Distributed Cognition Learning in Collaborative Civil Engineering Projects Management

Jaume Domínguez Faus and Francisco Grimaldo

Abstract Due to the diversity and complexity of its projects, the Civil Engineering domain has historically encompassed very heterogeneous disciplines. From the beginning, any Civil Infrastructure project is systematically divided into smaller subprojects in order to reduce or isolate the overall complexity. However, as a parallel design work, these subdesigns may experience divergences which often lead to design conflicts when they are merged back to the global design. If a high-quality design is desired, these conflicts need to be detected and solved. We present a Multi-agent system able to manage these design conflicts by detecting them, by assisting the engineers in the negotiation of solutions, and finally by learning how to solve future similar problems. The advantage of the system is that what is learned is not one individual's knowledge but the project's global distributed cognition.

Key words: Multi-agent systems, Civil Engineering Projects, Semantic validation, Negotiation, Machine learning

1 Introduction and Related Work

Civil Engineering projects are a collaborative work. The integral development of any civil infrastructure demands expertise in many different disciplines. Since developing a project as a whole is an overwhelming job, the works to be done are systematically classified and assigned to teams of experts holding the required skills for each of them. Thus, for instance, structure engineers take responsibility for design-

Jaume Domínguez Faus
Centre for 3D GeoInformation, Aalborg University. 9220 Aalborg, Denmark, Tel. +45 9940 3699, e-mail: jaume@land.aau.dk

Francisco Grimaldo
Departament d'Informàtica, Universitat de València, Av. Universitat, s/n, 46100, Burjassot, Spain, e-mail: francisco.grimaldo@uv.es

ing the skeleton of the infrastructure; sewerage engineers design the management of waste waters and drainage; signaling is done by traffic experts, etc.

When working on the project, each team develops its part separately and periodically they all align their work by merging all their "subdesigns". Then, engineers evaluate the progress in order to find and solve the problems the project contains. Nowadays, this task is performed manually by interpreting the project's documentations and, when possible, by navigating a 3D computer model created by combining all the subdesigns. However, this process can fail in detecting all the issues since documentation may become difficult to be analyzed and some information may be lost when constructing the global model. As a result, if a problem remains undetected during the design time, it causes potentially large loses in resources afterwards. Some authors have estimated the average costs caused by these undetected problems in 5-10% of the total project budget [4]. Thus, given the size of Civil Engineering projects, there is significant potential for improvement.

To improve interoperability between the different stakeholders, the industry has traditionally defined new data exchange formats and it has extended the existing ones . Whereas probably the most common format is DWG from AutoDesk, which is still the *de facto* standard, there are others, e.g. CityGML or IFC [7], that also enjoy a relative success. The latter even include extension mechanisms for "user-defined" data structures as a way to consider particular needs. However, they can hardly guarantee that those user-defined structures are understood by outsiders, given that consumer applications need to implement mechanisms to support them. This shows that the interoperability has not been completely reached yet. On the other hand, Building Information Model (BIM) servers [4] offer a higher degree of interoperability by storing CAD models together with management spreadsheets and other documents in a centralized way and, thus, easing the project management.

Even though these efforts facilitate the control of the project, the detection of problems and their resolution still require human interaction. Traditional approaches have shown difficultes in tackling the challenges derived from distributed collaborative work but literature suggests that Multi-agent Systems (MAS) can better equipped for facing these kind of problems. MAS have been already used in Civil Infrastructure research in different situations. An example of controlling construction equipment is the system described by Zhang, Hammad, and Bahnassi where sets of cranes synchronize their movements to transport materials from one location to another in the construction area without crashing [12]. It is also possible to find examples of MAS that focus on the distinct phases a typical project consists of. Thus, Denk and Schnellenbach describe an agent-based tendering procedure that covers the initial phase when the project is published and interested companies bid for it [13]; Udeaja and Tah [14] focus on the construction material supply chain that is managed collaboratively. Within the Construction phase the main focus has been put on the formalization of expert knowledge and the negotiation mechanisms to solve unexpected situations. Peña-Mora and Wang [10] presented a Game Theory approach to solve various known conflict situations between the Architect/Designer/Constructor settings when each agent competes for reducing the impact of unexpected situations in the Construction area in its own interest. More recently Shen et al. studied the applicability of cognitive maps MAS in collaborative-competitive working in construction projects where these maps are used to parameterize the agent's beliefs within the MAS negotiation [16]. We propose using MAS to assist in the detection of problems in the Design phase, where the definition of a problem depends on how an expert sees the world according to her perspective. Problems are solved attending to the project's global benefit by means of negotiating alternatives to it. Finally, the MAS uses humans' decisions as a way to capture the project's *distributed congition* that is exploited afterwards to train the system so that it can suggest its own alternatives from the knowledge it has gained.

