18th International Conference on the Application of Computer
Science and Mathematics in Architecture and Civil Engineering
K. Gürlebeck and C. Könke (eds.)
Weimar, Germany, 07–09 July 2009

COUPLING PATTERNS IN CIVIL ENGINEERING APPLICATIONS

Toni Fröbel*, **Berthold Firmenich, Christian Koch**

**Bauhaus-Universität Weimar*
Graduiertenkolleg 1462, Berkaer Strasse 9, 99423 Weimar
E-mail: Toni.Froebel@uni-weimar.de

Keywords: coupling, design pattern, building model, coupling quality, coupling classification

Abstract. *This paper addresses the problem of coupling partial models in civil engineering. According to the state-of-the-art, applications and partial models are formulated by the object-oriented method. Although this method solves basic communication problems like subclass coupling directly it was found that many relevant coupling problems remain to be solved. Therefore, it is necessary to analyse and classify the relevant coupling types in building modelling. Coupling in computer science refers to the relationship between modules and their mutual interaction and can be divided into different coupling types. The coupling types differ on the degree by which the coupled modules rely upon each other. This is exemplified by a general reference example from civil engineering. A uniform formulation of coupling patterns is described analogously to design patterns, which are a common methodology in software engineering. Design patterns are templates for describing a general reusable solution to a commonly occurring problem. A template is independent of the programming language and the operating system. These coupling patterns are selected according to the specific problems of building modelling. A specific meta-model for coupling problems in civil engineering is introduced. In our meta-model a specific coupling design is an instance of a general coupling pattern.*

1 INTRODUCTION

Buildings can be divided into various types and described by a huge number of parameters. Within the life cycle of a building, especially during design and construction phases, a lot of engineers with different points of view, proprietary applications and data formats are involved. The collaboration of all participating engineers is characterised by a high amount of communication. Due to these aspects, a homogeneous building model for all engineers to use is not feasible. The status quo of civil engineering is the segmentation of the complete model into partial models. The interdependencies of these partial models are not in the focus of available engineering solutions.

2 STATE OF THE ART

Due to the nature of the planning process in civil engineering a lot of engineers, using different proprietary applications and data formats, are involved. Therefore, this cooperation is distributed and based on the coupling of several applications. Usually, civil engineering applications are implemented utilizing the Object-Oriented Method (OOM). As is generally known, this method alone does not ensure the consistency of distributed planning information.

One approach for distributed cooperation is based on the integration via a common object model. The most famous model standard in civil engineering are the Industry Foundation Classes (IFC), which have been developed to describe, exchange and share building information [1]. It is generally believed that one building model tends to be too complex and therefore is not feasible. A different approach based on a common meta-model including partial models and coupling was proposed in [2]. Due to its very generic nature existing engineering applications cannot be integrated and reused directly. For this reason this approach is also considered not to be applicable in a practical engineering environment.

The planning environment is characterized by many engineers with different points of view. This leads to different and separate partial engineering models. However, these partial models have to be coupled in order to support technical dependencies. Based on the object-oriented method a generic version-oriented approach for distributed cooperation was introduced by [3]. In this approach the dependencies between partial models are described by versioned objects and bindings. Bindings are directed couplings between object versions without any specific semantics. However, the distributed cooperation between engineers requires the semantic coupling of applications implementing partial models. This topic has been covered in [4] for the coupling of CAD and FEM models.

In [5] a processing-oriented approach for the semantic coupling of applications is presented. The conclusions from this research work are that semantic couplings cannot be achieved by considering single objects only. Instead the scope of the whole engineering model has to be taken into account when cooperatively processing this model. For these reasons it is necessary to investigate the semantics and the processing logic of couplings which results in the development of coupling patterns.

According to computer science coupling refers to the relationship between modules and their mutual interaction. In [6] different types of coupling are presented. They differ in the degree by which the coupled modules rely upon each other. Furthermore, coupling quality formulations for traditional as well as for object-oriented criteria can be described via metrics [7, 8]. Besides many others the meta-model architecture [9] and design patterns [10] are important

methodologies in software engineering. The first allows for modelling on different abstraction levels. The second describes general reusable solutions to commonly occurring problems in terms of a template that is independent of the programming language and the operating system. The application of a design pattern results in one or more specific UML[1] diagrams [11].

Generic coupling models and quality metrics from computer science are considered to be the basis for civil engineering applications. Therefore, existing approaches have to be extended and adapted to coupling patterns in civil engineering. It is envisaged to formulate coupling patterns on the basis of the meta-model architecture and software design patterns.

3 SCENARIO FROM CIVIL ENGINEERING

The scenario is used as an example environment for testing and evaluating coupling patterns. We try to give a qualitative assessment for selected patterns.

3.1 Holistic Engineering Model

It is assumed that the simplified building in figure 1 has to be designed. The main basis for the coupling assessment is a holistic system consideration. The holistic system consists of four parts. The soil is described as a half space with bi-linear material behaviour. The circular foundation with a linear material behaviour is embedded centrically on the top of the soil. The superstructure, a circular and thin-walled pillar with a bi-linear material behaviour and a point mass at the top, is located at the center of the foundation. A dynamic wind load represents the fourth submodel.

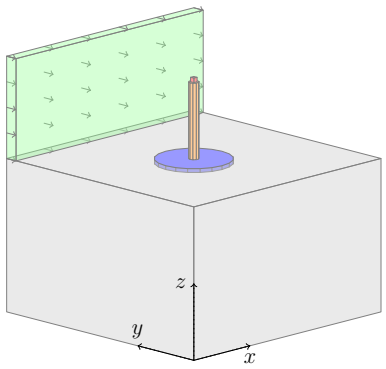


Figure 1: Holistic system consideration

3.2 System Parameters

As a basis for evaluating coupling patterns the system parameters (Figure 2) are to be identified. The displacement on the top of the pillar u^{top}, the displacement u^{con} and the rotation φ^{con} at the top of the foundation as well as the displacement u^{bot} and the rotation φ^{bot} at the bottom of the foundation are of particular interest.

[1] Unified Modelling Language

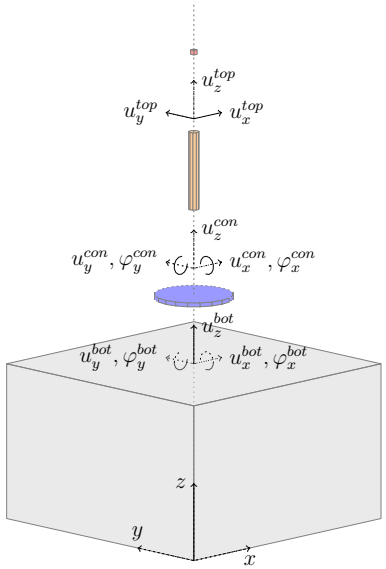


Figure 2: System parameters

Further system parameters are the deformation energy for the single parts such as the pillar, the soil and the foundation as well as the internal and the external energy of the entire system.

3.3 Case Studies

For a more practical consideration, the holistic system has to be subdivided into partial models. As an advantage each partial model can now be created and modified separately in independent applications. This is a basic prerequisite for a distributed engineering environment that enables the synchronous cooperation. Furthermore, it means that the quality of each submodel may differ, e.g. according to material, loads and dimension (2D vs. 3D). Figure 3 illustrates the four partial models.

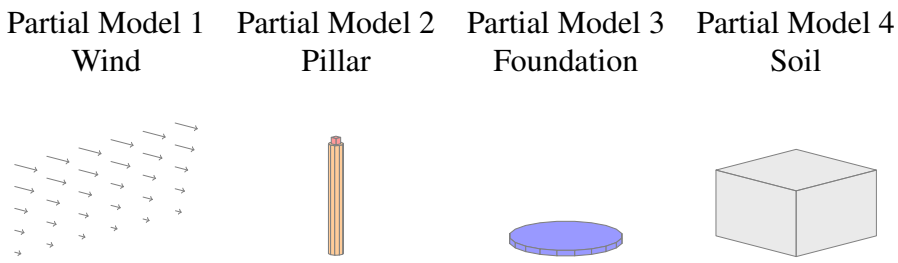


Figure 3: Partial models of the scenario example

Due to their interdependencies the partial models have to be semantically coupled. The sum of all submodels and their couplings then again leads to a holistic system consideration which is necessary to solve the overall design problem. From an engineering point of view, coupling strategies between the partials models are selected and described below.

