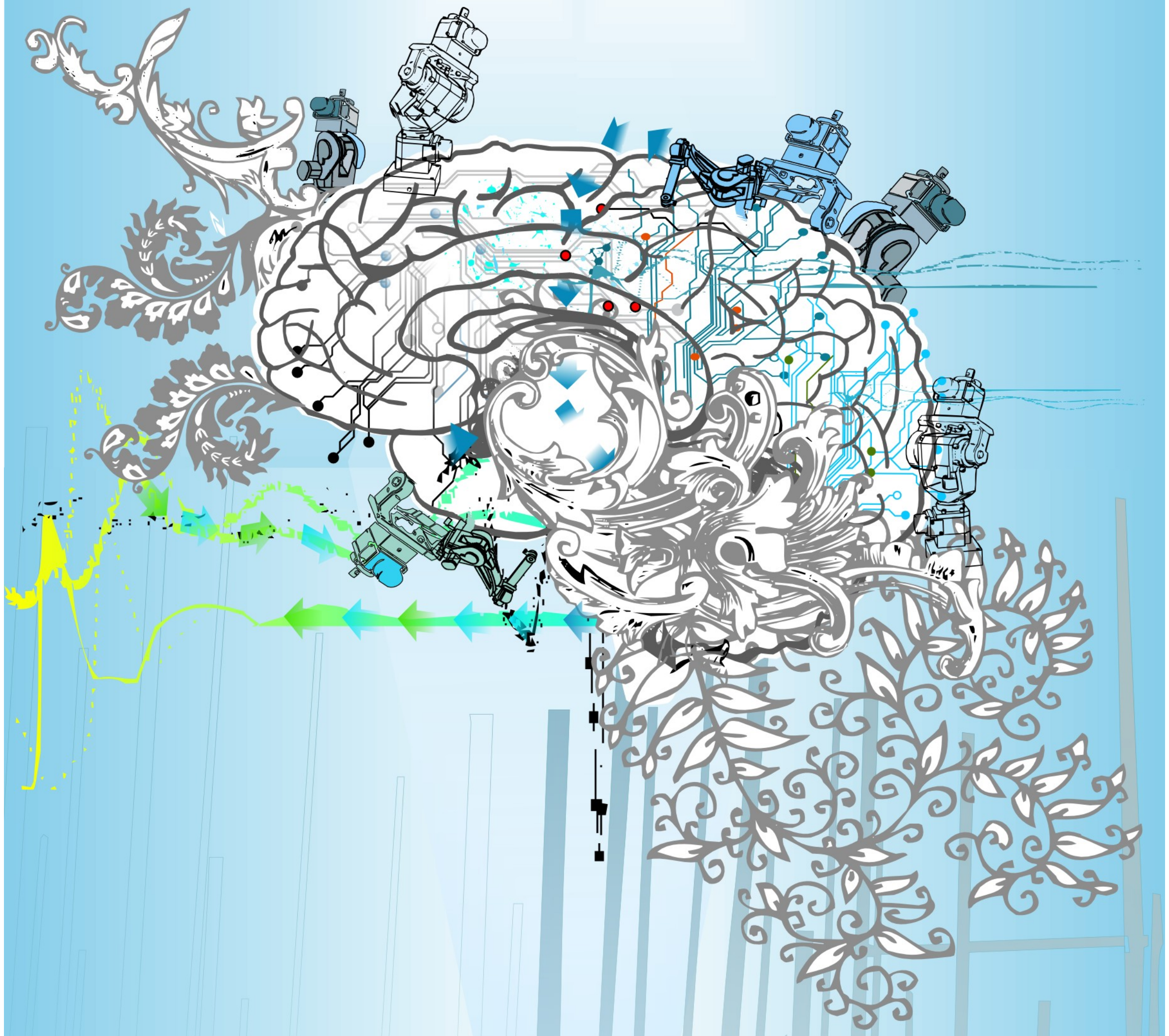


Multi-agent Theory and Managed Objects Applied to Civil Infrastructure Interoperability

PhD. Thesis

Jaume Domínguez Faus

Multi-agent Theory and Managed Objects Applied to Civil Infrastructure Interoperability

by

Jaume Domínguez Faus
A thesis submitted to
Department of Development and Planning
Aalborg University
for the degree of

DOCTOR OF PHILOSOPHY (PhD.)

April 2013

DECLARATION

I declare that the work presented in this thesis and its contents are, to the extent of my knowledge, the result of my own work, except for those cites, quotes, references and other source that are duly credited to their own authors.

Jaume Domínguez Faus

ACKNOWLEDGEMENTS

It is incredible how many people are directly or indirectly, consciously or not, involved in a PhD. research project. How many people have been next to you in critical moments of the amazing journey of becoming PhD.

I would like to thank Dr. Erik Kjems and Dr. Francisco Grimaldo Moreno, my supervisor at the Aalborg University, my university, and my co-supervisor at the Universtitat de València, for their guidance and extremely valuable help they provided me during these years and also for their belief in the viability of the ideas that shaped this work. Thank you Francisco for showing up and offering your unselfish help in a very critical moment of this research when everything seemed impossible and giving up seemed to be the only option. Your generosity went far beyond what I could expect. I am in debt with you.

Vianova Systems AS, my company. I want to express my most sincere gratitude to the company and the people for their comprehension, vision of future, and for their incredible values that are not easy to find out there, unfortunately. Thank you for demonstrating that the most important asset of a company is not their share holders but its employees.

Jan Erik Hoel, my project manager at Vianova Systems AS and my dear friend, your care and your level of professionalism always amazed me. I will never know how do you manage to convince people to believe in themselves and make them succeed. The world is scarce of managers like you. To you, my particular and "extreeeemely" big thank-you. Thanks to you, this thesis exists. I am in debt with you.

The Infraworld group (Jan Kolar, Andreas Haugbotn, Dr. Rune Åsgård, Manuel Madrid, Vicente Caballero, Ville Herva, and Matti Manninen), the interviewees (Vicente Chorques and Raúl Rodríguez Fajardo) for their contribution to this work. It has been a pleasure to work with you and I wish you all the best.

Wan Wen, my fellow PhD student and an everlasting smile, with whom I had the pleasure of doing research, share concerns and enjoyed comparing the differences between our cultures. Thank you and I wish you a complete recovery.