This work aims at developing an intelligent system devoted to aid engineers in managing design conflicts in civil infrastructure projects. The proposed system is able to detect these conflicts, to assist engineers in the negotiation of solutions, and finally to learn how to solve future similar problems. It follows a distributed multi-agent approach incorporating well-known artificial intelligence techniques in order to tackle the problem in a flexible and extensible way, thus providing the necessary level of abstraction to be applied to different projects within this domain. Hence, this ad hoc intelligent system contributes to increase strategic intelligence (i.e., knowledge management + business intelligence + competitive intelligence) of companies whose projects join teams with heterogeneous expertise working collaboratively.

Section 2 introduces the main components of the system: Semantic Conflict Detection, Distributed Conflict Solving, and the Learning Subsystem. A deep description of the Semantic Conflict Detection and Distributed Conflict Solving components is out of the scope of this paper since it can be found in [5]. However, they are are briefly introduced in sections 2.1 and 2.2 for the sake of readability. The Learning Subsystem is described in section 2.3. Finally, in section 3 we describe a use case in which the system was applied in order to illustrate its usefulness.

2 The Multi-agent System

Our system follows a distributed architecture approach allowing the engineers of different expertises to design, through their client interfaces, a common Building Information Model (BIM) that is stored at the server (see Figure 1). Notice that the meaning (the semantics) of an object in the model, and in particular what it implies, varies from who is looking at it. For instance, while a gas conduction could be seen as a mere tube-shaped obstacle by an electrician engineer, it means much more for the expert that is designing it. The former may only need to ensure that her designs are not putting anything in the location already used by the gas conduction to avoid object overlapping. On the other hand, the latter will have other concerns like, e.g., whether the material of the pipe is compatible, in terms of safety, with the distance of such an electric installation. Hence, it is not only the object's shape what is important, but also what the object *means* (i.e. its semantics) and, therefore, what it implies. We refer to a semantic conflict when a problem of this type is detected. In other words, a situation that may be correct from a single engineer's point of view but where different interests collide when it is considered globally. In our system, we define the semantics that each engineer cares about by means of OWL [1] ontologies and SWRL [9] rules. They are used by each engineer's Validator agent for detecting and inferring problematic situations. When a conflict is detected, a solution for it can be negotiated. The Validator agent that detected it notifies a Coordinator agent residing at the BIM server. The Coordinator then conducts the negotiation with all the stakeholders' Negotiators which, upon an agreement on the solution for the conflict, apply it and register it in the Learning Subsystem for further analysis in order to suggest solutions for future similar situations. Our agents ecosystem uses JADE [2] as the MAS engine. We define a Profile as the set of the human expert engineer, her semantic knowledge base, her Validator and Negotiator agents, and her Learning Subsystem.

