Automated course configuration based on automated planning: framework and first experiments*

Carla Limongelli1, Andrea Sterbini2 & Marco Temperini3
1 DIA, Università Roma Tre, Italy
2 DI, Università di Roma “La Sapienza”, Italy
3 DIS, Università di Roma “La Sapienza”, Italy

Abstract
The work presents an application of automated planning to the process of course configuration. Configuring a course means to build a course based on the effective needs of the specific learner. The course is built by selecting and sequencing learning objects (learning components) from a repository. Parameters in that process are the starting knowledge that the learner already possess and the target knowledge that is expected to be gained through the course.

In order to have more powerful ways to reason on its properties and structure we propose to treat this problem as a planning problem: a configuration is a plan, a component is an action in the plan; required/acquired knowledge are modeled, resp., as action preconditions and postconditions. The initial state of the plan encodes the student’s starting knowledge, while the goal of the plan encodes the course target knowledge.

We have run a short experiment on a class of 26 third-year undergraduate students, regularly attending the course of Algorithms and Data Structures.

1 Introduction

In this paper, we extend the original framework proposed in [1, 2] and use automated planning to implement the configuration process. As a matter of fact, with automated planning we can easily obtain several alternative configurations for the same course or to verify the consistency of sets of learning components.

*contact author: Carla Limongelli limongel@dia.uniroma3.it
We show how the configuration of a course can be defined in terms of a planning problem, how the specification of learning components can be manipulated to feed the planning process and how the solution of such process can be interpreted as the specification of a configuration (or, the solutions can be used as alternative courses, each one responding to the intended learner’s need).

A short experiment, run on a class of 26 third-year undergraduate students, shows the applicability and the success of this approach.

2 The Logical Framework

We recall the basic definitions introduced in [1, 2] for course configuration.  
**Definition 1 (Knowledge item)** A knowledge item is an expression meaning the (sufficient) knowledge about a certain topic.

**Definition 2 (Component)** A component $C$ specifies a teaching contents which may be included in a course about a given notion/aspect/technique/teaching-aim. The component also specifies which knowledge is required to understand the teaching contents of the component and which knowledge is gained by going through the teaching content.

- **id** (component identifier);
- **rk** (required knowledge) The knowledge required to take the component, expressed as a set of knowledge items to be possessed before taking the component;
- **tc** (teaching contents) Teaching resources used in the component (hypertext, hypermedia, guided exercises, plain exercises for self evaluation, plain exercises to be evaluated by the tutor, final tests for self evaluation, final tests to be evaluated by the tutor);
- **ak** (acquired knowledge) The set of knowledge items which are gained by studying the component.

Once the starting knowledge of a given student has been defined (e.g. through an initial test), together with the target knowledge that should be reached about the topic, a course can be configured for the student, i.e. especially adapted for his/her case.

A course tailored to the peculiar needs of a given student is a configured course, where all the components needed to satisfy the target knowledge requirements are sequenced according to their prerequisites.

3 Course Specification = Planning Problem

The generation of a course can be seen as a planning problem. A planning problem is characterized by:

- **the description of the initial state:** in this case the initial state is represented by all the knowledge the student already has in the specific field: the Starting Knowledge;
the description of the executable actions and their effects: each action represents a didactic module (with the specific Required Knowledge, and the effects represent the knew knowledge acquired once that the module has been studied: the Acquired Knowledge;  
the description of the goal: it is represented by all the knowledge that the student has to acquire after the entire course: the Target Knowledge.

As a particular kind of planning problem, the generation of a course shows the following characteristics:
- everything is known: the environment is accessible;
- the environment is deterministic: actions effects are known;
- the environment is static: during the plan generation the environment does not change.

Under the above conditions, the solution found by a planner guarantees the goal reachability, starting from the description of the initial state and applying some of the actions. Moreover, in our course configuration problem, the plan comes out to be complete and consistent.

In a complete plan the goal $G$ and the actions preconditions are supported by the initial state and by one or more actions in the plan. Notice that in our case we never loose knowledge: fluents can only become true and they do no longer change.

4 The System

The core of our system uses the planner Blackbox [3]. Blackbox is a planning system based on the “planning as satisfiability” approach: it translates planning problems, written in PDDL, into boolean satisfiability ones; then it tries to solve them by means of different powerful SAT engines. It is one of the most popular planners and it is available under different platforms. These characteristics made us to choose Blackbox as a planner to implement course configuration.

In general a Didactic Course is constituted by two (pddl) files [5], the problem and the domain specification. The problem specification contains: the student’s Starting Knowledge, and the Target Knowledge of the course (see Fig. 1).

```lisp
(define (problem PROBLEM-NAME)
  (:domain DOMAIN-NAME)
  (:init STARTING-KNOWLEDGE)
  (:goal TARGET-KNOWLEDGE))
```

Figure 1: The general structure of the problem definition file.

The domain file represents all the modules in the course i.e. a list of components, represented by different actions. This file is unique for the course (see Fig. 2).
The planner takes these two input files and produces a plan that represents a set of didactic modules that the student has to learn in order to reach its target knowledge.

5 Experiments

We have run a short experiment on a class of 26 third-year undergraduate students, regularly attending the course of Algorithms and Data Structures.

Our aim was to test if and how the personalization is effective on the the students’ learning process and satisfaction. Components have been defined to cover a 3-hours section of the face-to-face course (an introduction to graphs). Two groups of students have been selected: the first group was ready to study that section on line; the second was the reference group, attending face-to-face lectures. At the beginning and at the end of the section the students answered two short tests: the first one aimed at assessing their starting knowledge and at configuring each course; the second one aimed at evaluating their final knowledge on introduction to graphs.

