DIDATIC TESTING BENCH FOR AUTOMATED PLANNING

João Paulo da Silva Fonseca, jpaulosfonseca@gmail.com
José Jean-Paul Zanlucchi de Souza Tavares, jean.tavares@mecanica.ufu.br
Federal University of Uberlândia, João Naves de Ávila Avenue, 2121 – Uberlândia MG

Abstract. *In the last 50 years there were developed many tools related with artificial intelligence, such as expert systems, neural networks, genetic algorithms, fuzzy logic and especially, automated planners, however, it has not been properly disseminated in practical applications. Specifically automated planners emerged in 1971 with the STRIPS or "Stanford Research Institute Problem Solver", the first automatic solver problems. The development of the automated planners created a standard formal language called PDDL or "Planning Domain Definition Language". In 2008, itSIMPLE was developed as a knowledge engineering tool used for modeling planning domains to several automatic planners, in order to develop a plan that meets the requirements of the project. ItSIMPLE assists in plans evaluations and to better understand the problem situation, but it is still far from the real applications. It is then necessary evaluate if it is possible to apply the solutions of these tools in practical cases. This paper proposes the development of a didactic testing bench for application of automated planning tools, and thus evaluates the actual distance between theoretical plans and practical systems. In this case, the didactic bench simulates motion systems, widely used in manufacturing process and logistics. With this objective, this bench was developed to simulate a system of product distribution from a supplier to two distinct customers using an autonomous vehicle controlled by a Programmable Logic Controller (PLC), responsible for transporting product programmed in function of the customer stock variation. The supply of product is performed using a water electropump that loads the car at the supplier and unload the car at customers. Each customer caters to an internal electropump in its own reservoir in three different predefined demands, it means, fixed, probabilistic and uncertain. These different demands are based on real cases of a large petrochemical company. The car is commanded by two 12V DC motors so that the vehicle can moves to the right or left side, depending on system needs. There are three mechanical microswitch on the bench, in customers and supplier positions. Each customer and the vehicle have an internal level sensor to assist the product stock control. When the customer level sensor reaches critical level, the customer makes a request for a pre-defined amount of product delivery by the vehicle, which may transport more than customer needs. The level state analysis and vehicle position are PLC inputs; and electropump in charge and vehicle movement are PLC outputs. PLC and automatic planning tools are integrated and a solution-plan example is presented. This bench can split decision accountability from PLC to automated planner tool and it provides practical examples to evaluate automated planners solutions in mechanical systems.*

Keywords*: Programmable Logic Controller – PLC, Automatic Planning Tools, Didactic Testing Bench, Supervision and Control, Supply Chain Management*

1. INTRODUCTION

Industrial automation always deals with new technologies and approaches, although, their implementation requires time and expertise. Artificial intelligence is one of them.

In the last 50 years there were developed many tools related with artificial intelligence, such as expert systems, neural networks, genetic algorithms, fuzzy logic and especially, automated planners, however, it has not been properly disseminated in practical applications. Specifically automated planners emerged in 1971 with the STRIPS or Stanford Research Institute Problem Solver (Fikes and Nilsson, 1971), the first automatic solver problems. The development of the automated planners created a standard formal language called PDDL or Planning Domain Definition Language (Vaquero, 2007).

Despite of PDDL, this area still focused on new automatic problem solvers until 2008, when itSIMPLE was developed as a knowledge engineering tool used for modeling planning domains to several automatic planners, in order to assist plan analysis whether it meets the requirements of the project or not. ItSIMPLE helps plans evaluations and to better understand the problem situation, but it is still far from the real applications. With this assistance, it is possible evaluate if it is able to apply the Automated Planning solutions of these tools in practical cases.

This paper proposes the development of a didactic testing bench for application of automated planning tools, and in thus evaluates the actual distance between theoretical plans and practical systems using PLC – Programmable Logic Controller. This paper focuses on bench development and characteristics. Until today automated planning tools are still applied to theoretical problems. With this bench it is possible to verify and validate automated planning results for a specific and didactic system through itSIMPLE modeling process, assisting automatic planning tools deployment.

This paper presents Strips and Artificial Intelligence review in section 2, followed by itSimple. Section 4 shows Didactic Testing Bench schema. ItSIMPLE Didactic Test Bench model is presented in Section 5 and the Didactic Testing Bench integrated with PLC (Programmable Logic Controller) solution is showed in section 6. Discussion and conclusion are presented in section 7, followed by Acknowledgments, References and Responsibility Notice.

2. STRIPS AND ARTIFICIAL INTELLIGENCE

The creation of Artificial Intelligence occurred in 1940s when McCulloch and Pitts (1943) proposed an artificial neural network whose goal was to simulate the human brain in computational operations. Since then, there were developed many tools related with artificial intelligence, such as expert systems, neural networks, genetic algorithms, fuzzy logic and especially, automated planners, however, it has not been properly disseminated in practical applications.

The history of Automated Planning as an area of Artificial Intelligence began in the 1960s, from scientific work focused on general problem solvers development (especially with the use of first order logic). However, only in the early 1970s, a planner able to effectively make use of representations of the domains during the obtaining solutions to problems was proposed by researchers at Stanford Research Institute. Emerged here the STRIPS (Stanford Research Institute Problem Solver) (Fikes and Nilsson, 1971), it would be, beyond a reference, a pioneer in the field of Automated Planning.

