ConfigNow: a knowledge based approach to configuration software

Kumar Abhinav Hanne Vlaeminck Joost Vennekens Johan Wittocx
Marc Denecker

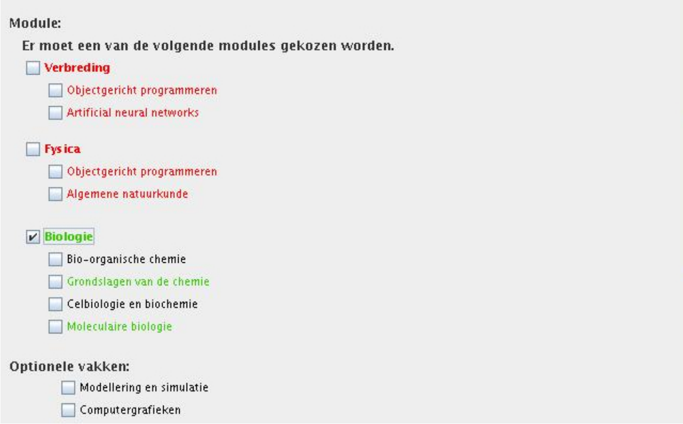
Katholieke Universiteit Leuven

Abstract

ConfigNow is package that allows users to create configuration software following a knowledge based approach. The user can write the constraints on the domain down in a separate file, using (extensions of) first order logic as knowledge representation language. The ConfigNow package then provides an interface to perform a number of tasks that one typically wants to solve in configuration software, and link this to the user interface.

1 Motivation

The idea of a knowledge based approach to software development is an old idea within AI. In this approach, the user writes down the knowledge he has about the domain in a *logical theory*, using a knowledge representation language. Several tasks can then be solved by the use of different *inference methods*. Especially software where the difficulties in implementing and maintaining come from the complex domain knowledge, can greatly benefit from following a knowledge based approach. Configuration software clearly satisfies these criteria, and [2] proposes a knowledge based approach based on FO($\cdot$), a rich knowledge representation language and an extension of classical logic. Furthermore they identify a number of tasks that one typically wants to solve in configuration software and show that they can be efficiently solved by applying the correct logical inference methods.



Module:

Er moet een van de volgende modules gekozen worden.

□ Verbreding
- □ Objectgericht programmeren
- □ Artificial neural networks

□ Fysica
- □ Objectgericht programmeren
- □ Algemene natuurkunde

☑ Biologie
- □ Bio-organische chemie
- □ Grondslagen van de chemie
- □ Celbiologie en biochemie
- □ Moleculaire biologie

Optionele vakken:
- □ Modellering en simulatie
- □ Computergrafieken

Figure 1: Course selection configuration software

Figure 1 shows a part of a configuration tool that helps students to complete their course selection. On the screen you can find a number of compulsory courses, optional courses, and a number of modules and their courses. An example of domain knowledge in this application is the constraint that 'exactly one module has to be selected'. An example of a task that we need to solve is to color the other modules red if a user selects a certain module, to indicate that according to the rules that module can no longer be selected.

The screenshot in figure 1 comes from a proof of concept [2]. In this proof of concept the domain knowledge was indeed written down in a separate theory, and the tasks solved by two reasoning systems, the IDP-system [1], and the Approx-tool [3]. However in this proof of concept, the links between the user interface, the theory file and the reasoning systems were all ad-hoc implemented.

2 ConfigNow

The ConfigNow package provides an interface to connect the user interface with the theory file, the database that contains the partial selection the user has made up to now, and the reasoning systems. When the user wants to implement a new piece of configuration software, he has to do the following. First of all he the user has to define the vocabulary that will be used and write down the relevant domain knowledge in a *theory file*. Next, he has to design a user interface. Currently, this UI is required to be a *Java Swing* user interface. As a last step, the user then needs to connect the components of the user interface to the logical vocabulary and reasoning systems, through a *configuration file*. Thus, the ConfigNow package allows for very a fast implementation of configuration software. We have used the ConfigNow package to implement a number of configuration tools, e.g. a Bike Configuration Tool (as one can find on e.g.[4]), and we were able to implement this tool in a couple of hours, using the ConfigNow package.

We believe that for software with a lot of domain knowledge, the knowledge based approach is the right approach. The ConfigNow package is a first step in an attempt to create a complete package that developers can use to implement knowledge based software. In the future, we aim at including additional reasoning systems in the package to add extra functionality, and allow for more complex configurations files, where we can e.g. specify that a part of screen can only be generated after a certain condition has been satisfied.

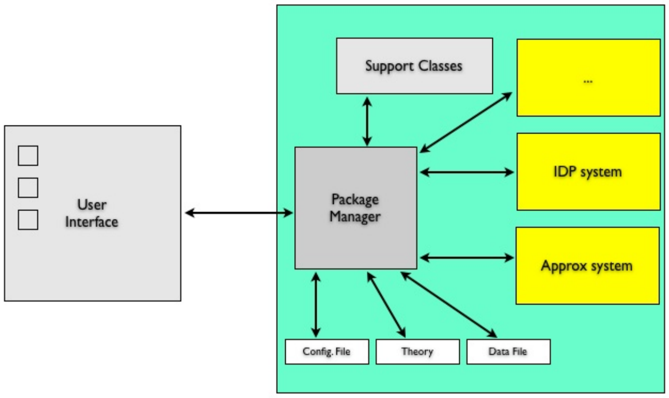


Figure 2: ConfigNow architecture

References

[1] Maarten Mariën, Johan Wittocx, and Marc Denecker. The IDP framework for declarative problem solving. In *Search and Logic: Answer Set Programming and SAT*, pages 19–34, 2006.

[2] Hanne Vlaeminck, Joost Vennekens, and Marc Denecker. A logical framework for configuration software. In *PPDP*, pages 141–148, 2009.

[3] Johan Wittocx, Maarten Mariën, and Marc Denecker. Approximate reasoning in first-order logic theories. In *KR*, pages 103–112, 2008.

[4] www.configit.com.