Dear Jaume Garcia i Segarra, thank you for providing me with your inspiring ideas and the most valuable gift you could give me, your friendship. The ups and downs of this life are incredible. We met for first time by chance 11 years ago a late night study day for final exams at the 24h library of the Universitat Jaume I. Who could say that you, economist, and me, engineer, whould share not only names, but also interests and a great and everlasting friendship? Who could say that 8 years later you, PhD student in Bargaining Theory at Universitat Jaume I, and me, PhD student at Aalborg Universitet in Denmark and engineer in Norway would also share PhD research? Jaume, my brother, meeting you has been one of those wonderful coincidences that very seldom happen in a lifetime.

My proofreaders Ana Pla, my friend, and Dr. Rosa Domínguez, who is also my sister, thanks for helping me to improve the quality of this work to a degree I could not reach by myself.

To my friends: Marcos, Juane, Jesús, Steinar, my band mates and many others who I can't include because it would be unreasonably long. Thank you for being part of my life.

Pedro, Maria Rosa (my parents), Lidia and Rosa (my sisters); my family. Thank you so much for you faith and your support throughtout all these years. Thank you for planting the seed of curiosity and criticism in me and for teaching me respect for what makes human race big: Science.

Also thanks to the Research Council of Norway for their financial support and their Industrial-PhD programme case no: 195940/I40 as well as the Spanish MICINN, Consolider Programme and Plan E funds; the European Commission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04-04; and the Vice-rectorate for Research of the Universitat de València (UV) funds, under grant UV-INV-AE11-40990.62.

I left the best to last. Patricia Gadea Company, my love. I cannot express my gratitude for your patience. You have endured much more than what it is reasonable to expect and given more than I can ever hope to repay. I can hardly explain how much do I admire you, your beautiful soul, tenacity, intelligence, and the many other wonderful qualities you have which would take unreasonable space just to name here. It is not who you are but who I am when I am next to you. You are my light. Thank you so much for existing and for choosing me.

Contents

Preamble

When I was enrolled in the Aalborg Universitet as a PhD student, one of the first things to decide was what the structure of my thesis would be. The decision consisted in choosing between a monograph thesis or an article-collection thesis. In the last years, it seems to be a tendency towards choosing the latter approach in connection with the predominant global and distributed research. This tendency is due to the believing that the more often you publish the more chances of connecting to other similar research teams you have. Also, there is the fact that partial works are peer-reviewed beyond the boundaries of the researcher's immediate network. In the article collection approach, though, one might end up with a set of pieces that the reader could find somehow disconnected. In contrast, the monograph thesis should give a stronger feeling of *continuum* since the contents reflect a time linewhere the research evolution is more naturally seen. By choosing the monograph approach, it is probably easier to write the thesis, but this comes at the price that the effort has been done with the only feedback of the researcher's immediate environment (i.e. colleagues, supervisors, etc.). Due to the nature of my research, I considered that the article-collection approach was more suitable than a monograph. This thesis deals about a known topic, the interoperability, which could suggest that the monograph approach could have been more appropriate. Interoperability is an intense field of study. However, I decided to address it in a way that, to my knowledge, is not really explored. Hence, I considered that publishing partial results as they came from work progress was a good beacon that puts some light in the long and uncertain journey that research is. Especially with the hopefully innovative ideas that are exposed here.

Thus, the main corpus of the thesis are the 4 articles I published, which are connected with some materials dealing with the concepts it is based on as a means to connect the pieces. The articles presented in here are a faithful reproduction of the ones I have published, but differing only in the formatting in order to make them better fit in the thesis template. In addition to these articles, the connecting materials were included with the aim to give a more read-through feeling to this thesis. If I succeeded, these materials should take the reader from a starting point to partial results that are realized in each published article. A side effect of including these materials is that sometimes the reader might find him- or herself reading content on a chapter that has been already mentioned in another earlier one. Regarding this, I kindly ask for reader's patience as my solely intention was to reduce the feeling of a disconnected set of articles as much as possible.

Summary

This PhD thesis deals with the concept of Interoperability within the Civil Engineering domain. The process of Civil Engineering comprises four phases that every project follows: 1) Data Gathering (where information about the area where an infrastructure will be built is collected and preprocessed), 2) Design (where engineers design the infrastructure itself), 3) Construction (where the infrastructure is built following the designs produced in the previous phase) and 4) Maintenance (where the infrastructure is in service and monitored, maintained and, when needed, fixed). This thesis focuses in the Design phase. The field of Civil Engineering might find this work interesting because today there is a lot of money that is lost due to bad designs. As it is shown, interoperability has an influence in the quality of the design and, even though it is not the only reason, it is studied through this work as a way to improve the quality of Design with the aim to contribute reducing the current resource waste.

Two types of interoperability are identified: the interoperability of engineers sharing data (data exchange interoperability) and the interoperability of engineers interacting with each other when they collaborate (distributed behavior interoperability). The former was studied in collaboration with another research project and resulted in one published article. The study of the latter constitutes the largest portion of this thesis effort and resulted in three published articles, one of them in the most influential conference worldwide in the field of Multiagent Systems, the Autonomous Agents and Multi-agent Systems Conference (AAMAS) 2012.

The impact of interoperability in the performance of the projects is analyzed. Problems are detected and solutions are explored.

Data exchange interoperability issues are tackled with a technology called Managed Objects which guarantees that the information is not lost at any of the ends when it is transferred. So, the traditional issues of information loss in exporting and importing from one application to another are eliminated. It is demonstrated how a paradigm shift allows

to use the same data and behavior of the model unequivocally in heterogeneous consumer applications and thus, avoiding the aforementioned issues. By creating a network of strategically designed and orchestrated objects, it is possible to solve some of the problems caused by poor data exchange interoperability.

Distributed behavior interoperability is tackled with the use of Multiagent Systems and Machine Learning techniques that allow capturing the project's distributed cognition (the combination of the skills, expertise, tastes, etc. of the members of the project that collaborate towards its achievement). Due to the collaborative nature of Civil Infrastructure projects, design conflicts happen during the design phase. Nowadays, they are treated manually because current applications are unable to implement every expertise's knowledge for every subdomain that Civil Engineering involves and combine them in order to compute a reasonable solution for the conflict. It is demonstrated how Multiagent Systems can conduct their interactions by means of negotiations which allow capturing the project's overall cognition in a simple and elegant way. The cognition captured is used to teach the system which, by means of Machine Learning algorithms, gains in competence and becomes able to solve these conflicts even though it was not initially programed to do it.

Finally, it is worth mentioning that this is an industry-related Ph.D. research. Therefore, the desirable outcome of the research is the creation of a new generation of technology that, by improving interoperability, boosts the quality of the Civil Infrastructure models used in industry.

The research fieldwork performed initially pointed out the subjects to review in the literature as well as the future tendencies that, even if they are only used by the most advanced companies, they are called to be a standard in the near future. These tendencies -either technologies or ad hoc methodological developments- are evaluated in order to detect weaknesses and to suggest alternatives.