The STRIPS was very famous for its formulation and representation of actions (or operators). With a simple formulation, this planner was the beginning of the Automated Planning Classical Era that lasted until the beginning of the 1990s (Ghallab *et al.*, 2004 *apud* Vaquero, 2007).

In mid 1995, the story of Automated Planning got a big boost when Avrim Blum presented the planner GRAPHPLAN (Blum and Furst, 1995) which used a method of extracting plans differentiated by the graphs. Its simplicity combined with its superior performance to the planners of the time stimulated the development of new techniques and research planning. This planner marked the beginning of the Automated Planning Neoclassical Era which revived the research on the classical planning problems.

3. ITSIMPLE

The itSIMPLE - Integrated Tools Software Interface for Modeling PLanning Environments – is an integrated design environment whose the main objective is minimize the problems found during the project life cycle and real applications of planning, predominant phases of requirements, modeling and analysis, when the different participants viewpoints should be taken into consideration (Vaquero, 2007). In 2008, itSIMPLE was developed as a knowledge engineering tool used for modeling planning domains to several automatic planners, in order to develop a plan that meets the requirements of the project. ItSIMPLE assists in plans evaluations and to better understand the problem situation, but it is still far from the real applications.

The itSIMPLE have flexibility to work with different languages, such as UML (Unified Modeling Language), XML (eXtensible Markup Language), PDDL (Planning Domain Definition Language) and Petri Nets, moreover, the designer can use the same features modeled in *Use Case*, *Class* and *States Diagrams* to also evaluate this situation with different agents and resources as well as new restrictions.

A planning domain modeling with itSIMPLE follows the sequence described by UML (Unified Modeling Language) literature. Initially, the *Use Case* Diagram is drawn up – this step the designer defines the constraints (pre-conditions and post-conditions) for each *Use Case*; next the designer draw the *Activity* Diagram – stage where one has the action and decisions necessary for the problem objective is defined; the next step it is the preparation of *Class* Diagram – where it is done the modeling of the domain's static structure based on the description of use cases; the next UML diagram is the *State* Diagram – here the designer is responsible by the relevant classes dynamics aspect definition; the development of the last diagram it is the problem modeling and is known as *Object Diagram* – this step is divided into three diagrams namely the *repository*, the *initial snapshot* (where is determined the initial scene) and the *snapshot goal* (where is determined the goal scene). The *Snapshot* allows the user to instantiate classes (creating objects), give value to the attributes of each instance of classes and associate the objects according to the situation that the designer wants to build. (Vaquero, 2007)

With the aim of clarify this modeling will present a simple example modeled in itSIMPLE. The *Blocks world* is one of the most popular domains of Automated Planning in AI. This domain is composed by a robotic arm, three blocks and a table. The arm can move only one block at a time. The Figure 1 illustrates this domain.

As described previously, the modeling process starts by building the *Use Case Diagram*. Based on the characteristics of the *Classic Blocks World*, the diagram could be constructed with only four *use cases* (with their restrictions), it means, "*Pick up block*", "*Put down block*", "*Stack block*" and "*Unstack block*". All of these *use cases* are performed by the robotic arm, in other words, in this domain there is only one agent *Hand*, as presented in Fig. 2.

The *Blocks world Class Diagram* is modeled based on the description of *Use Cases*. The main elements of the *Class Diagram* are *blocks*, *hand* and *table*, as showed by Figure 3. In this model only two classes have dynamics aspects relevant to be represented and analyzed: *Hand* (*Agent Class*) and *Block* (*Resource Class*). Focusing exclusively in the *Hand class* is possible to trace the transitions between the states (the actions) and their respective pre and post conditions in the *Hand State Diagram* (Figure 4).

Following the modeling process, the planning problems can be modeled by two *Snapshots* (*Objects Diagram*) representing the *initial state* and the *final* (goal) *state* of the problem with three different *blocks*: A, B and C (Fig. 5).

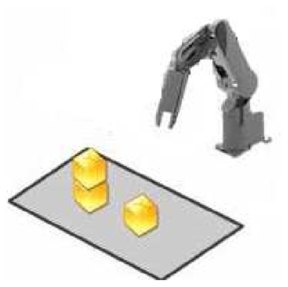

Figure 1. Illustration of the *Blocks World*.

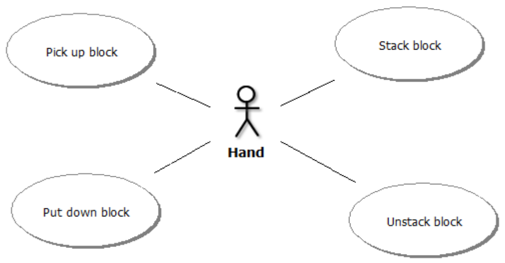

Figure 2. *Use Case Diagram* of the *Blocks World*.