- The coupling 1-2 (wind ↔ pillar) is formulated as a wind-load function.

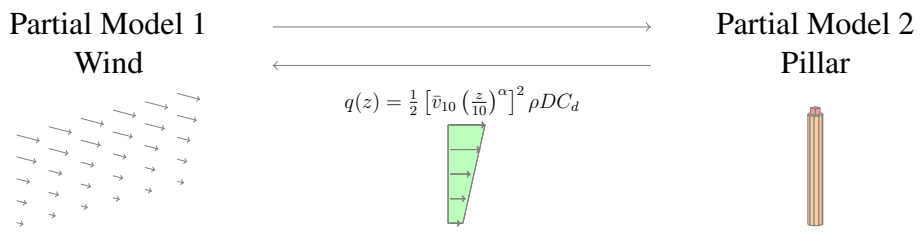


Figure 4: The coupling between the partial models 1 and 2

- The coupling 2-3 (pillar ↔ fondation) is described via forces and fixed-end moments at the bottom of the pillar.

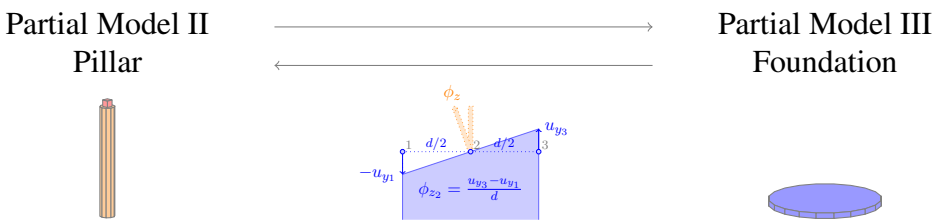


Figure 5: The coupling between the partial models 2 and 3

- The coupling 3-4 (foundation ↔ soil) is based on a set of contact springs.

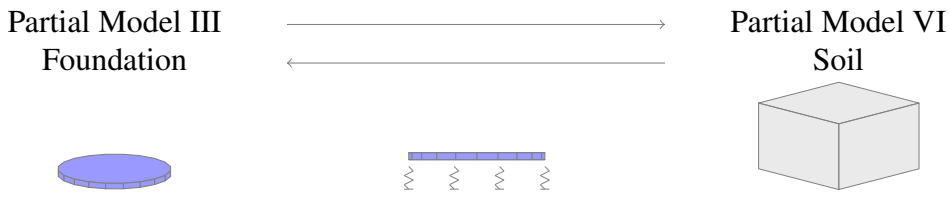


Figure 6: The coupling between the partial models 3 and 4

The quality of a practical system consideration is highly influenced by the applied coupling strategy.

4 MODELLING IN CIVIL ENGINEERING

A core task in civil engineering is the description of real buildings and their environment. Due to their complexity, buildings are never completely describable. Therefore, civil engineering models only represent a more or less accurate description of the reality. The level of detail usually depends on the problem under consideration.

4.1 Civil Engineering Models

The basic idea of modelling is the simplification of complex phenomena. This is achieved by the reduction of the complexity utilizing an abstraction procedure that finally results in a model. Typical engineering models are mathematical, physical and numerical models. Mathematical models describe significant properties of phenomena by systems of equations. A special kind of mathematical models are physical models. In this case the system of equations describes a physical behaviour. If analytical solutions are not available or not feasible then numerical models (e.g. Finite Element Model) are applied. Nowadays computers usually support engineers when designing buildings. As a requirement, engineering models have to be implemented as computer models.

4.2 Computer Models

In contrast to engineering models, computer models are primarily based on programming paradigms, for instance the Object Oriented Method (OOM). According to the State of the Art the meta-model architecture methodology allows for modelling on different abstraction levels, which is helpful when facing the complex problems of civil engineering.

4.2.1 Meta-Model Architecture

A well known and established concept for multilayered-based modelling is given by the common meta-model architecture. The general idea is to describe a model from a higher model layer. This is formulated by a mapping function f that maps a layer M_i to the next higher layer M_{i+1}.

$$\forall_{m \in M_i} \exists_{n \in M_{i+1}} : (m, n) \in f, \quad f : M_i \rightarrow M_{i+1} \tag{1}$$

The basic four-layered meta-model architecture is introduced in [12] and is illustrated in figure 7. Layer M_0 contains the data of a specific model instance. Layer M_1 represents the model that describes the specific model instance M_0. Layer M_2 in turn defines a meta-model for describing the model of M_1 and so forth. Based on each single layer M_i any number of instances M_{i-1} can be created.

The meta-model architecture is applied to the OOM (figure 8a). In this case, the data of a specific model instance is encapsulated in objects (M_0). Objects are described by classes (M_1). The layer M_1 is a real implementation based on an object-oriented programming language. Classes and their relationships are described independently from any programming language. This is achieved by the graphical-based unified modelling language UML (M_2). The top layer M_3 contains the meta object facility (MOF) specification [12] for describing the UML. Enhancing the general object-oriented meta-model architecture by building semantics results in a building meta-model architecture as illustrated in figure 8b. Here, the data of a specific building is stored in M_0. The software that is used by engineers implements a specific building model (M_1) in terms of classes. When designing an engineering software application the UML is utilized in

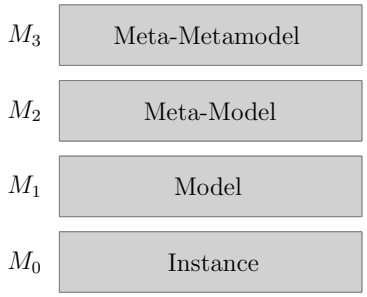
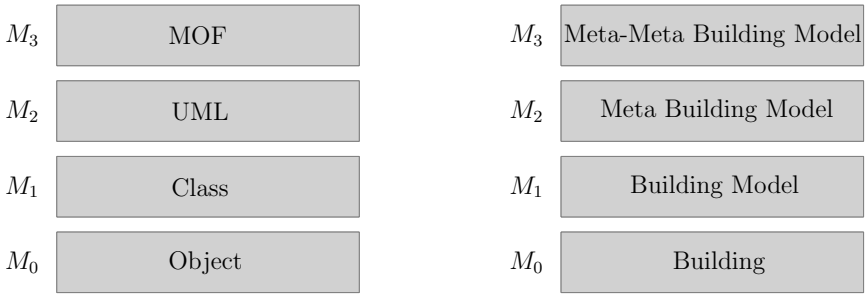


Figure 7: four-layered meta-model architecture

order to formulate a meta building model (M_2) that in turn describes the building model. The layer M_3 defines meta building models (M_2) in the most generic form. Further meta layers are not feasible for practical purposes. As an analogy, a poet (M_0) is written in a natural language (M_1). This language is defined by its grammar (M_2), that in turn is described by the language itself.



(a) OOM (b) Buildings

Figure 8: Meta-model architectures: OOM and Buildings

Engineering applications and building models are usually object-oriented implementations. For this reason it makes sense to follow the OOM when describing coupling strategies within the scope of computer models. Furthermore, it is to be investigated how the meta-model architecture can be applied to couplings in civil engineering. The subsequent section presents coupling patterns in civil engineering applications based on the OOM and the meta-model architecture.

4.3 Model quality

The overall model quality Q depends on both the qualities of the engineering model Q_e and the computer model Q_c:

$$Q = Q_e \cdot Q_c \ , \quad Q \in [0, 1] \tag{2}$$

A quality of 0 indicates the worst quality whereas 1 describes an optimal solution. Both Q_e and Q_c can be interpreted as a product of single partial qualities Q_i

$$Q = \prod_{i=1}^{n} Q_i \tag{3}$$

Examples for Q_i are

- Knowledge of the civil engineer as a software user
- Knowledge of the software engineer
- Quality of the communication network
- Quality of the engineering methods
- Quality of the software methods
- ...

It should be noted that the complete quantitative assessment of these qualities is not in the focus of this paper.

5 COUPLING PATTERN

Coupling in computer science means the linking of software modules with different purposes and complexities. A module can be understood as a data element, a method, a component or an application. Data elements are small entities and parts of a specific data base. Methods instead are more complex and often knowledge-based, describe a behaviour and can be used for mathematical calculations. Methods can use both data elements and methods. Components are software parts for a common purpose and consist of other components, methods and data elements. In order to be able to describe interdependencies, software modules have to be coupled among each other. Due to the different module complexity a lot of coupling strategies are available. Furthermore, various coupling strategies can be combined. In general, software modules are implemented in different programming languages, depend on operating systems and use different data bases and media. This leads to a vast quantity of coupling strategies.

5.1 Meta-model Coupling Architecture

The application of the introduced object-oriented meta-model architecture as a basis for describing coupling strategies leads to the architecture shown in Figure 9. This meta-model coupling architecture enables the formulation of coupling strategies on different levels of abstraction: coupling pattern, coupling design, coupling model and coupling instance.

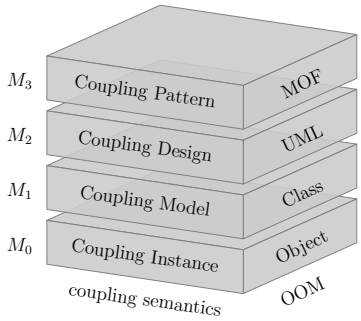


Figure 9: Four-layered meta-model coupling architecture

In the subsequent table each coupling layer of the meta-model architecture is described.

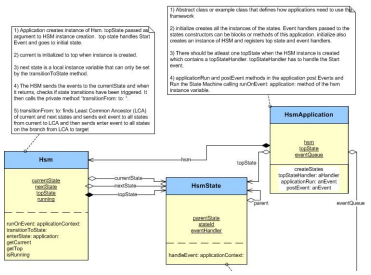
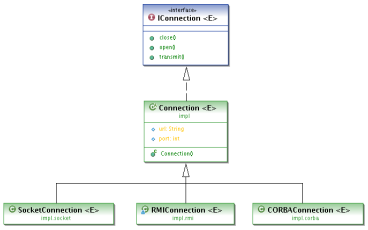
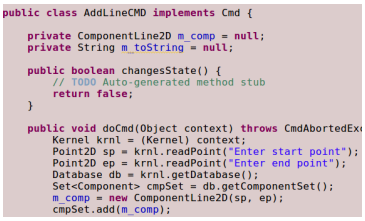
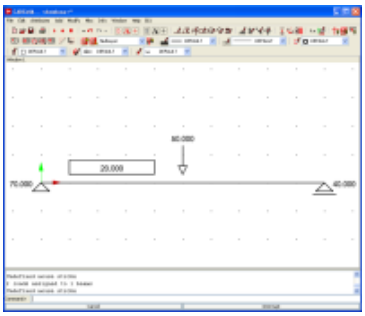
M_3		Layer M_3 contains the coupling patterns analogously to the well-known design patterns. This includes several formal and informal descriptions like graphical diagrams, textual descriptions and images. The patterns are structured as follows: • Pattern name • Problem description • Design proposals • Quality metrics • Examples
M_2		Layer M_2 contains the object oriented coupling design in a graphical notation called UML. Frequently used diagram types are: • Class diagrams • Sequence diagrams • Use case diagrams
M_1		In OOM this layer contains the coupling model in the form of classes and their methods and attributes.
M_0		Layer M_0 contains the specific coupling instances at runtime.

Table 1: Layer descriptions of the meta-model coupling architecture

5.2 Coupling Patterns

Coupling patterns are located in the fourth and most abstract layer M_3. Here, a coupling description is most generic, graphical-based and independent from any programming language. Its classification can be based on former formulations of coupling levels proposed in [6]. Since these formulations were originally designed for couplings in stand-alone procedural-based software, they have to be adapted to object-oriented and independent partial models in engineering. General coupling patterns for civil engineering applications are described subsequently. Each pattern contains its basic coupling strategy as well as its general quality criterion. Furthermore, the relevance of each pattern in the context of the scenario introduced in section 3 is presented. Qualitative assessment is given for selected patterns.

5.2.1 No Coupling

Modules are not coupled if they are completely independent and do not communicate at all among each other. It is well known that missing couplings lead to numerous problems in real civil engineering projects.

5.2.2 Message Coupling

Message coupling is characterized by the exchange of unstructured messages between modules. When the connection is established the coupled modules are almost independent of each other. Because unstructured messages cannot be interpreted by software, engineers have to evaluate them on the basis of their knowledge and experience. Figure 10 illustrates four engineers A, B, C and D cooperating via message coupling using a communication network.

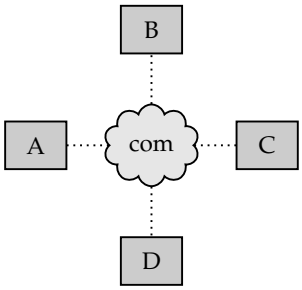


Figure 10: Graphical notation of the message coupling pattern

In the design and construction process message coupling is used to exchange ideas, suggestions and notifications about state changes. The concept of instant messaging is based on message coupling. The messages are plain text and the Internet serves as the communication platform. Since the messages are understood by the engineers the concept of instant messaging supports their cooperation.

Message coupling is exemplified in the context of the scenario example and illustrated in figure 11. The coupling between partial model 3 (foundation) and partial model 4 (soil) is based on contact springs. In order to calculate the foundation displacements the stiffness of the springs, which model the soil, have to be known. Therefore, engineer A communicates the foundation loads (model 3) to engineer B via email, phone or instant messaging. Based on these loads engineer B is now able to determine the soil displacements (model 4) that in turn

describe the spring stiffness. After receiving the stiffness parameters engineer A computes the displacement of the foundation.

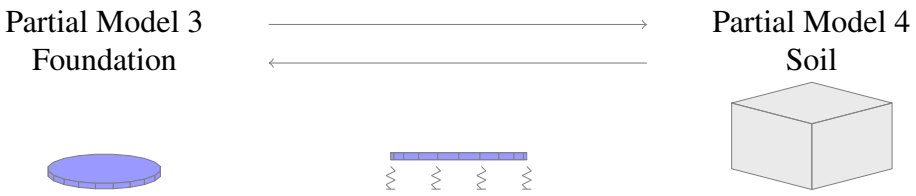


Figure 11: Message coupling between partial model 3 and 4

If the quality of the engineering model is 1 then the overall model quality depends to large degree on the network quality Q_N and the knowledge of the participating engineers, where

$$Q_N := \{0, 1\} \text{ where } \begin{cases} 0 & : \text{communication successful,} \\ 1 & : \text{communication failed.} \end{cases} \tag{4}$$

5.2.3 Data Coupling

Two modules are data-coupled if they communicate via input and output parameters. In contrast to message coupling all input parameters are interpreted by the called module. Typically, a result is returned to the calling module as output parameter. Figure 12 illustrates a data coupling between modules A and B where module B is called from module A.

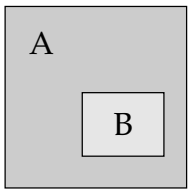


Figure 12: Graphical notation of the data coupling pattern

This type of coupling is most commonly used within software. As an example, a module A needs the square root of a value x. Therefore, it calls an external method B which provides this computation. The parameter x is passed to B and the result of the square root of x is returned to the calling module A. Hence, the module A is data-coupled with the method B.

Data coupling is exemplified in the context of the scenario example. The coupling between partial model 1 (wind) and partial model 2 (pillar) is based on an altitude-dependent wind load function $q(z)$ which is a result of partial model 1. Partial model 2 uses this wind load function and assigns the function values for a certain altitude as external loads on the pillar. The partial models 1 and 2 are data-coupled via a method representing the function $q(z)$.

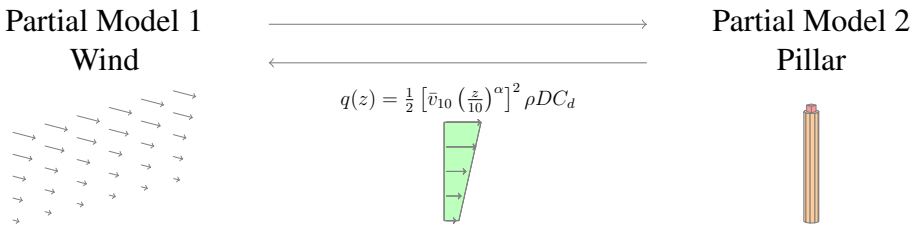


Figure 13: Data coupling between partial model 1 and 2

It is assumed that engineers select appropriate methods for the problem under consideration. The data coupling quality primarily depends on the qualities of the engineering method and the software implementation. Both qualities Q_i cannot be described as being 0 or 1, but they are located in between these two extreme values:

$$Q_i \in [0, 1] \tag{5}$$

5.2.4 Data Structure Coupling

Two modules are data-structure-coupled[2] if they exchange data through a common data structure. Figure 14 shows the typical situation where instances of model A are to be transferred to model B as instances of a third model F. Referring to the meta-model architecture both the model and the instance layer are shown. Typically, the data structure F does not originate from A or B but is a (de-facto) standard on its own. More often than not the exchanged data structure is not available in a machine-readable form. Instead, the standard specification is communicated via massive text documents describing the model in an informal way. As a rule, only a subset of the common data structure is used for a specific data exchange.