Chapter 1
Introduction

This thesis is the result of a joint effort between academia and industry through the Industrial-PhD programme started in 2009 by the Research Council of Norway. The Industrial-PhD programme is an initiative that aims to create added value in Norwegian companies with the intention to provide them with distinctive technology in their competitive sectors.

The company from which this thesis emerges is Vianova Systems AS in combination with its sibling company Vianova Plan og Traffikk AS. The former is a company that specializes in the development of software for Civil Engineering design. The latter is a Civil Engineering company that designs and plans some of the most relevant infrastructures in Norway such as Operatunnelen in Oslo [StatensVegvesen, 2010] using tools created by Vianova Systems and other companies. This thesis, was written while the author was an employee at Vianova Systems and it is an attempt to identify interoperability problems in Civil Engineering and explore solutions from a software development point of view.

This thesis deals with the interoperability problems that ultimately influence the decision making processes. Concretely, the problems related to data and knowledge exchange among the members that collaborate on a project. The thesis tries to provide tools to help in the decision making. Currently, engineers make mistakes because the way the design decisions are made implies errors. The consequences are that budgets are often long overrun because of bad decisions. It is widely accepted in the industry that “estimating [budgets] has always been one of the weakest link in the construction process” [DelPico, 2004] even though “there is a consensus on the fact that cost management is one of the most important components of projects” [Mubarak and Means, 2012]. Many factors influence these deviations. This work identifies some of those problems that happen when a construction is being designed. More concretely, in the problems that arise from poor interoperability and cause misunderstandings, both among people and the computer tools systems used today, and proposes solutions for them with the aim of saving these extra costs. In order to more clearly delimit which part of the process this thesis deals with, it is necessary to briefly introduce the process globally.

1.1 The Civil Engineering Process

Very often the Civil Engineering process is explained by means of the phases that form it. Thus, from its beginning to its end, a Civil Engineering Project goes through several phases. They are: Data Gathering, Design, Construction and Maintenance.

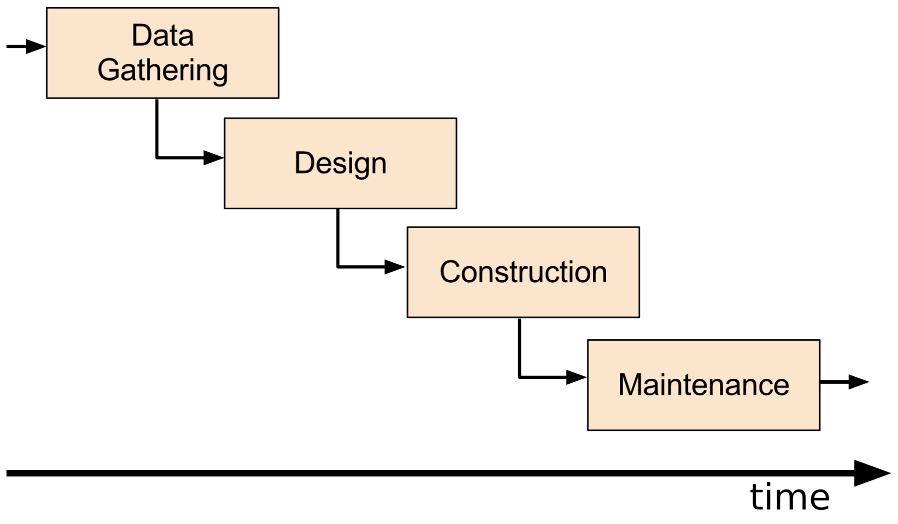


Fig. 1.1: The distinct phases composing a project[1]

Very often, due to the size and diversity of tasks each phase involves, they are carried out by different companies. Because this thesis deals with interoperability, it focuses on the flow of information.

[1] It is worth clarifying that this sequence defines the order in which the distinct phases occur. It means that, for instance, the Construction Phase would not begin before the Design Phase. But the transition to one phase to another is not necessarily abrupt in the sense that there is a point in the project's timeline in which the project leaves Design Phase and tackles Construction Phase. However, the transition from one phase to another necessarily abrupt, but rather gradual. During the transition from phase to another the amount of work of the previous phase gradually decreases while the amount of work of the next phase increases.

Each of the phases consumes and produces information according to its needs and works. A flow of information is created along the life of the project. The project's lifecycle is depicted in figure 1.1. Phase-by-phase -and paying attention to the information flow- a Civil Engineering process consists of:

1.1.1 Data Gathering phase

The gathering of information is a process of collecting the necessary mapping data and documentation that describes the environment where the infrastructure will be built. Because infrastructures are built in specific locations, it is required to know the existing situation of the area of the works. Data Gathering is often regarded as the very first task of the Design phase, i.e. within the Design phase. In this thesis though, it is presented as a phase in its own right due to the challenge it represents and the fact that it is a job that is always done prior to any subsequent design.

Civil infrastructure projects are offered in public bid for the companies to disclose their offers. In order for these offers to be as close to reality as possible, data about the existing situation is provided by the authorities. However, this data is normally not detailed enough to be the base of a final design. It often needs to be completed once the project is awarded. For instance, the information provided by the authorities might be precise enough for, say, road networks, but data regarding electricity networks or piping is normally more reliable if it is obtained from their owners, i.e. the electricity companies or the water supply companies. Thus, this kind of information requires that contacts with these companies are made and the corresponding agreements on the data terms of use are settled. Geological information is also necessary, the goal is to know what kind of terrain is going to hold the infrastructure (e.g. rocky, sandy, etc.) because it will determine which land movements, foundation settlements and so on strategies fit best. Traditionally, geological data has been encoded as an abstraction of the composition of the ground by means of vector maps with polygons denoting the type of land in the area they represent. More recently, other surveying techniques like laser scanning (LiDAR) are becoming more frequently used to obtain a more realistic and massive data that complements in the 3D maps. Demographic and land use information is also needed for accurate decision making. For instance, decisions like the type of road to be built (a populated area is more likely to prefer low traffic roads than highways).

This phase concludes with data treatment to adapt it to the formats that will be used in subsequent phases. The destination formats to be used are agreed over by the project participants, especially when there is more than one company involved, because it has to be guaranteed that the information is usable by all the parties. If it is possible that one format is suitable for all, then the tendency is to use that one. If such a format does not exist then the parties also agree on how the information will be transformed from its original format to the desired ones.