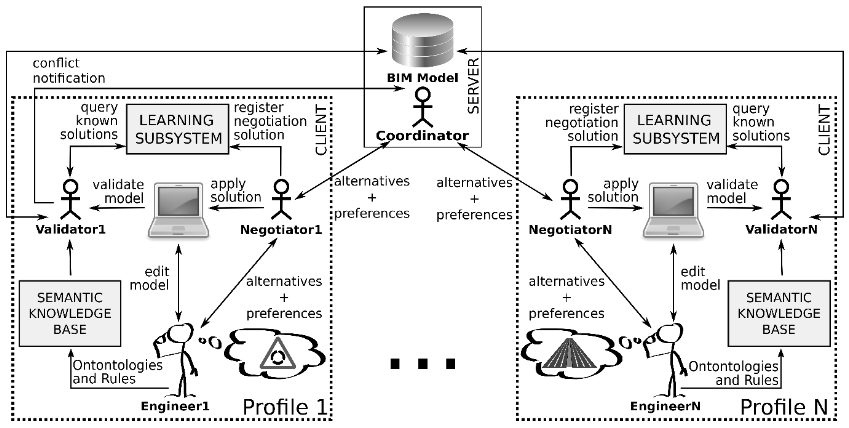

Fig. 1 Overview of the system

2.1 Semantic Conflict Detection

Ontologies are the formalization of concepts from which knowledge is built and that, in contrast to the classical attribute/value approach, provide semantic abstraction to the model. Our system builds the semantics of the BIM model by means of a layered structure of ontologies. A Base Ontology defining the core concepts needed for any civil infrastructure project is provided by default to each Profile and, on top of it, engineers can stack more Profile-specific ontologies to achieve the level of abstraction desired. The core concepts of the Base Ontology are: 1) `Feature`, the basic object of a model; 2) `Geometry`, the shape of a `Feature`; 3) `hasGeometry` and its inverse `isGeometryOf`, used to set the relationship between a `Feature` and a `Geometry`; 4) `Attribute`, defining one of the properties of a `Feature`; 5) `hasAttribute` and its inverse `isAttributeOf`, used to set the relationship between a `Feature` and an `Attribute`; 6) `hasRelationship` and its inverse `isRelationshipOf`, defining a generic relationship between two `Features`; 7) `Problem`, to capture individual errors in the project; and 8) `Conflict`, which is used to mark a subclass of `Problem` that has to be negotiated in order to be solved. In turn, Profile-specific ontologies extend these concepts with other ones that are of interest for particular fields of expertise such as: a `Building` or a `Road` for a building designer. Complementing the ontologies, engineers also provide SWRL rules to allow advanced reasoning. SWRL rules infer more concepts by specifying an antecedent that, when it turns out to be true, it implies that what is expressed in the consequent is also true. For instance, a `RoadExhaustedConflict` can be detected by a road designer when the building designer places a new building in a parcel where the road connecting to that parcel cannot hold the new population brought by the building. As engineers work in parallel, changes are performed to the model. These changes are monitorized by the Validator agents, that executes the Reasoning Engine when changes to the model are detected. Thus, both the set of ontologies and the rules defined by the engineers are used by the Validator agents to analize the model from each Profile's perspective in order to find `Conflicts`. More details about the semantic conflict detection can be found at [5].

2.2 *Conflict Solving Protocol*

When conflicts are detected, engineers have the possibility to solve them by means of negotiation. The negotiation follows a Multi-Agent Resource Allocation [3] approach, as a general mechanism that assists the engineers in evaluating the possible alternative solutions and in making socially acceptable decisions. Thus, they are required to express their preferences about the solutions, which are then used to select those that maximize the welfare of the group. As the allocation procedure, we use a ContractNet-like protocol involving two rounds. In the first round, once a conflict has been detected by a Validator agent, it is notified to the Coordinator agent. Then, the Coordinator informs all the Negotiators about the conflict and asks for alternatives to solve it through a Call For Solutions. In turn, Negotiators record the alternatives provided by the engineers and send them back to the Coordinator, who is in charge of collecting all the proposals. The solutions consists of operations that, when applied to the model, avoid the conflict. Invalid or repeated solutions are discarded and the set of remaining solutions is distributed again, thus starting the second round of the protocol. In this phase, engineers express their preferences by giving a score ranging from 0 (lowest) to 10 (highest) to each alternative and their corresponding Negotiators send this information to the Coordinator. Then, the Coordinator performs a winner determination process that leads to the selection of the alternative that maximizes the global welfare. The Nash social welfare function is used, as it ensures that the chosen solution is the most preferred one while also balancing the utility level among the negotiators. Finally, the solution is broadcasted to all the engineers and it is applied to the BIM model. As previously stated, more details about this distributed conflict solving mechanism can be found at [5].

2.3 Learning Subsystem

It seems possible to predict what the result of a negotiation will be if we have a model that is semantically rich enough to contain sufficient information. We used Machine Learning (ML) techniques to capture the distributed cognition that results in the selection of a feasible solution for a given type of conflict. From the advent of Nash's formalization of the bargaining problem [8], the negotiation can be methodically approached. Thus, the negotiation can be seen as a black box that captures all the subtleties (the context, the points of view, etc.) in which we input the problem and a solution is obtained in the output. But it requires that the engineers express their utility functions which are especially elusive to be defined a priori in dynamic and complex environments. That is the case of Civil Infrastructure projects where not even the set of parameters to such functions are easily known. The negotiation protocol we use solves this problem by asking each engineer's preferences on the possible alternatives, resulting in a solution. This way, we can capture the global behavior pattern while the complexity of the system remains at reasonable levels. We can keep a record of the inputs to the problem (i.e. the conflict and the context in which it occurs) and the solution (consisting of operations applied to the model), given by the engineers and use it as a training set from which the agents in the MAS can learn. As a result, after several rounds of negotiation to conflicts showing similarities, the agent can suggest the solution to a newly detected conflict.

In our system, when the Validator agent detects a conflict, it queries the Learning Subsystem (LS) for known solutions to the conflict. The query consists of the conflict and the `Feature`s involved in it and the relationships among them, that is, the conflict's context. If possible solutions are known, they are presented to the user so she can choose and directly apply the most suitable. If there are no known solutions or the ones proposed by the Validator are not satisfactory then the negotiation described in section 2.2 occurs. After this negotiation, all engineers will agree on a new solution. The Negotiators then apply it to the model and register it into the LS.

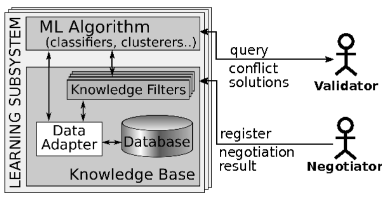

Fig. 2 Learning subsystem

Reich [11] carried out an extensive review where he analyzed experiments with ML in Civil Infrastructure concluding that there is no ML technique that solves all the problems but instead some work better than others depending on the problem. We designed our learning system in a way that it is easily extendable with standard techniques (e.g, ID3, C4.5, K-Means, etc. [15]) as well as with experimental ones.

As already mentioned, a solution consists of operations applied to the model that change its state from a, say, “conflicted status” to a “valid status”. Thus, the solutions directly rely on what the supported operations are, and also what and how the parameters for those operations are. In Civil Infrastructure, spatial operations are predominant. Consider, for instance, the operation “move object A 3 meters in direction D”. A closer look at that operation shows that it takes five parameters: the object, the distance the object is going to be moved and the direction (which in 3D needs three coordinates). The spatial nature of the operations forces some preprocessing to be done before it becomes a useful solution for the learning. Imagine that this solution was produced as a result of the conflict “the object A is overlapping the object B”. If we store the solution as it is, in the eventual event that new similar “object F is overlapping the object G” conflict is detected, upon a request to the LS for a solution to this conflict, it will respond with the “move object A 3 meters in direction D” which is obviously wrong. A way to overcome this problem is to store the solutions in a symbolic way. In other words, instead of “move object A 3 meters towards direction D” we would express it as “move *TheObjectThatIsOverlapping AwayFromTheObjectItOverlapsWith*”. And make the corresponding substitutions in each different case of the conflict. The Knowledge Base of the LS is equipped with the Knowledge Filters (see Figure 2) which is an extendable set of filters meant for endowing the solutions with a level of abstraction as necessary. The filters are applied forward and in reverse order for storing and querying respectively.

On the other hand, Machine Learning (ML) algorithms are very sensitive to the way data is stored and to its *noise*. In order to accommodate the data to the algorithm in use, the Data Adapter (DA) is included. The prediction of a solution the algorithm does is based on the conflict and its context. However, while the conflict belongs to a finite set of conflict types (otherwise it could not have been detected), the context where it occurs varies. The DA is in charge of transforming the data in the database to fit the algorithm’s needs while no relevant information is lost.

3 Use case

To test the proposed approach, we have applied our system to a real project consisting in the design of the electricity installation of a power plant carried out by “Vianova Plan og Trafikk AS” in Norway. We prefer using a real use case as there is to our knowledge no standard and suitable benchmark to test our approach against. Even though some CAD benchmarks have been proposed [6], they focus on the size of the dataset as well as on the computation time. Instead, we aim at finding conflicts and learning how to solve them without the need to write a routine on purpose.