In appendix we show an example of a possible plan generated by Blackbox. Figure 4 shows the problem specification obtained from an initial test, figure 5 shows a snapshot of the domain, i.e. the specification of all the available components, and 6 shows the configuration generated by the planner.

The results of the initial and final assessments are reported in figure 3, that shows the gap between the initial (light blue) and final (blue) outcomes for the on-line students, confirming the effectiveness of the on-line course. In appendix we show also the questionnaire delivered to the students at the end of the course to evaluate...
the perceived effectiveness of the method. The final average results scored by the two tested groups is 8.60/10 for the reference group vs. 8.06/10 for the on-line group. Both the perceived quality of the course and the results of the assessment tests show that the method is effective as much as the face-to-face teaching.

![Comparisons between starting and final knowledge](image)

Figure 3: Comparison between the initial (light blue) and final (blue) tests, on-line group.

**References**


6 Appendix

(define (problem sample)
   (:domain introduzione-ai-grafi)
   (:objects k - knowledge)
   (:init (and (grado k) (componente-connessa k)
                (ciclo k) (intro-rappresentazione-grafo k)))
   (:goal (and (dfs k) (bfs k)))
)

Figure 4: Problem description: the student has basic notions about graphs and their
representation and should learn depth-first and breadth-first searches.

(define (domain introduzione-ai-grafi)
   (:requirements :strips :typing :equality)
   (:types knowledge)
   (:predicates
    (grado ?k - knowledge)
    (componente-connessa ?k - knowledge)
    (ciclo ?k - knowledge)
    (rappresentazione-grafo ?k - knowledge)
    (bfs ?k - knowledge)
    (dfs ?k - knowledge)
    ...
    )
   ...
   (:action id10
    :parameters (?k - knowledge)
    :precondition (rappresentazione-grafo ?k)
    :effect (lista-di-adiacenza ?k)
   )
   (:action id11
    :parameters (?k - knowledge)
    :precondition (lista-di-adiacenza ?k)
    :effect (matrice-di-adiacenza ?k)
   )
   (:action id12
    :parameters (?k - knowledge)
    :precondition (and (matrice-di-adiacenza ?k)
                       (lista-di-adiacenza ?k))
    :effect (complessit ?k)
   )
   (:action id13
    :parameters (?k - knowledge)
    :precondition (complessit ?k)
    :effect (richiami-complessit ?k)
   )
   (:action id14
    :parameters (?k - knowledge)
    :precondition (and (complessit ?k)
                       (richiami-complessit ?k))
    :effect (intro-ricerca ?k)
   )
   (:action id15
    :parameters (?k - knowledge)
    :precondition (intro-ricerca ?k)
    :effect (bfs ?k)
   )
   (:action id16
    :parameters (?k - knowledge)
    :precondition (bfs ?k)
    :effect (dfs ?k)
   ))

Figure 5: Snapshot of the domain describing the course components to the planner.

1 (id10 k) // Adjacency lists
2 (id11 k) // Graph representation: Adjacency matrices
3 (id12 k) // Graph representation: Representation complexity
4 (id13 k) // References on asymptotic complexity
5 (id14 k) // Search algorithms: introduction
6 (id15 k) // Breadth first search
7 (id16 k) // Depth first search

Figure 6: A plan generated form the previous problem and the course domain.
Figure 7: Final questionnaire (more frequent answers are highlighted).
Our experiments show that despite the complexity involved with the simultaneous consideration of both functional and non-functional properties our configuration technique is scalable. @inproceedings{Soltani2012AutomatedPF, title={Automated planning for feature model configuration based on functional and non-functional requirements}, author={Samaneh Soltani and M. Asadi and D. Ga[vs]evi{c} and M. Hatala and E. Bagheri}, booktitle={SPLC '12}, year={2012} }. Same works for automation testing. You canâ€™t automate everything, so you should be smart about your priorities if you are focusing on high ROI. Tweet this Share on Facebook. The most important thing at this stage is to define the scope of testing. When choosing a test automation framework, you should consider the technologies used in your project, the associated costs, the available talent, and your requirements. #3 Choose the right testing tools. From my own experience, Iâ€™d recommend building a custom test framework based on a free open-source tool (e.g. Selenium). For instance, Selenium is only a library for working with DOM elements via browser API; alone, itâ€™s not enough to run automated tests. AI planning, and specifically model-based planning, can be explained as a problem-solving method where the software developer describes (models) a problem, rather than codes the algorithm to solve the problem - which is radically different from how the conventional software development is practically-always done today. Not having to invent and code the algorithm has obvious benefits: developer productivity goes to extremes, you can write software with humanly-impossible complexity of algorithms, any tasks that require combining actions into meaningful chains can now be automated. Poodle introduces a pair of Python functions called xschedule and schedule that implement an automated planning mechanism, and a new base object Object View Automated Planning Research Papers on Academia.edu for free. The instrument first light occurred in the first semester of 2001. The results confirmed the possibility of using the adopted fibers and construction techniques for the SIFUS. We present the features of the instrument, some examples of the scientific data obtained, and the status of the commissioning, calibration and automation plans. The efficiency of this IFU was determined to be 53% during telescope commissioning tests. Save to Library. Home Conferences SPLC Proceedings SPLC '12 Automated planning for feature model configuration based on functional and non-functional requirements. research-article. Automated planning for feature model configuration based on functional and non-functional requirements. Share on. Authors We also provide tooling support to facilitate the use of our framework. Our experiments show that despite the complexity involved with the simultaneous consideration of both functional and non-functional properties our configuration technique is scalable. References. K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson.