Automation in Construction 86 (2018) 190–198

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

A tool-supported framework for work planning on construction sites based on constraint programming

Azahara Camacho[a], Pablo C. Cañizares[b], Sonia Estévez[b], Manuel Núñez[b,*]

[a] Department of Aerospace & Defense, Carbures Engineering, Cádiz, Spain
[b] Dept. Sistemas Informáticos y Computación, Facultad de Informática, Universidad Complutense de Madrid, Spain

ARTICLE INFO

Keywords:
Construction sites planning
Constraint programming
Formal specification and analysis

ABSTRACT

This paper presents a framework taking advantage of the capabilities of current constraint solvers to plan the work on construction sites. It combines different constraints under a common framework to facilitate the definition of temporal relations between different tasks and provides a user-friendly web interface, which facilitates the planning of construction sites. Even though the framework uses complex mathematical models, the users do not need to know the underlying theoretical framework. Another important feature of the framework is, in contrast to usual *static* planning, that solutions can be dynamically adapted to take into account delays happening during the actual construction process. In order to show the applicability of the methodology, the paperspaper shows how a real construction project can be planned by using the framework.

1. Introduction

In order to construct any structure, even the smallest ones, it is usual to have a *model* of the building that it is going to be developed.[1] Therefore, planning is an important step in construction and it is fundamental in order to ensure the successful management of construction sites. In fact, the awareness of the importance of *formalizing* the planning of different tasks in construction sites is increasing [2]. In this line, it is very important to plan how the different teams and machinery will be deployed in the construction site. It is necessary to take into account both the availability of these resources and the time constraints associated with the different tasks. There are tasks that should be sequentially performed (e.g. terrain movement should be completed before the construction of a concrete surface bed is started) while others can be performed in parallel and a correct planning will minimize the amount of idle resources. The temporal planning of a construction site is reflected in a *graph*. In this line, Gantt diagrams are specially popular. These diagrams show, in a very intuitive way, causal relations between different tasks and phases of the site. Unfortunately, these graphs are usually made by hand, based on the experience of the architect(s) planning the site. Even though experienced planners were in charge of this phase, it would be more accurate, and reduce the number of errors, to automatically compute solutions based on a formal design [3].

The framework strongly facilitates the generation of *static* plans but it is possible to go one step forward. In addition, the use of constraint programming [4] strongly simplifies the management of unexpected situations after the construction has started in the site. Specifically, it is not necessary to manually recompute the impact of delays on the tasks that depend on its conclusion.

The main goal of the framework is to provide good solutions, while minimizing human intervention, in the computation of optimal temporal assignment of resources to tasks. As a first step, it facilitates the work of the planner by providing tools where (s)he can easily describe the temporal constraints between activities. Afterwards, the framework will automatically compute solutions, taking into account different priorities (e.g. minimize the construction time). A description of the framework will be given in Section 3 of the paper; its basic scheme is:

1. There is an HTML5-based interface where the user can introduce the data of the project and visualize the current state of the provided information. In particular, the user will provide preferences on the solution (e.g. minimize the project time).
2. The visualization layer will be connected to a Java engine. This engine will provide the data of the project to a constraint solver.
3. Constraints are grouped into different sets, according to their characteristics (e.g. precedence, optimization). The constraint solver is launched.
4. The constraint solver sends the result back to the resolution engine

* Corresponding author.
E-mail addresses: azahara.camacho@carbures.com (A. Camacho), pablocc@ucm.es (P.C. Cañizares), soesteve@ucm.es (S. Estévez), manuelnu@ucm.es (M. Núñez).

[1] While this is common sense in Architecture, it is not the case in other disciplines such as Computer Science where researchers still advocate the use of formal modelingmodelling before initiating the *construction* of a software system [1].

https://doi.org/10.1016/j.autcon.2017.11.008
Received 13 March 2017; Received in revised form 30 October 2017; Accepted 10 November 2017

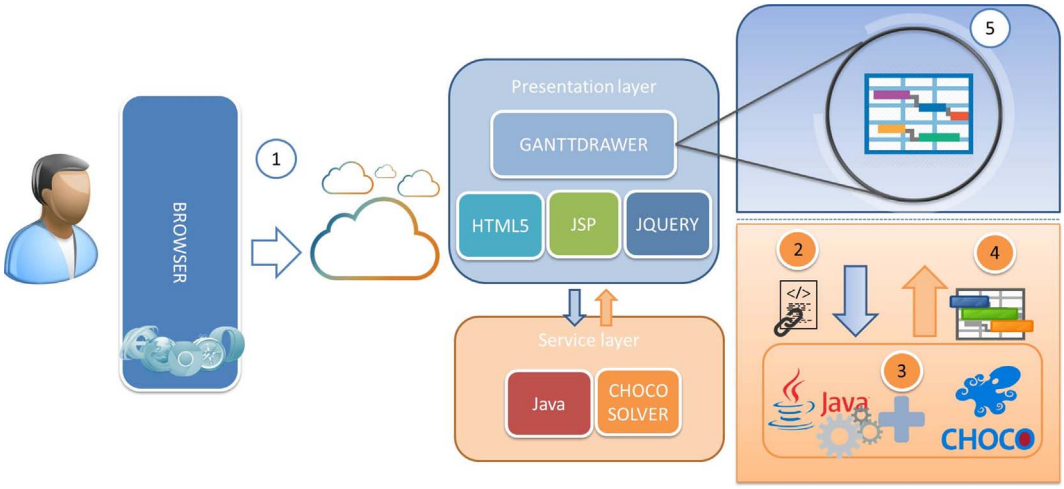

Fig. 1. Basic scheme of the approach.