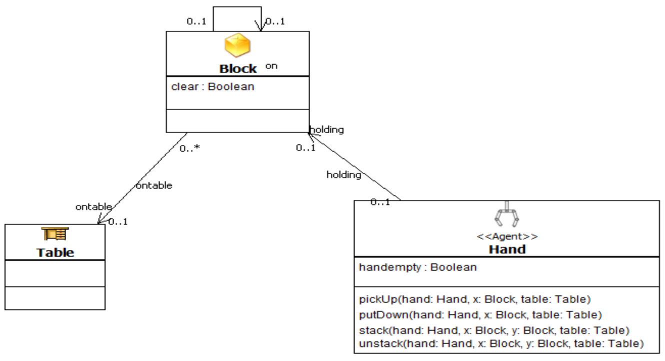
0..1
0..1
Block
on
clear : Boolean
holding
0..*
0..1
ontable
holding
ontable
0..1
0..1
Table
<<Agent>>
Hand
handempty : Boolean
pickUp(hand: Hand, x: Block, table: Table)
putDown(hand: Hand, x: Block, table: Table)
stack(hand: Hand, x: Block, y: Block, table: Table)
unstack(hand: Hand, x: Block, y: Block, table: Table)

Figure 3. *Class Diagram* of the *Blocks World*.

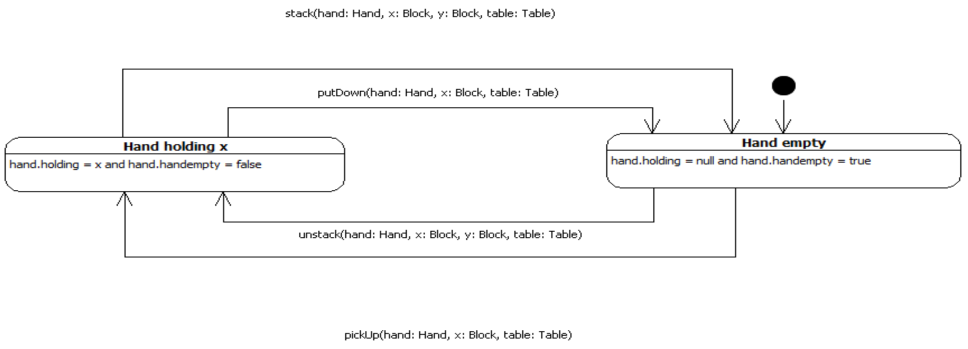
stack(hand: Hand, x: Block, y: Block, table: Table)
putDown(hand: Hand, x: Block, table: Table)
Hand holding x
hand.holding = x and hand.handempty = false
Hand empty
hand.holding = null and hand.handempty = true
unstack(hand: Hand, x: Block, y: Block, table: Table)
pickUp(hand: Hand, x: Block, table: Table)

Figure 4. *State Diagram* of the class *Hand*.

Initial state presents *A block* on *B block* on *C block* on the *Table*. *Final state* presents *C block* on *B block*, on *A block* on the *Table*.

This domain, modeled in UML is automatically converted to PDDL by itSIMPLE, and this result can be accessed by different planners. These planners, according to their functionality, develop an *action plan* from the *initial state* to the goal. Each planner is free to generate a different plan, everything will depend on their characteristics, their robustness and the platform it is inserted.

In this planning problem, the planner must use the *hand* agent to modify the *block*'s position in order to reach the *final state*. In the test phase with planners, for verification and refinement of the model, this domain was performed with the algorithm Metric-FF (Hoffmann, 2003), one of several planners available in itSIMPLE.

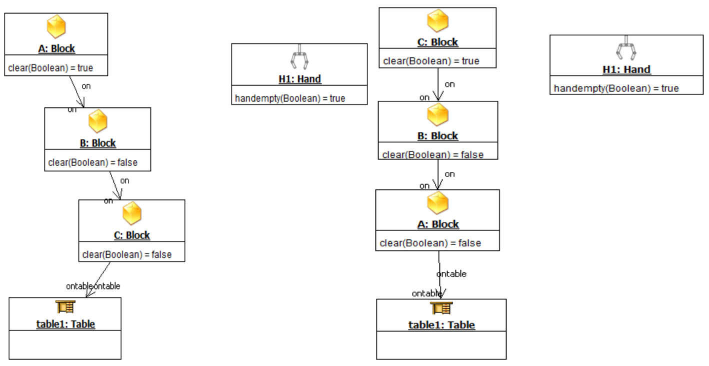

Figure 5. *Initial* and *Final Snapshot* of the planning problem related with *Blocks World*.

The Metric-FF solved the problem modeled with an *action plan* of 6 steps. The solution-plan for this planning problem is represented bellow.

```
0: UNSTACK H1 A B TABLE1
1: PUTDOWN H1 A TABLE1
2: UNSTACK H1 B C TABLE1
3: STACK H1 B A TABLE1
4: PICKUP H1 C TABLE1
5: STACK H1 C B TABLE1
```

4. DIDACTIC TESTING BENCH SCHEMA

From the issue raised, the distance between the use of automated planning software and implementation in real cases, this paper proposes to develop a didactic testing bench. The Fig. 6 illustrates the initial idea for the development of this bench.

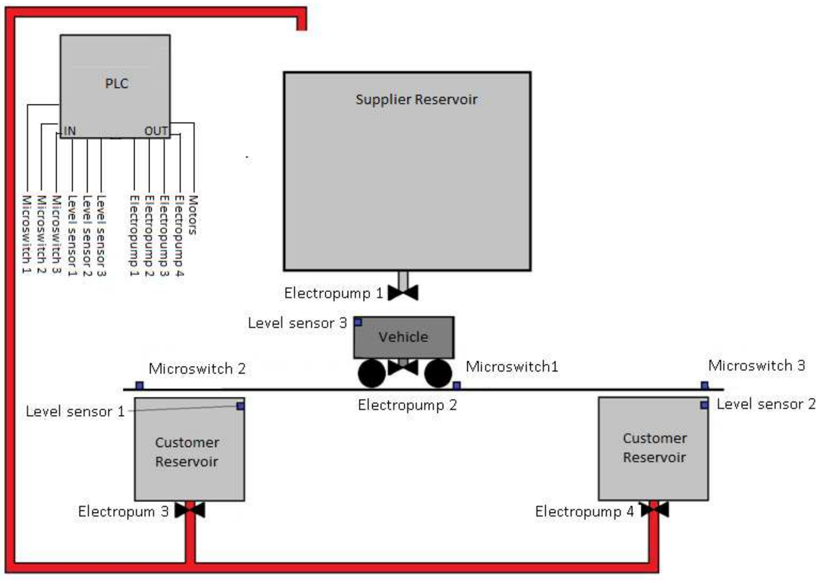

Figure 6. Illustration of the proposed system.

In this case, the didactic bench simulates motion systems, widely used in manufacturing process and logistics. With this objective, this bench was developed to simulate a system of product distribution from a supplier to two distinct customers using an autonomous vehicle controlled by a Programmable Logic Controller (PLC), responsible for transporting product programmed in function of the customer stock variation.

As shown in Fig. 6, the Didactic Test Bench is composed by 1 (one) vehicle, 1 (one) supplier reservoir, 2 (two) customers reservoirs and 2 (two) client demands electropumps. The vehicle must receive product on the supplier reservoir and carry it up the customer reservoirs. The level of each customer reservoir will be a function of client demand for each customer, represented here by electropumps. These client's electropumps simulate three different types of demands (it means fixed, probabilistic and uncertain) based on real cases of a large petrochemical company. The product comes as demand return to the supplier reservoir, closing the cycle and ensuring the continued functioning of the system.

5. DIDACTIC TEST BENCH MODEL IN ITSIMPLE

The modeling process begins by the construction of the *Use Cases Diagram*. An analysis of the characteristics of the proposed problem allows us to notice that this diagram is composed of two agents, one vehicle and another customer. The agent *Vehicle* will be the responsible for carrying out of *Use Cases Move*, *Load* and *Unload*. While the agent *Customer* will be the responsible by the *Use Cases Unload*, *PartialSale* (for partial deliveries), *FinalSale* (to fulfill partial deliveries) and *CompleteSale* (for full deliveries). The *Use Case Unload* requires the activities of the agents *Vehicle* and *Customer* simultaneously. The Fig. 7 illustrates the *Use Case Diagram* of the domain in question.

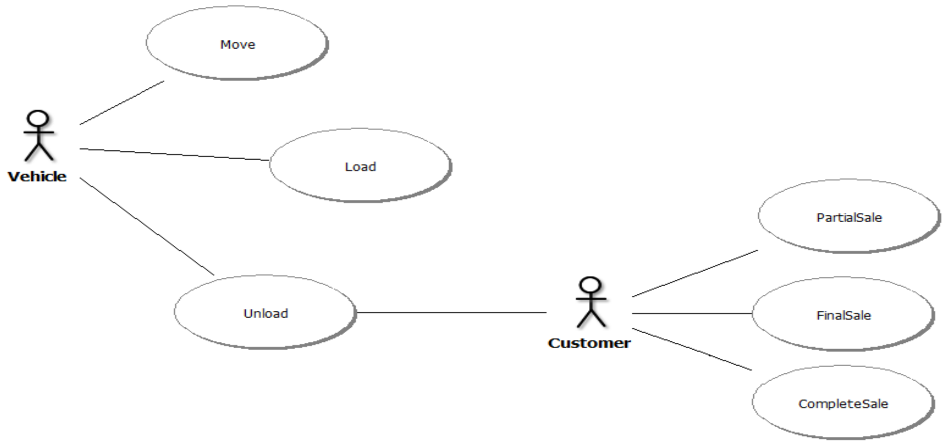

Figure 7. *Use Case Diagram* of the Bench.

Following modeling process the static structure of the domain represented by *Classes Diagram* must be done based on the description of the *Use Cases*. The main elements of this domain are the *Vehicle*, the *Customers*, the *Supplier* and the *Client Class*. In addition to these *Classes*, the diagram is formed by the *LevelTransf* (responsible for the discretization of quantities of product sold and displayed by the level sensors) and *Global* (containing all global variables of the domain), the first being a *Resource Class* and the second a G*lobal Class* (*stereotype <<utility>>*).

In this model the *Vehicle Class* has two attributes: *maxlev* (Int), identifying the maximum level of the vehicle's reservoir; and *lev* (Int), representing the current level. The *Customer* and *Supplier Class* are generalizations of the *Place Class*, which has a single attribute *busy* (Boolean), identifying the state's place as the presence or absence of vehicle. Besides the attribute *busy*, inherited by the generalization, the *Customer Class* has more three attributes: *capacity* (Int), identifying the capacity of the reservoir; *level* (Int), representing the current level of product; and *critical_level*, identifying the critical level of the customer. The *ClientDemand Class* has three attributes: *amount_requested* (Int), representing the amount requested by demand; *amount_received* (Int), representing the amount received so far; and *attended* (Boolean), identifying if their demand has been met or not. The *LevelTransf Class* has only one attribute called *amount_transfered*, which represents the discretized value of the transfer level. Finally, the *Global Class* has three attributes: *distance (p1:Place, p2:Place)* (Int) symbolizing the distances between places in the domain (values in cm); *transportcost* symbolizing the cost of transport in a real system, to solve planning problem (minimization goal); and *lostcost* representing the cost for an incomplete delivery, to solve planning problem (another minimization goal).

Moreover, the *Vehicle Class* has an association *isAt* with the *Place Class* in order to identify which place the vehicle is at the exact moment, and the *ClientDemand Class* has an association *buysfrom* with the *Customer Class* to identify which is the *Customer* responsible for fulfill the demand of each *Client*.

To ensure proper functioning of the system, *Agents Classes* must take actions to ensure the functionality of the plant. So the *Vehicle Class* has three operators: *move*, *load* and *unload*. And the *Customer Class* has other three operators to ensure the supply demand: *partialsale, finalsale* and *completesale*.

The *Class Diagram* resultant of the static structure modeling of the model is show in Fig. 8.

In this model, only two classes have dynamics aspects relevant to be represented and analyzed: *Vehicle* and *Customer* (both *Agent Class*). For instance, for the *Vehicle Class*, the behavior can be model taking in consideration the following points:

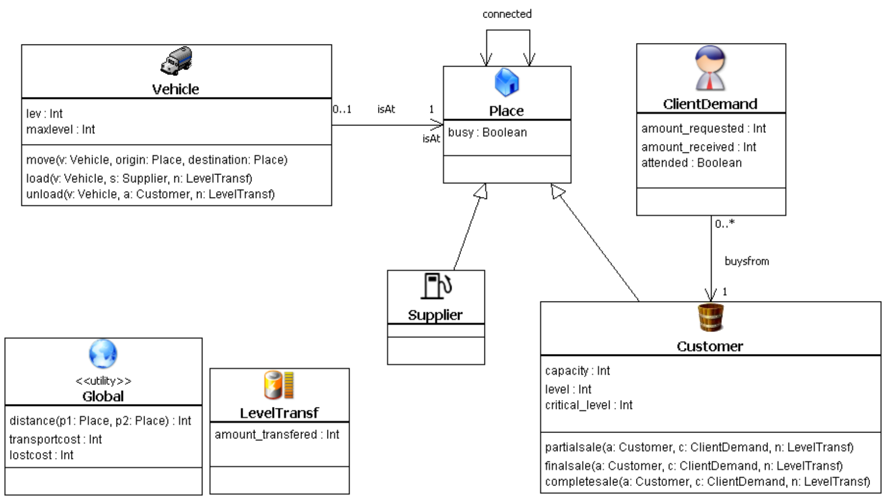
connected
Vehicle
lev : Int
maxlevel : Int
move(v: Vehicle, origin: Place, destination: Place)
load(v: Vehicle, s: Supplier, n: LevelTransf)
unload(v: Vehicle, a: Customer, n: LevelTransf)
0..1
isAt
1
isAt
Place
busy : Boolean
ClientDemand
amount_requested : Int
amount_received : Int
attended : Boolean
0..*
buysfrom
1
Supplier
Customer
capacity : Int
level : Int
critical_level : Int
partialsale(a: Customer, c: ClientDemand, n: LevelTransf)
finalsale(a: Customer, c: ClientDemand, n: LevelTransf)
completesale(a: Customer, c: ClientDemand, n: LevelTransf)
<<utility>>
Global
distance(p1: Place, p2: Place) : Int
transportcost : Int
lostcost : Int
LevelTransf
amount_transfered : Int

Figure 8. *Class Diagram* of the Bench Domain.

1. An *agent* object of *Vehicle type* can be found in three relevant states: "*Moving from an origin place to a destination place*", "*Stopped in the Supplier place and Load the Vehicle's reservoir*" and "*Stopped in the Customer place and Unload product*";
2. Actions that can affect an object of *Vehicle type* are all that it performs, in order words: *move, load* and *unload* (performed by own *Vehicle Class*);
3. The pre and post-conditions of actions that the objects of *Vehicle Class* performs are extracted from descriptions of *Use Cases* and these are represented in OCL (*Object Constraint Language*) [OMG - Object Management Group, 2003];

The Fig. 9 shows the *States Diagram* of the class *Vehicle*.

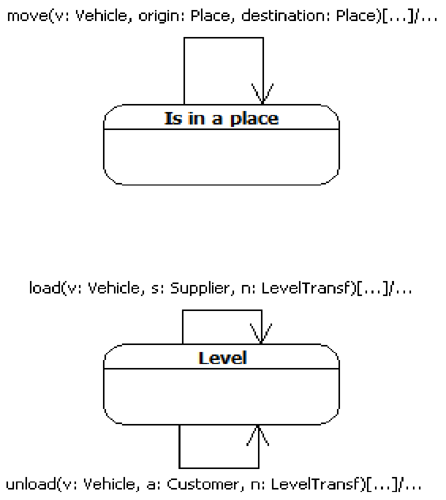
move(v: Vehicle, origin: Place, destination: Place)[...]/...
Is in a place
load(v: Vehicle, s: Supplier, n: LevelTransf)[...]/...
Level
unload(v: Vehicle, a: Customer, n: LevelTransf)[...]/...

Figure 9. *States Diagram* of the Class *Vehicle*.