In this use case, the design is done by two profiles, one designing the foundation of the installation (Foundation Profile) and the other designing the structure holding the electric cables (Structure Profile). The use case consists of 4592 ontology entities and the reasoning time takes 50 seconds. During validation, conflicts can be detected such as the bolts of a foundation that is used to fix a structure not fitting in the holes the Structure Profile designed for them due to some measurement misunderstanding. This kind of conflict is detected 80 times, one per each bolt in the foundation, through the following SWRL rule:

$$\begin{aligned}
&Bolt(?b) \wedge Hole(?h) \wedge \\
&isGeometryOf(?b, ?bg) \wedge isGeometryOf(?h, ?hg) \wedge \\
&isClosest(?bg, ?hg) \wedge distance(?bg, ?hg, ?d) \wedge \\
&isGreaterThan(?d, 0) \rightarrow BoltDoesntFit(?b, ?h)
\end{aligned}$$

This rule would tag the feature b as a conflict of type `BoltDoesntFit` if, when transversing the model, b is a `Bolt` feature and h is a `Hole` feature such that they are the closest Hole/Bolt pair in the model (given by their geometries) and the distance between them is greather than 0.

The two profiles negotiate to solve the first conflict. In the alternative proposal round, the Foundation profile suggests to move the left leg of the two-legged structure so it fits with the bolts. The leg is represented by a `StructurePart` feature which contains an attributed called `Leg` with value “Right”. To make the leg fit in the foundation bolts it needs to move 3 centimeters to the left (X axis) and 1 cm downards (Y axis), i.e. it needs to move to the distance $d1 = (-0.03, -0.01, 0)$ (alternative $A1$). The Structure profile suggest to move the bolt to the distance $d2 = (0.03, 0.01, 0)$ (alternative $A2$). In the second round, the profiles express their preferences. Structure profile gives a value of 1, and 5 to $A1$ and $A2$ respectively because, say, she knows that the structure parts are already delivered by the supplier and changing them would be very costly while moving the bolts does not seem to be a big deal. The Foundation profile, in turn, gives a value of 7, and 4 because she does not know about the structure parts situation but would prefer others to change while she admits that moving the bolts is a relatively small effort. As a result, $A2$ is selected as a solution. Similar negotiations follow for the rest of conflicts with same results with the only difference that when the bolt is supposed to fit the “Left” leg of the structure, it is moved in the opposite direction, i.e. $d1 = (-0.03, -0.01, 0)$. The results were stored in the LS using only one simple Knowledge Filter that replaces the name of the features involved by symbolizers (#@#, in this case) so they can be more generally applied. Table 1 shows the (simplified) database created from this. We only show the necessary columns on the table due to space restrictions. In this case, the actual table is composed of 15 columns storing other relationships the bolts have and attributes that are not relevant since they take the same values in all the records. We used an ID3 decision tree as the ML algorithm which automatically detects that what defines the direction where the bolt is to be moved is the Leg attribute of the “StructurePart” feature.

Table 1 Simplified database for this conflict. Each row represents a conflict and its negotiated solution (CTF=ConflictedFeatureType, CWF=ConflictedWithFeature, CWFRF=CWF-RelatedFeature, “#@#“ is a symbol refering to the feature that is in conflict that is substituted and restored by the Knowledge Filter)

CTF	CWFType	CWFRF1	...	CWFRF1AttrLeg	...	Solution
Bolt	Hole	StructurePart		Right		MoveFeatureCmd(#@#,-0.03,-0.01,0.0)
Bolt	Hole	StructurePart		Left		MoveFeatureCmd(#@#,0.03,0.01,0.0)

Thus, once the system collects enough information, Validator agents can find the learned solutions from the LS that can be applied without further negotiations. Notice, however, that the amount of negotiations required could be reduced by having an extra Knowledge Filter substituting the distance vector within the parameters of the solution by a symbol as explained in section 2.3.