and the Gantt diagram is presented.

5. The user can access the temporal evolution of the project and can input new information (e.g. one resource is not available when it was planned to be, meteorological conditions produce a delay).

A graphical description of the previous process is shown in Fig. 1. It is worth noting that only the first and last steps require human intervention. If modifications to the original plan arise (the last step of the scheme presented above), then the interaction of the user with the system is simple and no previous knowledge about the implementation details is necessary. It is worth to emphasize the importance of the last step. Usually, planners prepare a Gantt diagram. Since this is a *static* artifact, any unplanned changes cannot be easily reflected. In the best case, the planner can manually redo the original diagram by delaying all the tasks accordingly. However, this is not necessarily the best solution because it is very likely that resources can be assigned in an alternative way and/or a different ordering of tasks can be found. Several examples of these situations will be presented.

In order to show the relevance and usefulness of the framework, the planning of a real project is presented: the construction of a water treatment works. All the phases that conform the project are considered, from the removal of trees and vegetation in all areas to the installation of pipes. It is also necessary to take into account the needed tools, from trucks to the software used to design blueprints.

During this process the benefits of the approach are shown, in particular, it is possible to optimize the use of available resources.

Concerning related work, the use of formalisms to facilitate the planning of construction sites is not new. In fact, many formalisms have been adapted from Computer Science (CS) and Artificial Intelligence (AI), most notably, multi-agent systems [5–8]. In this line, it is worth mentioning the role of Building Information Modelling (BIM).

During the last years, and mainly due to the increase of the computational power of even the cheapest systems, the BIM framework [9–11] is becoming very popular and there exist many successful applications [12]. Finally, CS and AI have also influenced planning activities by contributing with novel mechanisms and methodologies based on semantics and ontologies [13–16]. The framework strongly depends on the use of constraints. Since this field does not belong to the main topics of the journal, Section 2 includes the basic rudiments concerning how constraints can be defined, how constraint programming works, and different types of constraints and constraint solvers.

The rest of the paper is structured as follows. Section 2 reviews the main concepts related to constraints. Since this is a very big field, and far from the interest of most readers of the journal, this part of the paper will concentrate on the use of constraints in this specific setting. Section 3 describes the main inputs of the chain of tools, the basic functioning of these tools, and the methodology. Section 4 explains the constraint solving resolution method. Section 5 presents a case study showing how the methodology and tools can be applied to plan a real construction project. Finally, Section 6 presents the conclusions and some lines for future work.

2. A brief introduction to constraints

Constraint programming (CP in short) has its beginnings in the 1970s [17,18]. The big advancement of the field happens during the 1980s when classical unification of logic programming is replaced by constraint solving to create constraint logic programming [19]. Afterwards, other programming languages have been extended to include constraints. The use of constraints has clear advantages in terms of conciseness and neat formulation.

In fact, many programming languages, such as Prolog, C++, Java and Python, have constraints libraries. This work uses the Java constraint library Choco [20].

Constraints are defined over some specific domains, being *finite domains* the most used ones. In this setting, constraints are defined over finite sets. A *constraint solver* is a decision procedure that checks whether a constraint, or a set of constraints, can be satisfied. A *finite domain solver* is a constraint solver where variables range over finite sets of values, usually finite subsets of the set of integer numbers. Finite domain solvers rely on a systematic exploration of the search space until either a solution is found, or it is shown that the problem does not have a solution. In order to reduce the search space, these solvers are combined with inconsistencies filtering techniques that narrow variable domains.

CP is a broad research area and has many applications in real life. For instance, some systems with applications in industry are the constraint logic programming system CHIP [21] and the IBM ILOG CPLEX Optimizer [22,23].

Constraint satisfaction models are often benchmarked on hard problems. For example, CSPLib [24] is a library, which is independent of any particular constraint solver, containing different problems organized by subject area. In addition, there exists a world-wide competition of CP solvers: the MiniZinc challenge [25].

It is impossible to cover all the current CP applications. Therefore, three classic problems have been chosen to show the essence of this paradigm. The first problem is “The *n* queens”. This problem belong to a family of problems, including sudoku, solitaires, and others games and puzzles, that can be solved by using an elegant constraint satisfaction mechanism. The second problem is “The car sequencing”, a problem of