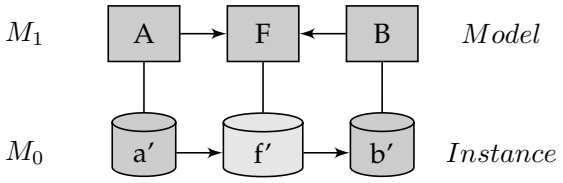


Figure 14: Graphical notation of the data structure coupling pattern

In the planning process many different engineers with incompatible engineering applications have to cooperate. Apparently, data-structure-coupling is very important in such a situation. As an example, a data exchange between partial models 2 and 3 of the scenario example is described and illustrated in figure 15. Firstly, the involved engineers must agree on a common data structure. Secondly, the instances of partial model 2 are to be exported as accurately as possible according to the exchange format. Thirdly, the instances are transmitted as a file. Fourthly, the information is imported as a partial model 3 instance. In our scenario example the most important exchange information is the reaction forces at the bottom of the pillar.

[2]In this paper the terms 'data structure' and 'model' are synonymous.

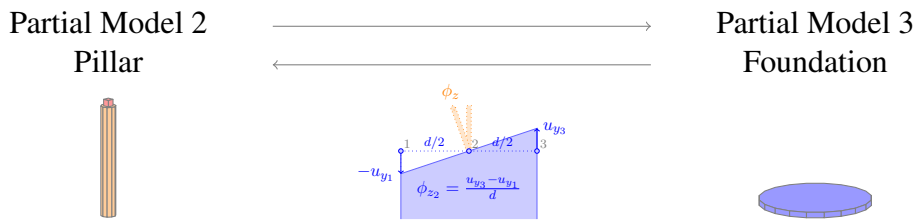


Figure 15: Data structure coupling between partial model 2 and 3

Figure 16 shows the incompatibility of the data structures involved. If A is the data structure of the source system and F is the exchange data structure then instances of type $A \cap F$ can be exchanged with a quality of 1.

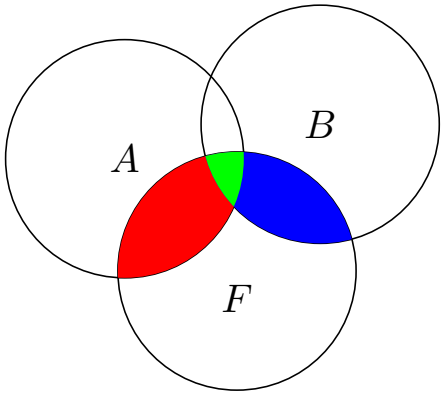


Figure 16: subset of data structures

If however the source partial model contains instances of $A \setminus F$ then this must not necessarily result in a quality of 0 for these instances. If the instances can be approximated by instances of model $A \cap F$ then even in these cases a very good quality close to 1 can be obtained.

When importing the instances into the destination application, then data structure B narrows the transferable model to $A \cap F \cap B$. Again, this must not necessarily mean a loss of quality. If most instances can be approximated by the model $A \cap F \cap B$ then a high coupling quality can be obtained.

5.2.5 Further coupling patterns

The remaining coupling patterns are described in less detail because they are of lesser relevance in the context of our example scenario:

Control coupling: Two modules are control-coupled if one module passes parameters that are used to control the internal logic of the other module. In contrast to all other introduced coupling patters control coupling allows for controlling the behaviour of another module. Figure 17 illustrates the control coupling pattern. The World Wide Web as a well-known

example for control coupling is considered. Different WWW browser instances A, A' and A'' communicate with a WWW server B that sends the requested web sites.