1.1.2 Design phase

In this phase the infrastructure to be constructed is designed. Taking the data collected from the previous phase as the existing situation the engineers produce the drawings and the documentation describing the result of the project. This information is composed of paper drawings, 2D digital plans, documents and, more recently, 3D models are becoming more frequent as well. The design is carried out by teams of engineers from different disciplines. For instance, the construction of a road is based on a design mix with contributions coming from different domains like earth moving, water supply, drainage, telecommunications, gas lines, electricity, road engineers, etc..

Depending on the design company methods, the designs are created and evolve in parallel in a rather disconnected manner or, on the other hand, a coordination policy can exist by which the engineers discuss what they have done and evaluate how that affects the overall project. In the first flavor there are high chances that when the finished design is delivered to the constructor, misunderstandings, omissions and errors happen so that the designs need to be changed. Even in designs that were accepted often suffer from problems of regulations that are not complied because these regulations changed in the middle of the process. This causes lots of disputes between both parties as well as losses due to the replacement of portions of the already constructed parts of the infrastructure.

On the other hand, it is not unusual still nowadays that the constructor wants to work with paper drawings. Thus, even if the design company has already switched to a more sophisticated methodology and tools for design such as 3D models, the design is delivered in a support that is more difficult to understand and prone to misinterpretation errors that can derive in legal disputes and overheads.

1.1.3 Construction phase

According to figure 1.1 the Design phase precedes the Construction phase and provides the designs used to build the infrastructure of the project. In the Construction phase, the tasks that compose the work are sorted temporary- and spatially in order to ensure that a task does not start before all its preconditions are met. I.e, tasks depending on previous tasks are performed only once the former ones are finished or have progressed enough to let the latter one start without problems. Some examples are: earth movements to establish the foundations of the infrastructure; establish the supply chain in order to provide the raw materials for the constructions; and more.

If the designs received in this phase are inaccurate or contain errors then conflicts will arise in the field. Solving these conflicts onsite can be very costly. Particularly if they require the demolition of any already constructed part. The errors can be due to the fact that the designs are not made in the field and thus some conditions cannot be controlled (e.g. weather conditions), or were not anticipated (e.g. finding an unexpected underground water body, old industrial installations, changes in the availability of materials, etc.), or simply because the design has inconsistencies or omissions. Thus, it is common that the design suffers many changes. Depending on the size of these changes, the project might suffer delays, budget overruns, legal disputes, etc. It could even become unfeasible with the entire project needing to be started from scratch.

1.1.4 Maintenance phase

The Maintenance phase is the longest one and, at least in theory, it spans for the rest of the infrastructure's life or until a new construction project replaces the old one. The Maintenance phase is often carried out by the owner. It consists of ensuring that the installation is in good shape and it involves reparations of broken parts or areas. From components that have been broken due to accidents or natural catastrophes, to the replacement of parts which reached the end of their life like, e.g., light bulbs of a road illumination system or some equipment that has to be moved in order to let other construction to progress. For instance, a gas conduction that has to be moved because another project's manhole is going to be installed at the same place.

The Maintenance phase also benefits from the designs delivered in the Design phase. For instance, if a bridge has been damaged as a consequence of an unusual flooding, the experts that will evaluate if the consistency of the structure has been compromised need to know certain details such as the beam connections, load hot spots, the materials used and so on.

It has to be noticed that the data describing the infrastructure still evolves during the Maintenance time. But this evolution is very different from the evolution in the Design phase. Normally, the owner that is maintaining the infrastructure is mainly interested in maintenance information such as the wattage of the bulbs or similar. Thus, while the quality of the data can remain acceptable in these aspects, it will not be in other aspects if, for instance, changes in the real construction (and thus in the geometry) are not registered. This happens because once built, the geometry and structure data become much less used than the descriptive data used for maintenance. Hence, there are more chances that the formers become outdated. This phenomenon is important because it means that there is a point in which data becomes unusable for future projects and, therefore, future projects cannot benefit from it. If this happens future projects are forced to perform a full "Data Gathering" phase again.

1.2 The Business Model

The civil infrastructure business model can be classified in two groups regarding the contractual relationships established between partners. The first one is what is called Design-Bid-Build (DBB) and the second is Design-Build (DB). In the DBB model the Design and the Construction phases are more decoupled than in the DB model. That is because in DBB the construction works are awarded to another company than the one making the design as a result of a bidding process. Construction companies look at the design; they make estimations on it and issue an offer. The design company then selects the most interesting offer and that company receives the task of realizing the design, i.e. of building it. DBB is the preferred one for public constructions as it involves less political pressure to the owner, i.e. the Administration, because the responsibility on project deviations is delegated to the designer who chose the constructor. In turn, the constructor can decide which supplier will be used. On the other hand, the DB process is more closed and simpler because both the Design and Construction are contracted directly and at the same time in a closed package. Since the DB project structure is closed and all the relationships are contractually established, the Design and its cost estimation need to be agreed and signed. As a consequence, the owner is who carries responsibility. However, DB tends to be faster and cheaper to incorporate changes and is becoming more common [Eastman et al., 2008]. When it comes to data, DB approach has better conditions to keep good quality in the models because, in principle, the data needs to flow to more integrated users.

1.3 Interoperability

A Civil Engineering project can be executed thanks to the collaboration among participating engineers. Engineers in a project come from different domains and contribute with their know-how to it. Interoperability is what enables the engineers to work together and collaborate and it is the central concept of this thesis.

Thanks to Interoperability, engineers of a project coordinate and act as a whole. The project benefits from its members' knowledge, skills, and -especially- cooperation to achieve tasks that would not be possible individually.

Necessarily, Interoperability exists in any team working collaboratively (see Figure 1.2a). Otherwise, it would not be a team but a collection of people without a structure who cannot benefit from what others can achieve (see Figure 1.2b).

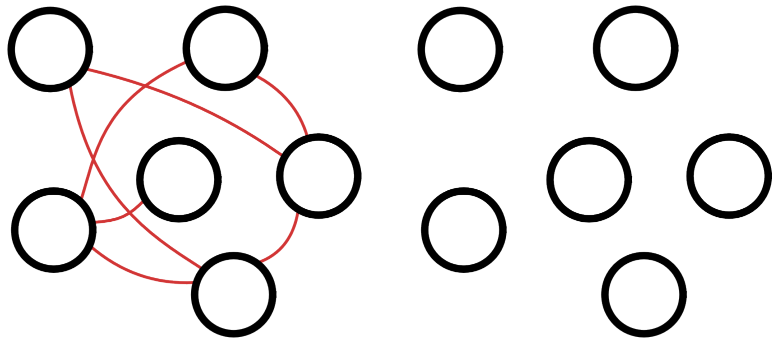

a) Members of a project interoperating b) collection of isolated individuals

Fig. 1.2: Schematic representation of interoperating team vs isolated individuals

The importance of a good Interoperability in Civil Engineering derives from the fact that the knowledge created in a project is product of not only the sum of the individual competences but also of the interactions among them. As a product is designed, it experiences changes due to corrections and contributions from all the individuals. Such changes are the result of decisions made when the project participants meet to make decisions throughout all stages the project is composed of (Figure 1.1.). During the decision making process, engineers decide about things that despite they are designed separately by different disciplines (e.g., drainage, structures, signaling, etc.) will have to coexist.

1.4 Impact of interoperability in the Civil Engineering Process

In a collaborative distributed system where information is shared and travels from one member to the rest, interoperability denotes how fluent and transparent this information exchange is. Higher interoperability is achieved when the necessary time to transfer information is shortened, the effort reduced, and the amount of information lost in its way to the destination is minimized.

The impact of interoperability has been acknowledged by advanced industries like the automotive or the aerospace manufacturers who have benefit of increasing it [Laurenzo, 2005]. More recently, Civil Engineering has also started the process of increasing it. Thus, 2D drawings are starting to be replaced by digital 3D models (data) and modern workflows (knowledge). Traditionally, 2D drawings have been considered the easiest way to transfer knowledge due to the easy access to paper support. However, with the increase of the information technologies, this belief is fading out. As an example of this, consider that reading 2D drawings is more difficult and requires more experience to avoid wrong interpretations than observing a realistic 3D model. Nevertheless, exchanging and managing data and knowledge is challenging and interoperability is of key importance.

All phases in the Civil Engineering process benefit from a good interoperability. In the Design phase, engineers benefit from interoperability because they can exchange their designs faster. Ideally, if changes in one design are propagated in real-time, other designs affected can be updated immediately such that mistakes can be avoided. In the Construction phase, constructors benefit because the quality of the designs increases. Also, if an unexpected situation in the field happens, a good interoperability can allow the constructor to notify engineers at the same moment it occurs so they can start working on a solution before it is too late. Because constructors can have more reliable designs, they can estimate the costs in material, labor and time of the works more accurately. In DB projects, this would also benefit the Design phase in which the constructor also takes part in order to issue a realistic and feasible budget. This allows owners to adjust their budgets and optimize their resources before and after the project is constructed, i.e. the Maintenance phase. Additionally, a good interoperability can dramatically help keeping the quality of data during the Maintenance phase for longer time and thus, as a side effect, increasing the chances that it can be reused by future project's Data Gathering phase. Finally, an interoperability that smoothly handles the heterogeneity of data will let the models to be more maintainable and, especially, richer.

1.5 Thesis coverage

This thesis circumscribes to the Design phase because it is the phase that conducts the development of the whole process and because it is the phase where the company where this research was conducted has its biggest interests. Thus, other types of interoperability challenges in the Construction phase are set aside since they are out of the scope of the research. In the Design phase, the collected and adapted information describing the existing situation is completed with the designs of the future construction. During the whole Design process, data and knowledge exchange is essential for the design teams in order to ensure the engineers have a detailed understanding of the project state and, thus, make the right decisions; in other words: good Interoperability. If the exchange of data and knowledge is not satisfactory (i. e., poor interoperability) then potential problems derived from misunderstandings might appear on its way to the recipient.

Take this situation as an example. An elevation model is delivered embedded in a mapping file and encoded by means of contour lines. The recipient of this data wants to use it to calculate the volume of moved lands. To do that, the recipient needs to get a triangulation of the relief. But the mapping file does not contain contour lines only. It also contains data about land use, buildings, forests, etc. In order to get an exact triangulation, only a subset of the objects in the map has to be considered. If during the processing some important objects are discarded or wrong objects are included by mistake the surface of the triangulated land will not be realistic; and so will the volume calculation be in consequence.

Conversely, it is easy to see that with a high quality design, a better result can be expected in the final construction.

Currently, interoperability among design teams has a very marked manual component. Project's control meetings where the designs are validated consist of engineers discussing how to solve the defects that could be detected. Many impressive constructions like the Ciutat de les Arts i les Ciències in València, Spain [C.A.C, 2012], the Operatunnelen in Oslo [StatensVegvesen, 2010], or other older ones such as the Panama Canal and Suez Canal to name only a few are high-level examples of what can be done with this approach.

Nevertheless, budget overhead still being the norm in the majority of the projects such that a portion of the budget is systematically added on top of the estimated costs and reserved for the correction of errors and covering uncertainties [Touran, 2003]. Figure 1.3 shows the average and normalized distribution of costs extracted from the historical records of Vianova Plan og Traffikk AS classified by size of projects. It can be observed that an important portion of the budget (between 5% and 15%) corresponds to errors and the unforeseen. There are authors that would even consider the sizes of these portions as very conservative (e.g. [Eastman et al., 2008] page 99) since they present numbers that depending on the nature of the project can rise up to 40%.

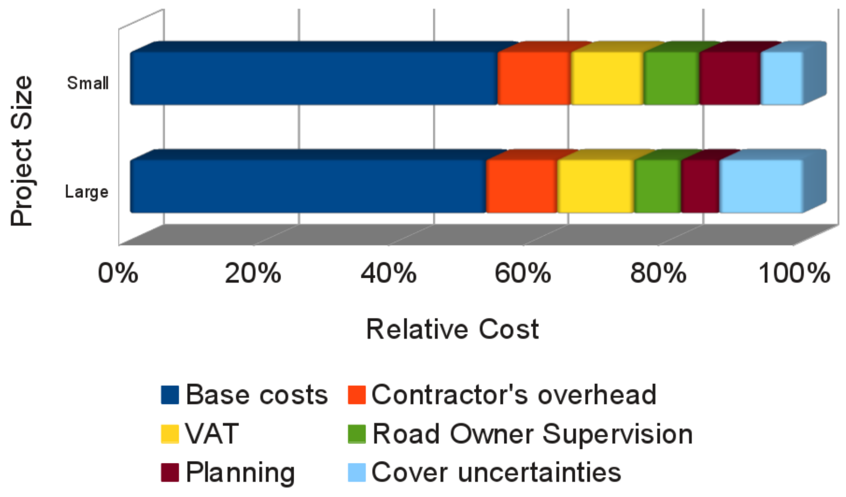


Fig. 1.3: Normalized Average Costs of Civil Engineering Projects classified by size. Small projects refer to small street or road stretches, roundabouts and the like. Large projects include highways, bridges, tunnels, and so on. Source: Vianova Plan og Traffikk AS. Used with permission.