As a result of the union of expressions of *States Diagram* of the two classes *Vehicle* and *Customer*, it is need to represent all actions in OCL. Following the representation of the action in OCL related with *Move* action from *Vehicle*, with pre and post conditions.

```
    context Vehicle::move(v: Vehicle, origin: Place, destination: Place)
pre:
  -- Vehicle conditions
    v.isAt = origin and origin.connected->exists(p : Place | p = destination) and origin.busy = true and destination.busy = false
post:
  -- Vehicle conditions
    v.isAt = destination and destination.busy = true and origin.busy = false and transportcost = transportcost + distance(origin,destination)*10
```

As described above, the global variable *transportcost*, observed in *Class Diagram* sums each *vehicle's move* action distance and is multiplied by a standard cost of 10 (ten).

After these representation, the planning problems can be modeled by two distinct *Snapshots* (*Objects Diagram*) representing the initial state and de final (goal) state of the problem. For didactic reasons, this paper admits only two *Clients* (*c1* and *c2*) fulfilled by *Customer a1* and *a2* respectively. As initial situation that the *Customer a1* are at *level 1* and the *Customer a2* are at *level 2*, the *Vehicle* is at *zero level* of product, and the *Vehicle* stays at the *Supplier* position. The *Customer a1* receives a demand of 9 (nine) from the *Client c1* and the *Customer a2* receives a demand of 8 (eight) from the *Client c2*. From this scene the planner may reach *Clients* demands minimizing the transport cost and lost cost due partial sales. Thus, the final scene is the *Vehicle* parked at Supplier position and Customers *c1* and *c2* with *level 10*.

This problem is show in Fig. 10 (represented the *Snapshot Initial*) and in Fig. 11 (represented the *Snapshot Goal*).

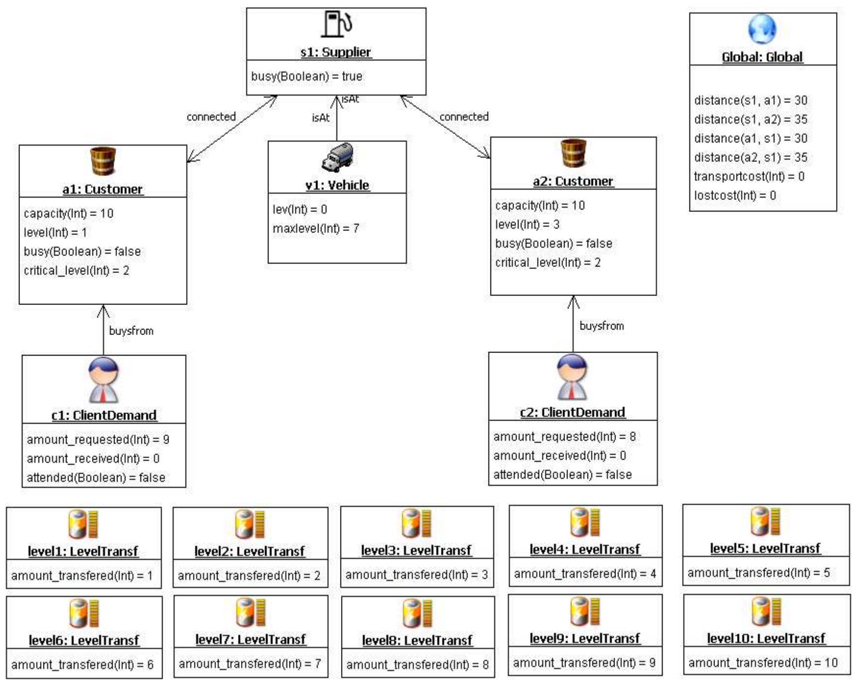

Figure 10. *Snapshot Initial* of a planning problem in the Bench Domain.

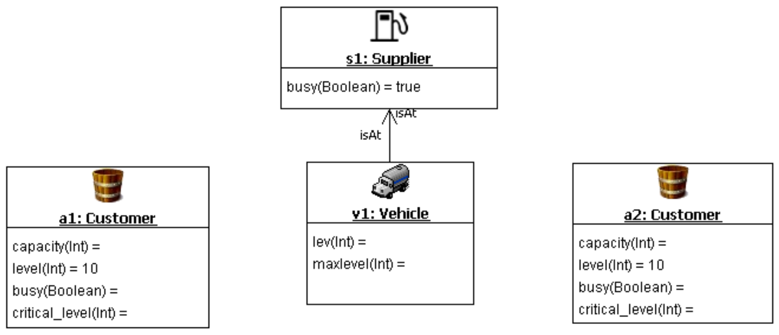

Figure 11. *Snapshot Goal* of a planning problem in the Bench Domain.

The test phase for verification and refinement of the model was performed with the Metric-FF algorithm (HOFFMANN, 2003). The planner Metric-FF solved the modeled problem with a plan composed by 12 (twelve) steps. The solution-plan for this problem, illustrated in Fig. 10 and Fig. 11, is represented bellow.

```
0: LOAD V1 F1 LEVEL 5
1: MOVE V1 F1 A2
2: UNLOAD V1 A2 LEVEL 5
3: MOVE V1 A2 F1
4: LOAD V1 F1 LEVEL 7
5: MOVE V1 F1 A1
6: UNLOAD V1 A1 LEVEL 7
7: MOVE V1 A1 F1
8: LOAD V1 F1 LEVEL 4
9: MOVE V1 F1 A2
10: UNLOAD V1 A2 LEVEL 4
11: MOVE V1 A2 F1
```

6. DIDACTIC TESTING BENCH INTEGRATED WITH PLC SOLUTION

The bench and the vehicle were developed in wood. The Bench's project can be viewed in Fig. 12 and the Vehicle's Project in Fig. 13.

