Infraworld, a Multi-agent Based Framework to Assist in Civil Infrastructure Collaborative Design (Demonstration)

Jaume Domínguez Faus
Centre for 3D GeoInformation
Ålborg Universitet
9220 Ålborg, Denmark
jaume@land.aau.dk

Francisco Grimaldo
Departament d'Informàtica
Universitat de València
Av. de la Univesitat s/n
Burjassot (València), Spain 46100
francisco.grimaldo@uv.es

ABSTRACT

Infraworld is an experimental framework for Computer Aided Engineering (CAE) systems which is designed for distributed design. The framework is based on Multi-agent systems that allow engineers to synchronize their work by keeping track of their changes and facilitating the detection and management of semantic conflicts that arise when different actors are working in parallel. Conflicts are detected according of each engineers semantics which are defined by using OWL ontologies and SWRL rules. When they are detected, the framework allows solving them by negotiating possible alternatives. Then the alternatives are evaluated by expressing preferences and the picked alternative, being the one that maximizes the global welfare, is applied in all the models in the distributed environment. The system is completed with a machine learning module that allows the agents to suggest similar solutions to future conflicts with similar semantic context.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems

Keywords

Distributed decision making, Semantic conflict detection

1. INTRODUCTION

Civil Infrastructure design has evolved from its very initial steps of paper design to Computer Aided Engineering tools that help in its complex tasks. These tools include sets of predefined features to compute concrete situations such as, e.g, the distribution of forces in a structure. However, these works are always carried out by sets of teams that specialize in some profile of the multidisciplinar Civil Infrastructure project. Unfortunatelly, most of the existing tools are geared to individual workplaces. Although there are some efforts to make this work more distributed like file repositories or the most advanced BIM [1] servers there is low support for handling conflict situations caused when several separated works have to be put together to align all the project designs.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

Unfortunately, despite previous works like [2] the process of alignment still a prominent manual one. The civil engineers meet regularly to align their works and in the best case they can do it using 3D representation of the models that are navigated and analyzed by the authors in order to find conflicts. Thus, there is a need for this process to be automated. When some error is not detected in design time and the project is already in the construction phase, it is very costly to fix it. And this situation is still happening today with the corresponding project overheads and delays. Studies estimate them at around 5-10% of the total budget in average [1]. To fulfill this gap, we present the Infraworld framework. Infraworld allows the definition of the semantics of a model on a per-engineering-profile basis. The semantics are defined by sets of OWL [3] ontologies from which the base knowledge is built, and the conflicts are detected by using SWRL rules. They are used by the JADE agents that control the evolution of the project in each workstation and allow, in front of a conflict, to negotiate how to solve it with the other stakeholders of the project.

2. THE INFRAWORLD FRAMEWORK

The Infraworld framework is composed of three main logical pieces: 1) a reasoning engine that can be used by Validator agents, 2) the collaboration module that defines the negotiation protocol that is carried out when solving conflicts, and 3) a learning module that, when the negotiation ends, captures the solution applied and the context of the conflict in order to infer solutions for future similar conflicts.

2.1 Reasoning Engine

Unlike the usual systems in which the conflict detection is based on pure geometric overlapping of the objects, also called Features, in the model, Infraworld framework extends the concept of conflict to the semantics. To do that, there is a Core Ontology that defines the concepts of Feature, Attribute, Geometry, and Relationship. A Feature represents an entity of the world and it is composed of the Attributes that parameterize it, the Geometry that gives its physical shape in the world and its Relationships with other Features in the model. The second level of abstraction is the FeatureCatalog ontology which gives the meaning of what Feature represents. For instance there is a Building concept in this ontology that when applied to a Feature defines it as a building.

Beyond these ontologies, each engineering profile provides with their own. This approach allows a feature to be treated

differently depending on the point of view. For instance, a sewerage conduction might only be an obstacle for an engineer that is planing a gas supply conduction and only needs to ensure it does not overlap his designs. However, the engineer designing the sewerage has to ensure that there are no other conductions underneath. In other words, the sewer profile needs to take care of other specific problems than just regular geometry overlaps.