Other surveys like [FMI/CMAA, 2005, FMI/CMAA, 2006] and [Clients, 1997] reveal that up to two-thirds of projects report cost overruns.

This thesis aims to enhance the designs in the Design phase by means of identifying and proposing solutions to interoperability problems in order to reduce budget overheads of Civil Infrastructure Projects.

The study of the interoperability problems at the Design phase drove the findings presented in this thesis and some solutions for those problems are proposed as a result of those findings. As will be shown, these solutions can open a new range of possibilities to enhance interoperability from which projects can benefit.

Chapter 2
Problem Statement

Despite the fact that Civil Engineering is one of the oldest and most resource consuming engineering disciplines, the decision making in design is still relying in a manual process. However, an automated coordination "can have a huge impact in strengthening the quality of decision made in Design phase based on quick feedback" [Eastman et al., 2008]. In his work, Eastman acknowledges the need for better tools for easing and automatizing the decision making. There are some reasons preventing these tools to exist derived from the size and complexity of projects. The complexity encourages the separation in semi-autonomous subprojects in concrete knowledge domains which, in turn, encompasses heterogeneous needs and points of view. No matter how, good coordination of those areas is a considerable challenge because on the one hand, the project data has to reach the components of the sub-projects who need it, and also because even though a project is divided in sub-projects, it is a whole and it has to exploit the contributions of its members.

In short, the problem of interoperability in the Design phase of Civil Engineering projects is about the interactions that occur there. The interactions benefit from good interoperability. In this thesis, two kinds of interactions are identified: (1) the interaction ocurring when a member of a project shares data produced so others can benefit from it, i.e. data exchange; and (2) the interaction that occurs when the members of the team put their effort and skills for the overall project, i.e. knowledge exchange. Each type has its own problems, which are described below. Afterwards, some basic definitions are given for terms that are used throughout this work.

2.1 The problem of data exchange

The existence of different domains resulted in a considerable variety of data models to represent those domains. For instance, structure calculation requires a data model which substantially differs from, say, that of electricity installations or water pipe networks. When trying to identify and solve problems, combining the different models becomes a challenge. The reason being that because in order to combine them, different models need to be adapted to a specific, perhaps previously agreed, format (or set of formats if it is not possible to single one out capable of combining all the models and needs). The adaption does not only refer to data *per se*, aspects like the coordinate reference and units used in the models also need to match, etc. which shows how cumbersome the adaption process can be. Unfortunately, the adaptation also comes at the cost that information contained in the original models might be lost in the process of transforming it to the destination model. Hence keeping the data throughout the process is so difficult.

As models become more specialized and sophisticated, this phenomenon gets amplified. In order to mitigate the costs of adapting information from one model to another, industry creates intermediate exchange formats[1]. Although it is true that these formats do mitigate the costs, it is also true that they are formats after all and, although maybe in less intensity, they suffer from the same problems of concept than the others.

2.2 The problem of knowledge exchange

Because there are subdomains (e.g. Road, Railing, Water, Landscaping, Structural, Architectural...) Civil Engineering projects are complex. This complexity becomes apparent when there is an attempt to implement it in a computer system. There are certainly tools for assisting in specific tasks which are already a good help. But it is significant that no tool has emerged as the flagship of managing and exploiting the knowledge for solving interdisciplinary design issues. When a design issue is detected as a consequence of interferences between two domains, the decision about how to solve it involves, at best, all the particularities of those two domains and all of the others at worst. Since each domain has its own type of decision consideration factors,

[1] A discussion on the current popular exchange formats is presented in Chapter 4. Background.

the amount of factors in decisions involving more than one domain is equal to all possible combinations of the individual factors from each domain. In the case of two domains, for instance, it is equivalent to the Cartesian product[2] of factors from the first and factors from the second domain. If there would be a third domain, then the Cartesian product will be of these all three and so one. Because the space of factors to consider increases exponentially, it will very soon become bigger than what is technically achievable. It does not seem feasible that a computer system could consider all the factors that allow shaping the deliberations on how to solve the conflicts that are detected. Not only because there are so many factors that they can be hardly catalogued in a taxonomy, but also because these factors change from a given situation to one another, and because often times engineers incorporate those factors subconsciously. Thus, they can very hardly be captured as a requirement for the tool. Hence, the decision making process remains an eminently manual one. In other words, it is difficult to define a clear and feasible set of requirements from which a satisfactory design for a computer system can be emerged.

Nevertheless, an automation of the knowledge exchange is desirable so that it is possible to increase the quality of the designs. Since the standard paradigms of software design and development[3] have not been able to respond adequately to this challenge, it seems necessary to explore new ways and approaches to this problem.

2.3 Definitions

The role of interoperability can be identified in the two problems above. That is why interoperability is addressed in two different levels along this thesis. They are named interoperability at the "**data exchange level**" and interoperability at the "**distributed behavior level**" corresponding to the problem of **data exchange** and the problem of **knowledge exchange** respectively. Hence, this thesis has been split in

[2] A "Cartesian product" between two sets is a well-known concept that takes two sets and computes a new set whose elements are all the possible combinations among all the elements in the first set and the elements in the second set.

[3] I.e., the traditional approaches of algorithms written to implement a set of requirements and features of the computer tool to be developed.

two parts to clearly delimit the two problems dealt with. At any rate, the two problems are addressed with the intention of tackle the main problem of **reducing costs** in Civil Infrastructure projects in two of the multiple faces it has.

Chapter 3
Methods

This chapter describes the methods applied in this project. As usual, the project started with a **literature review**. In this case, the initial review was regarding qualitative research in order obtain tools to do ethnomethodological research correctly [Blomberg et al., 2002]. Based on this review, an ethnographic study (described in section 3.1) was carried out with the aim of identifying what problems are not fully considered, or satisfactorily approached in today's Civil Engineering Design phase. Interviewing was another ethnomethodological method (section 3.1) used for the research. Even though interviewing had a less predominant role in the research compared to ethnography, good practices in interviewing are also necessary. So, literature regarding interviewing was also reviewed.

After ethnographic data were collected and analyzed, and conclusions were obtained, review of the literature was carried out to establish the research's starting point regarding (1) the data interoperability part of this work [Adachi et al., 2003], [Basanow et al., 2008], [Eastman et al., 2008], [Kolbe et al., 2005], [Kolar, 2007], as well as [Kjems et al., 2009] (Part I); and (2) the field of Computer Supported Collaborative Work (CSCW) including works like [Fitzpatrick, 1998], [Greenberg, 2001], [Halverson, 2002], [Erickson and Kellogg, 2000], [Crabtree et al., 2005] and [Pipek and Kahler, 2006] in the collaborative aspect of this thesis (Part II). Regarding Part II, it is worth mentioning special reference to Fitzpatrick's work, as it had considerable influence in the way the problem was later approached.