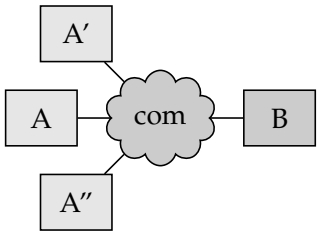


Figure 17: Graphical notation of the control coupling pattern

External coupling: Modules are external-coupled if they share an externally imposed data format, communication protocol or device interface. A typical application of external coupling is the USB interface on the hardware side and the plugin concept on the software side.

Common coupling: Modules are common-coupled if they refer to the same global module. A practical application of common coupling is the coupling of different engineering applications via a central database.

Content coupling: Two modules are content-coupled if one module embeds the content of the other module as an integral part. In other words, the embedding module cannot work correctly without the integrated one. Examples for content coupling are shared libraries for finite element analysis in the context of pre- and postprocessing procedures.

6 CONCLUSION

Civil engineering projects are processed by a huge number of engineers with different views of the common building. The state-of-the-art implies that buildings as a whole cannot be processed simultaneously by all involved engineers. Simultanous processing is only possible if the complete building is divided into partial model instances. Relevant research is primarily focused upon coupling in the form of generic bindings that must be interpreted by the engineers. Due to the high complexity of the coupled models in civil engineering this can be a very complex task. To reflect their interdependencies the partial building models have to be semantically coupled. This is true for both engineering models and computer models.

The paper introduces a set of coupling patterns for civil engineering applications. The coupling patterns are located in the top layer of the well-known meta-model architecture. In lower layers the developed patterns lead to computer solutions for specific coupling problems in civil engineering. The paper describes the coupling assessment in a qualitative way.

The research work is not yet finished and must be continued. Our goal is the quantitative assessment of computer model couplings in civil engineering. Our future methods to achieve these goals are the complexity theory and the sensitivity analysis [13].

REFERENCES

[1] C.M. Eastman, *Building Product Models*.CRC Press, Boca Raton et al., 1999.

[2] T. Hauschild, R. Hübler, H. Willenbacher, *Distributed Cooperative Building Models for Revivification of Buildings*, International Association For Bridge and Structural Engineering (IABSE) Symposium 2002, Melbourne, 2002.

[3] B. Firmenich, *Consistency of a Shared Versioned Model for Distributed Cooperation*. In: Computer-Aided Civil and Infrastructure Engineering, Volume 20, 424–430, 2005.

[4] A.H. Olivier, G.C. van Rooyen, B. Firmenich, K. Beucke, *Supporting consistency in linked specialized engineering models through bindings and updating*. In: Proceedings of the 12th International Conference (ICCCBE-XII), Tsinghua University, Beijing, 2008.

[5] C. Koch, B. Firmenich, *Building Information Processing in Cooperative Building Design*. In: Proceedings of the 16th International Workshop of the European Group for Intelligent Computing in Engineering (EG-ICE), Technische Universität Berlin, Berlin, 2009 (accepted paper).

[6] G. Myers, *Reliable Software Through Composite Design*. Mason and Lipscomb Publishers., New York (NY), 1974.

[7] L.H. Rosenberg, L.E. Hyatt, *Software Quality Metrics for Object-Oriented Environments*. Crosstalk Journal, Volume 10, 1997.

[8] Y. Lee, B. Liang, F. Wang, *Some Complexity Metrics for Object Oriented Programs Based on Information Flow*. Proceedings: CompEuro, 302-310, March, 1993.

[9] B. Firmenich, E. Rank, *Überblick zum Themenbereich Verteilte Produktmodelle*. In: U. Rüppel (Ed.): Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau, Berlin, Heidelberg: Springer-Verlag, 2007, p. 121–132, ISBN 978-3-540-68102-1.

[10] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995, ISBN 0-201-63361-2.

[11] M. Born, E.Holz, O.Kath, *Softwareentwicklung mit UML 2*. Addison Wesley, München, 2005.

[12] Object Management Group, *"Meta-Object Facility (MOF) Specification"*. Version 1.4, http://www.omg.org/mof, April 2002.

[13] K.-U. Bletzinger, A. Lähr, *Sensitivitätsanalyse netzwerkübergreifender Produkt- und Strukturmodelle bei Planungsprozessen des Konstruktiven Ingenieurbaus*. In: U. Rüppel (Ed.): Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau, Berlin, Heidelberg: Springer-Verlag, 2007, p. 251–272, ISBN 978-3-540-68102-1.