To complete the knowledge, SWRL rules are supported. They are also provided by each profile and they are meant for detecting the conflicts. These rules consist of an antecedent and a consequent. The antecedent is evaluated against the model and when it resolves to true, then the consequent is said to be also true. For instance a conflict like the one explained above could be captured with a SWRL rule as follows:

$$Conduction(?c1) \wedge Sewerage(?c2) \wedge isBelow(?c1, ?c2) \rightarrow PositionNotAllowedConflict(?c1, ?c2)$$

This rule would mark features that match the condition expressed in the antecedent (first row) as a PositionNotAllowedConflict.

2.2 Collaboration Module

The collaboration module defines a Multiagent society (see Figure 1) composed of Validators, Negotiators and Coordinators. As the engineers work in parallel, changes are performed to the model. These changes are monitorized by the Validator agent that executes the Reasoning Engine when changes to the model are detected. When conflicts are detected as a result of the execution of the reasoner, the engineers have the possibility to solve the conflict by means of negotiation. The negotiation is based on MARA (Multi Agent Resource Allocation) as a general mechanism to make socially acceptable decisions and follows a ContractNet-like protocol that is executed when a Validator agent wants to solve a conflict. It consists of two round negotiation. In the first round, the Coordinator agent notifies all the Negotiators that a conflict has been detected and asks for alternatives to solve it. Then, Negotiators record the alternatives provided by the engineers and send them back to the Coordinator agent. The Coordinator agent collects all the alternatives and send them again, in a second round, to the Negotiators so that their engineers can provide with preferences. The engineers express their preferences by giving a score ranging from -5 to 5 to each alternative and the Negotiators send them to the Coordinator. The Coordinator picks the alternative that maximizes the global welfare as the solution and notifies this decision to the Negotiators. This solution is finally applied to all the models in the distributed environment.

2.3 Learning Module

The third module is aimed to learn from the engineers experience and behavior. After a conflict has been solved in a negotiation the context of the conflict, i.e. the Features and their Attributes that were in conflict, as well as the related Features this Feature may have by means of its Relationships, are registered for future processing. If the same conflict occurs in the future, the Validator agent might find coincidences in the history and suggest the past solution to help the engineers to find a solution for it.

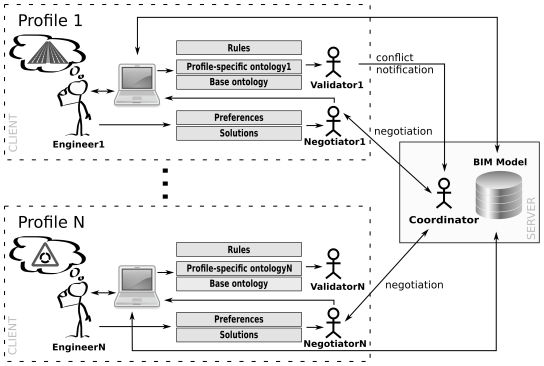

Figure 1: Overview of the system

3. EXPERIMENTS AND RESULTS

We applied our framework to two use cases to test it. The Reasoning Engine showed to be adequate to detect project-specific semantic problems. The Collaboration Module allowed to perform the negotiation. Finally, the Learning Module could suggest solutions for repeating problems.

- **Urban Development Use Case** consisting of a model with 4107 ontology instances covering the development of the city of Drammen in Norway. In this use case two profiles (a traffic engineer and a builder) were designing the model. The goal was to ensure that the road network was not exceeded by the population living in the buildings being planned.
- **Power Plant Electricity Installation Use Case** with 4592 ontology instances in which two engineering profiles in charge of the foundations and the wiring structure had to solve conflicts regarding the bolts connecting both elements that were misplaced. Since this conflict was repeating, the Learning Module helped solving it by automatically suggesting solutions.

4. ACKNOWLEDGEMENTS

This work was supported by Norwegian Research Council, Industrial PhD scheme case no: 195940/I40 through Vianova Systems AS, Norway; the Spanish MICINN, Consolider Programme and Plan E funds, European Commission FEDER and Universitat de València funds, under Grants CSD2006-00046, TIN2009-14475-C04-04, UV-INV-AE11 - 40990.62

5. REFERENCES

[1] C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Handbook, a guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors.* John Wiley & Sons, Inc. New Jersey, 2008.

[2] F. Peña-Mora and C.-Y. Wang. Computer-supported collaborative negotiation methodology. *Journal of Computing in Civil Engineering*, pages 64–81, April 1998.

[3] W3C OWL Working Group. OWL 2 web ontology language document overview. Technical report, W3C, Oct. 2009.