Finally, in later phases of this thesis, the Agile design and programming methodology used to implement the experiments also required a previous literature review (see 3.3 for more details).

3.1 Ethnography

Ethnography is derived from Anthropology in social sciences. Originally, ethnography was used to register the customs of non-western communities to make them known to others, mainly western ones. In the mid twentieth century, ethnography mutated into a way to explore the daily and mundane life of small-scale societies rather than whole civilizations. The ethnographic method is based on the assumption that humans are able to understand what is going on through participating in the social life of the society under study. Today, ethnography is used as a resource to design systems for very specific settings within a large advanced society.

An ethnographic approach for studying a community can be more adequate than other approaches like surveys, interviews, etc. as it allows direct immersion and firsthand view of the everyday work. It is also long recognized that what people say they do and what they actually do can vary significantly [Blomberg et al., 2002]. There are many reasons for this. First, people have only limited ability to describe what they do if they are not in direct contact with the social and material environment. Many aspects may potentially be skipped if relying on questionaires. Another reason is that studying processes in isolation (with no connection to the other processes it depends on, or that are dependent on it) can not provide a full understanding of those processes. Or, even, because some people might be interested in saying something that is not true if they consider that telling the truth could affect their position.

Ethnography is interested in the insider's point of view. This is something that is largely accepted in American technology companies where sociologists are hired for increasing the chances of getting a good design. If a specialized solution is to be proposed, a good understanding of its users is necessary. The outcome of an ethnographic study is descriptions of the people's everyday tasks. This text is expected to be agnostic, evaluation free. Only then, the descriptive text can be used to suggest how things could be different. That follows the convictions of: 1) to evaluate a situation and to improve it, one must know it as it is; and that 2) the “innovation is the imagination of what could be based on a knowledge of what is”[1].

An agnostic description is essentially a utopia. Any observer will always introduce a biased point of view, simply because the things we perceive are unavoidably shaped by our past experience. Given the Computer Science background of the author of this thesis, this research will inevitably perceive more details in the computer system than, say, an economist who probably will unconsciously focus on the decision and economic facets of the tasks carried on by the subjects under study. It is not intentional nor an intended hidden strategy. It is just a matter of competence. Since this research aims to improve the interoperability using our technological advances, it will not claim that this study is a universal truth but a record of as much as the author could capture with a special focus on how the different members of this team work and collaborate, i.e. interoperate. That can be seen as this thesis’ version of “The Postmodern Inflection” [Blomberg et al., 2002].

In any case, in order to obtain a good record of the processes, it is necessary that participants trust the researcher. Gaining this trust is possibly the most time consuming task. The people being observed need to know that the information the researcher is obtaining comes with a total respect for them. In order to ensure that the research would not interfere in the daily work, it was decided to take a low profile approach; a low profile in the sense that, for instance, instead of asking questions in the middle of a process and interrupting it, the questions were annotated in a notebook and asked later to a person of confidence. In this case, this person was Andreas Haugbotn. He is Civil Engineer at Vianova Plan og Traffikk.

3.2 Interviewing

While an ethnographic study provides an extensive insight of the in-place processes, it may suffer from a lack of generality. After all, what is being studied is a specific setting. In order to abstract a general com-

[1] The slogan of the work practice and technology group at the Xerox Palo Alto Research Center [Blomberg et al., 2002]

puter system, the information gathered was contrasted against other workplaces, possibly very separated in the world so that factors that are amplified or diminished too much by local (Norwegian) sociocultural are addressed with the correct accuracy [Silverman, 2011].

Some interviewing guidelines were applied:

- When possible, people were interviewed in everyday, familiar settings. This is for two reasons: 1) to make participants feel comfortable and, 2) being in context allowed them to point out the elements they normally use.
- Casual/informal conversations tend to give relevant information as the institutional or hierarchical barriers are lower.
- Unnecessarily interruptions, completion of a participant's sentences, or answering interviewer's own questions were avoided.
- Yes/no questions were avoided, as they provide less information. It is not about to corroborate or invalidate the interviewer's personal point of view but to obtain the experts' knowledge or points of view.

Interviewing is normally used as a primary source for data collection. In this research it was used to contrast information and underpin conclusions, though. The reason being, as stated before, that often times what people say they do and what they actually do may vary considerably. It does not mean that interviewing has no real value, it definitely has. But this research started in a favorable position for a pure ethnographic observation approach. The problem of gaining access very often drives researchers to prefer interviewing to "living-with-them" [Blomberg et al., 2002]. Many times it is simply not feasible. Fortunately, in the case of this research it was because the research took place in the same building and within the same company of the studied offices. Consequently, gaining access was not an extra effort nor was it to gain the people's trust.

3.3 Agile design and implementation

Given the nature of the research topic, an agile approach to the experiments has been followed. Generally speaking, an agile method means that the general goal is only conceptual and generally defined.

Instead, small milestones are scheduled. When a milestone is achieved it is evaluated and, afterwards, what to do next is decided. Unlike it could seem, and because it is a fairly unexplored approach to the problem of interoperability, agile approach presented itself as a more conservative and suitable methodology. It minimizes the likelihood of a getting-nowhere situation, which would likely occur if a traditional up-to-down research methodology used for this project.

Agile approaches are becoming more popular in the recent years. This tendency somehow recognizes the need for a more holistic approach towards research in general. At the same time, at the end of each small milestone, it gives the opportunity to recap and "take a breath" to compare partial results with what the research project is aiming at, and to take the next step with fresh energy.

In computer science this translates to Agile Development. There are many flavors of agile programming like Extreme Programming [Burke and Coyner, 2003], Scrum [Rising and Janoff, 2000], Feature Driven Development [Palmer and Felsing, 2001] to name a few. They all share some conviction that the development cannot be fully planned before start implementing. What they differ in is basically which aspect they emphasize most.

When it comes to programming, an Extreme Programming (XP) approach was taken so that short milestones were defined by means of unit tests. Each unit test defined what input a process receives and compared it to the expected output. Figure 3.1 depicts schematically the XP methodology applied during the development process.

Thus, when, and only then, a unit test was written, ran, and failed for the first time, I started implementing the logic for the process. The unit test is executed very frequently for each change in the logic until it passes. A similar strategy was followed when a bug was found, i. e., don't try to fix it directly, but instead, write a unit test that reproduces the bug. Once the unit test is able to reproduce it, i.e. it is ran and it can be observed how the process crashes or misbehaves, then, and only then, fixes can be applied in the logic until this test and all the previous ones pass.

