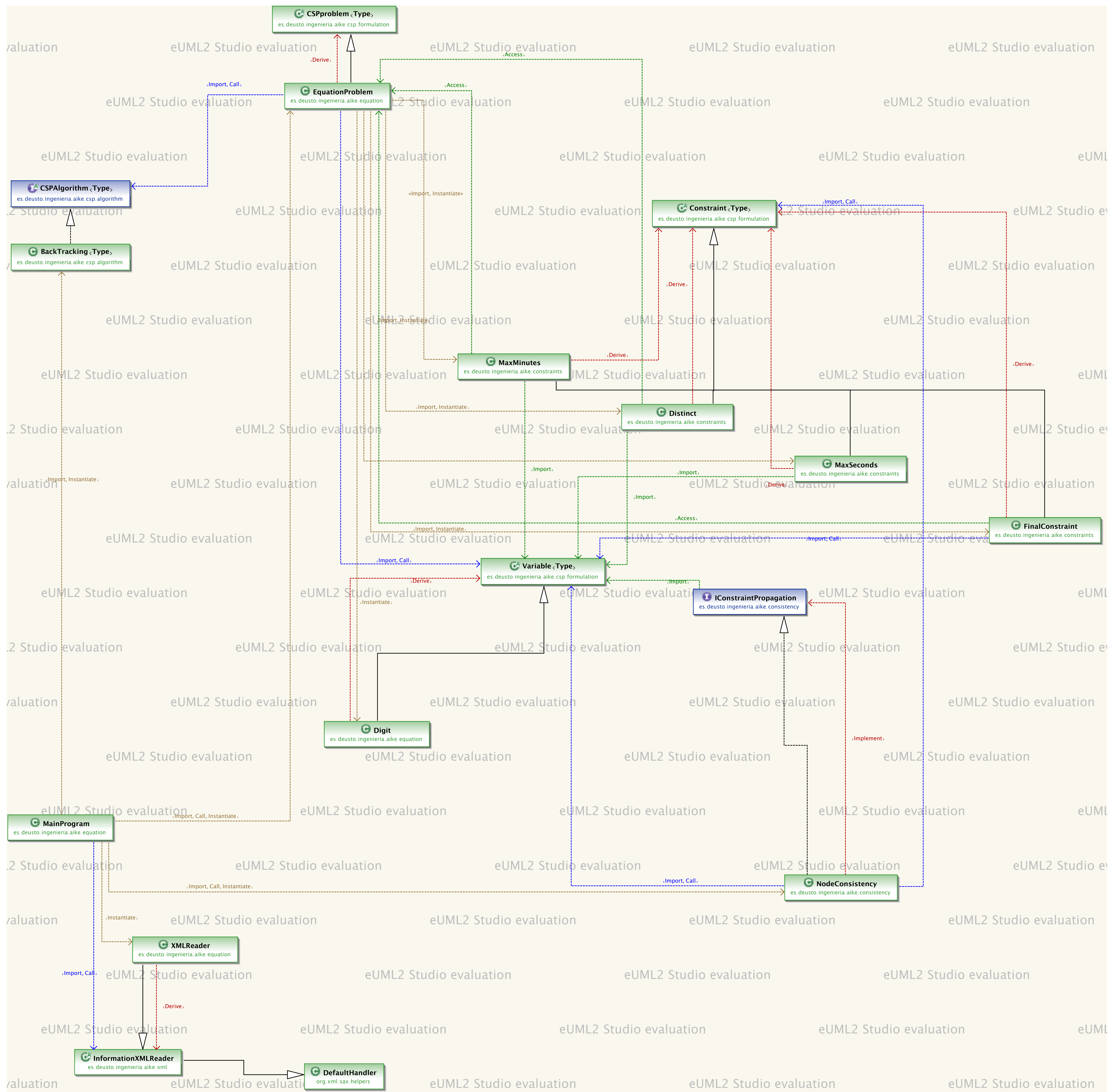
The incremental formulation of the problem would be:

- **Initial state:** no variable assigned.
- **Actions:** assign a value to a variable checking is that assignment fulfilled the constraints.
- **Goal test:** Is the correct assignment complete?

3. You will be required to solve the problem's unary constraints by enforcing Node Consistency before you invoke a search algorithm. Taking this into account, explain which search algorithm is more suitable for this CSP:

We have decided to use the backtracking algorithm because although enforcing node consistency there are still values that can be assigned but they will not be checked in the final constraint until all of the variables have a value. If any of these values do not fulfil that constraint the algorithm should go back and tries another values. That is why we have decided to do backtracking. In addition backtracking uses incremental formulation.

4. UML diagram of the project:



EquationProblem
es.deusto.ingenieria.aike.equation

multiplier: int

constant: int

maxMinutes: int

EquationProblem(multiplier: int, constant: int, maxMinutes: int)

createDigits()

createConstraints()

createDomain(): List<Integer>

toString(): String

solve(algorithm: CSPAlgorithm<Integer>)

Digit
es.deusto.ingenieria.aike.equation

Digit()

toString()

MainProgram
es.deusto.ingenieria.aike.equation

main(args: String[])

XMLReader
es.deusto.ingenieria.aike.equation


XMLReader(xmlFile: String)


getInformation(): Object


startElement(uri: String, localName: String, qName: String, attributes: Attributes)




**IConstraintPropagation**
es.deusto.ingenieria.aike.consistency


 makeConsistent(myVariables: List<Variable<Integer>>)




**NodeConsistency**
es.deusto.ingenieria.aike.consistency


 makeConsistent(myVariables: List<Variable<Integer>>)




**Distinct**
es.deusto.ingenieria.aike.constraints


  Distinct(variables: List<Variable<Integer>>, name: String)
 isSatisfied(variable: Variable<Integer>, value: Integer): boolean




**MaxSeconds**
es.deusto.ingenieria.aike.constraints

  MaxSeconds(variables: List<Variable<Integer>>, name: String)
 isSatisfied(variable: Variable<Integer>, value: Integer): boolean

**MaxMinutes**
es.deusto.ingenieria.aike.constraints

  MaxMinutes(variables: List<Variable<Integer>>, name: String)
 isSatisfied(variable: Variable<Integer>, value: Integer): boolean

**FinalConstraint**
es.deusto.ingenieria.aike.constraints

  FinalConstraint(variables: List<Variable<Integer>>, name: String)
 isSatisfied(variable: Variable<Integer>, value: Integer): boolean