4 Discussion

As computers became more powerful, they allowed to execute bigger and more complex programs. The industry producing software for Civil Engineering creates software packages with newer and more powerful features. Today’s engineers can perform more complex tasks with these advanced tools. However, these tools consist of pre-established algorithms with the only possibility of parameterizing them as a means to adapt them to one situation or to another. Furthermore, the tools are meant to be executed once the user decides to invoke them. In other words, they are not designed to make decisions but to execute them.

It is assumed that the only way to improve is creating new algorithms. In our opinion, this approach is, somehow, preventing the creation of software that solves conflicts in distributed design settings. Because of the overwhelming amount of factors converging in a conflict, it seems not possible to write algorithms covering all the issues. Even worse, the necessary information that influences a distributed decision may simply not be available for the computer system. In fact, most of the information is still residing only in the engineers head. It is simply too difficult to input every detail into a system which, in turn, needs to be prepared to store it.

The proposed system takes another approach. By following a semantic knowledge-driven approach, the system is able to detect conflicts and to propose automated solutions, which is of great utility for solving any managerial problem and decisional scenarios. While humans make decisions, the system tracks them and captures the distributed cognition that allows making the right decisions in the particular context the project occurs in. Thus, the system capitalizes on the conflict resolution made by humans. What is a problem in the begining turns out to be a valuable asset that is exploited by the system to learn and to improve. Then, engineers are no longer limited to what software packages allow since it is the software what evolves and adapts to their needs.

The future work will focus on testing the system in more use cases to deeper study how the system performance compares to humans with regard to the quality of the decisions made.

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, as well as European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04-04. It was also partly supported by Development and Planning Department of the Ålborg University (AAU), the Vice-rectorate for Research of the Universitat de València (UV) under grant UV-INV-AE11-40990.62. Authors also want to thank Dr. Erik Kjems from the AAU for his help without which this paper would not be possible.

References

1. J. Bao, D. Calvanese, et al. *OWL 2 Web Ontology Language Document Overview*. W3C, 2009.
2. F. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-agent Systems with JADE*. John Wiley & Sons Ltd, 2007.
3. Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaitre, N. Maudet, J. Padget, et al. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.
4. 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.
5. J. D. Faus and F. Grimaldo. Multiagent system for detecting and solving design-time conflicts in civil infrastructure. *Advances in Soft Computing*, (in press), 2012.
6. S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani. Developing an engineering shape benchmark for CAD models. *Computer-Aided Design*, 38(9):939–953, Sept. 2006.
7. T. H. Kolbe, G. Gröger, and L. Plümer. Citygml: Interoperable access to 3d city models. *Geo-information for Disaster Management*, pages 883–899, 2005.
8. J. F. Nash. The bargaining problem. *Econometrica*, pages 155–162, 1950.
9. M. O'Connor, H. Knublauch, B. G. S.W. Tu, M. Dean, W. Grosso, and M. Musen. Supporting rule system interoperability on the semantic web with swrl. *4th International Semantic Web Conference (ISWC), Galway, Ireland, Springer Verlag*, pages 974–986, 2005.
10. F. Peña-Mora and C.-Y. Wang. Computer-supported collaborative negotiation methodology. *Journal of Computing in Civil Engineering*, pages 64–81, April 1998.
11. Y. Reich. Machine learning techniques for civil engineering problems. *Microcomputers in Civil Engineering*, 1997.
12. Z. Ren and C. Anumba. Multi-agent systems in construction–state of the art and prospects. *Automation in Construction*, 13:421–434, 2004.
13. M. Schnellenbach and H. Denk. An agent-based virtual marketplace for aec-bidding. *Proceedings of the 9th International EG-ICE Workshop Advances in Intelligent Computing in Engineering, Darmstadt, Germany*, pages 40–48, 2002.
14. C. Udeaja and J. Tah. Agent-based material supply chain integration in construction. *Perspectives on Innovation in Architecture, Engineering and Construction, CICE, Loughborough University*, pages 377–388, 2001.
15. I. H. Witten, E. Frank, and M. A. Hall. *Data Mining Practical Machine Learning Tools and Techniques*. Elsevier, 2011.
16. X. Xue, Y. Ji, L. Li, and Q. Shen. Cognition driven framework for improving collaborative working in construction projects: Negotiation perspective. *Journal of Business Economics and Management*, 2010.