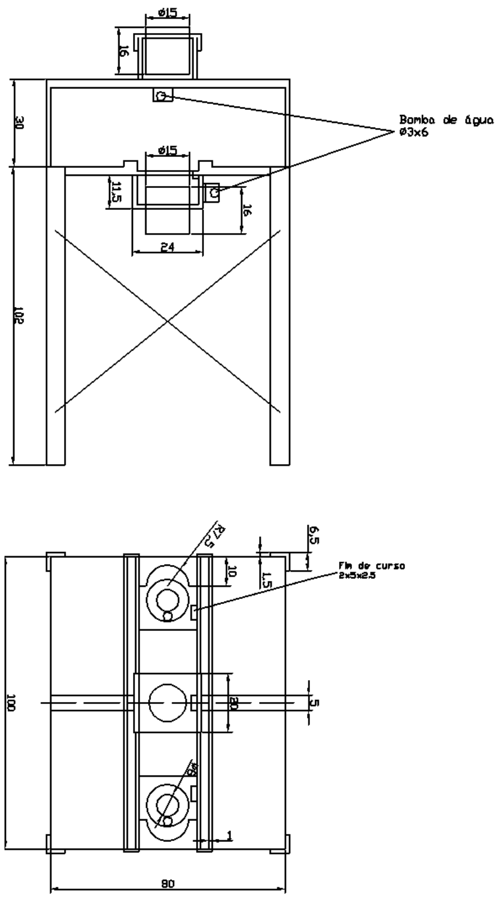

Figure 12. Front view and top view of the bench.

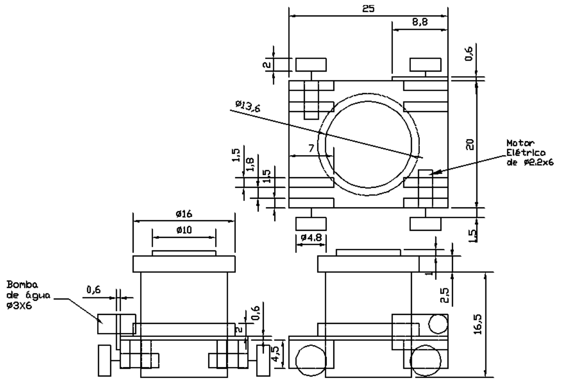

Figure 13. Top view, right side view and front view of the vehicle.

The electrical project for the system can be visualized in Fig. 14. The movement of the vehicle requires the performance of two motors. To control these motors this Bench uses the *L293 Driver*. The electropumps are powered by an external source 12V/20A and electrical relays are used for power transmission. Current transmitters provide level sensors' signal processing for 4-20mA PLC analog input. These transmitters are a Wheatstone bridge type.

The solution-plan with twelve actions must be translated in *Ladder* Language. Each action can be viewed as a specific PLC memory address as described in Tab. 1. These memory address can turn each action on when pre conditions are validated. Figure 15 presents partial *Ladder* Diagram related with the four initial actions (W10.00 to W10.03). This Bench is using Onrom CLP CJ1M CPU13 ETN (ONROM Corporation, 2001).

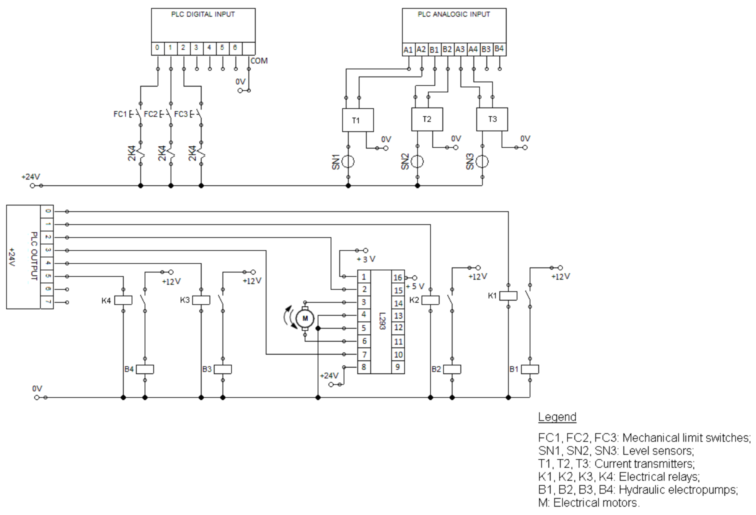
PLC DIGITAL INPUT
PLC ANALOGIC INPUT
COM
0V
+24V
PLC OUTPUT
+24V
L293
FC1
FC2
FC3
2K4
SN1
SN2
SN3
T1
T2
T3
K1
K2
K3
K4
B1
B2
B3
B4
M
+12V
+5 V
+3 V
Legend
FC1, FC2, FC3: Mechanical limit switches;
SN1, SN2, SN3: Level sensors;
T1, T2, T3: Current transmitters;
K1, K2, K3, K4: Electrical relays;
B1, B2, B3, B4: Hydraulic electropumps;
M: Electrical motors.

Figure 14. Electrical scheme for the system Bench.

Table 1. Stage x PLC Memory Address

Stage	PLC Memory Address
0	W10.00
1	W10.01
2	W10.02
3	W10.03
4	W10.04
5	W10.05
6	W10.06
7	W10.07
8	W10.08
9	W10.09
10	W10.10
11	W10.11

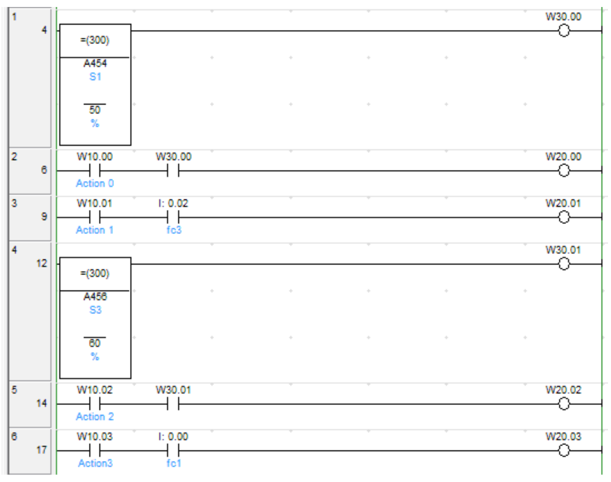

Figure 15. Partial *Ladder Diagram* for the planning domain Bench.

This paper results is the physical *Testing Bench* which is able to receive solutions from planner by *Ladder* programs and execute them. The Fig.16 shows physical *Bench´s* photos.

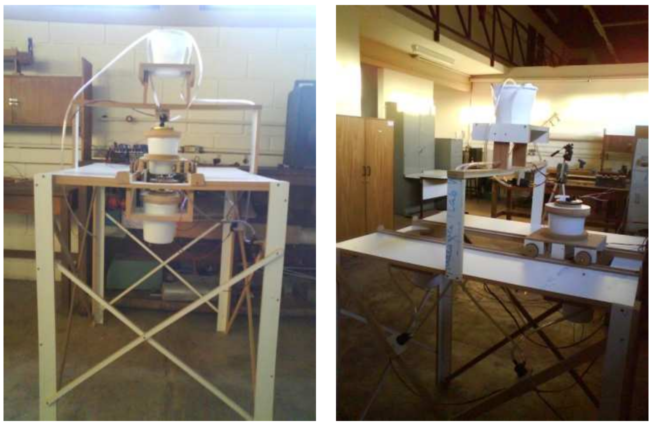
Figure 16. *Bench's* photos.

7. CONCLUSIONS AND FUTURES WORKS

This paper showed a *Didactic Testing Bench* integrating automated planning tools and PLC. It can be possible to note that automated planning tool allows optimization results, in this case, cost minimization. With itSIMPLE Model it is possible to generate several initial and final Snapshots, related with real cases. Each generated solution-plan action must be mapped as PLC Language, in this case, Ladder Diagram, to be implemented in real Bench. On the other hand, it is not possible to generate a cyclic and recursive solution, it means, each problem requires another initial snapshot and a new solution-plan must be created. In this direction, itSIMPLE and PLC must be integrated properly with a specific interface to reach real-time system requirements. This is an initial study which intends to stimulate automated planning deployment. In this direction, only an ordinary example was presented to demonstrate how an action plan can be mapped in Ladder Diagram. There is a need to compare planner solutions in more complex examples, as Tavares and Fonseca (2011).

The future work is the development of an automated interface between the automatic planner and the PLC. This interface is better described in Tavares *et al* (2011)

8. ACKNOWLEDGEMENTS

Authors are grateful to Prof. Dr. José Reinaldo Silva, Prof. Dr. Marcos Antônio Viana Duarte, Dr. Tiago Stegun Vaquero, Kauê Ribeiro, USP, UFU, FAU, CNPQ and FAPEMIG.

9. REFERENCES

Blum, A. L. and Furst, M. L. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90:281-300. 1995.

Fikes, R. E. and Nilsson N. J. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208. 1971.

Ghallab M., Nau D., and Traverso P. Automated Planning: Theory and Practice. Morgan Kaufmann. 2004.

Hoffmann, J. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*. Accepted for special issue on the 3rd International Planning Competition. 2003.

McCulloch, W. S. and Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115-133.

ONROM Corporation (Japan) (Org.). Programmable Controllers: Operation Manual. Tokyo, 2001. CD-ROM.

Tavares, J.J.P.Z.S., Fonseca, J.P.S. Supply Chain Didactic Testing Bench With Automated Planning Tool. In *Proceeding of 21st International Conference on Production Research*. Sttutgard, Germany, 2011. To be presented.

Tavares, J.J.P.Z.S., Fonseca, J.P.S., Vaquero, T.S., Silva, J.R.. Integração de Planejamento Automático e Sistemas Reais Baseados em CLP. In *Proceedings of X Simpósio Brasileiro de Automação Inteligente – SBAI2011*. São João Del Rey, Brazil, 2011.To be presented.

Vaquero, T.S., 2007. "ITSIMPLE**:** Ambiente integrado de análise de domínios de planejamento automático".2007. 316 f. Dissertação (Mestrado) - Universidade de São Paulo, São Paulo.

10. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.