The benefits of XP are considerable. In big projects, it allows to see the set of tests as the updated documentation of the code. At least from a developer point of view, there is no clearer way to document a system than flooding it with working examples for almost every as-

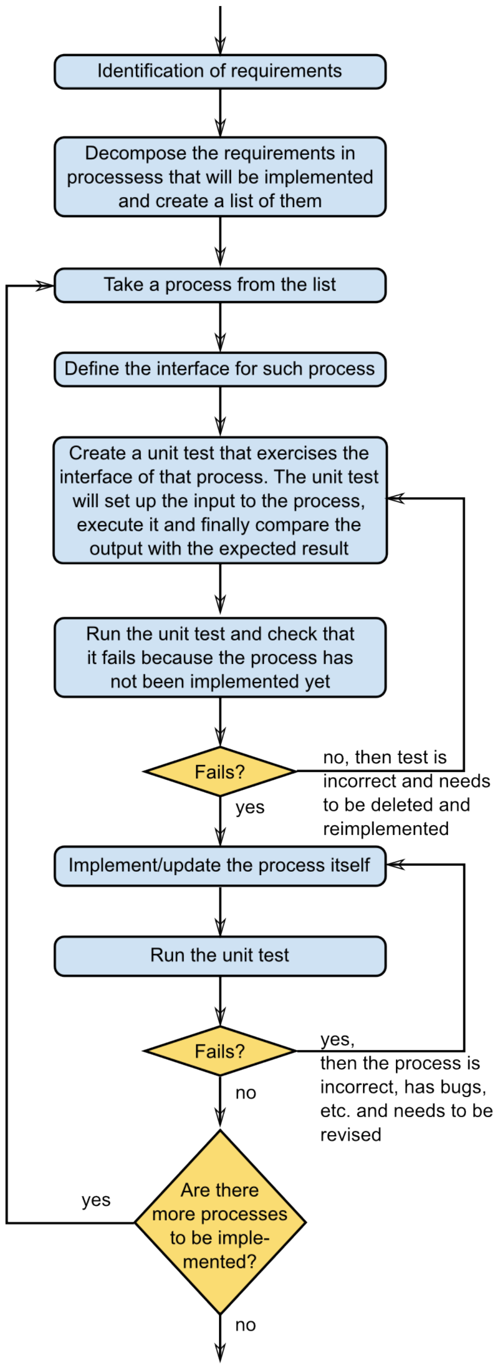

Identification of requirements
Decompose the requirements in processess that will be implemented and create a list of them
Take a process from the list
Define the interface for such process
Create a unit test that exercises the interface of that process. The unit test will set up the input to the process, execute it and finally compare the output with the expected result
Run the unit test and check that it fails because the process has not been implemented yet
Fails?
no, then test is incorrect and needs to be deleted and reimplemented
yes
Implement/update the process itself
Run the unit test
Fails?
yes, then the process is incorrect, has bugs, etc. and needs to be revised
no
Are there more processes to be imple-mented?
yes
no

Fig. 3.1: XP methodology applied to the development

pect and usage of the code. Moreover, the documentation is, by definition, up-to-date because otherwise either the tests would not be able to compile or would not pass. Also, testing and developing a big system through testing and developing the parts it is composed of makes it much easier to maintain. In XP, these principles are taken to the extreme -hence its name- where the system's pieces are granulated down to unit tests. In theory, a project that has been fully developed using an XP approach delivers a rock solid product since all the known issues have been covered by a dedicated unit test which, if it passes, it guarantees that it will not happen in a production environment nor will show up again in future versions of the software without being noticed.

However, XP has some cons which very often prevent it to be applicable. On the one hand, it is not only the real code what has to be maintained but also the set of unit tests. This could burden the work beyond the available resources. On the other hand, some situations are very difficult to capture in a unit test. For instance, visual graphics are difficult to test. In such cases what is tested is not the "View" but the "Model" and the "Controller" of the object of the MVC[2]. But even if the MVC pattern were used, it is often very difficult to write unit tests for it. Some suggest using Computer Vision tools for automatic testing. Nevertheless, using Computer Vision in this project is too difficult and unfeasible given the time constraints. Anyway, it was considered unnecessary given the fact that the graphics engine used was a third party library (Java3D) which relies in OpenGL. Both two libraries are tested enough for the purpose of this research by others. Another difficult situation to be unit-tested is asynchronous code. This is the case of testing interaction between agents[3] in a Multi-agent System (MAS) environment.

[2] MVC, Model View Controller pattern in which a component is internally represented by 1) a View which is dedicated only to show it in the screen it is read-only; 2) a Model which is the data representation of the object and that defines the View; and 3) a Controller component that encapsulates the input and output of both data and operations of the Model and also triggers and processes events from and to the object.

[3] The term "agent" can refer to two different concepts throughout this thesis: 1) to one of the parties involved in a negotiation [Nash, 1950], and sometimes regarded as "actor"; or 2) an agent as in Computer Science and Artificial Intelligence [Wooldridge, 2002] which refers to a program that acts autonomously, reactively, and/or proactively. Whenever a new use of the term agent is to be used, it will be explicitly said or marked in a footnote like this one. Here, "agent" refers to the Wooldridge definition.

However, it is not impossible. The unit test needs to be equipped with control variables in order to make the asynchronous code act synchronously. This makes the code for the unit test become harder to follow which could clash with the spirit of XP of making many very simple things and glue them instead of having a smarter, but also a bit more complicated, self-contained pieces. Consequently, unit tests covering agent interaction might require more effort to understand.

3.4 Foundation Software tools

It is a bit early to describe in deep the software tools that are the base of this research. However, given the multidisciplinary nature of this thesis, it is in order to provide with some first-contact information about them so that readers that are not familiar with some of the concepts can more easily follow the discourse. They are, in order of appearance, the following: **Geographic Managed Objects (GMOs)**, **Multi-agent Systems (MAS)** and the **JADE** framework.

3.4.1 Geographic Managed Objects (GMOs)

Any computer has a processor (or a set of processors) which is ultimately the responsible for executing the instructions that compose a program. A processor can only execute instructions that are expressed within its own architecture. Examples of architectures are *x86*, *amd64*, *ARMv7* and many more, which specify the set of instructions that the processor can execute. The processors can use different "word size" (what is classically referred as either 32-bit, 64-bit, 128-bit processors and so on). There is no need to go in deep about what all those terms mean. It is only necessary to clarify that a program that has been compiled [4] for a given architecture can't be executed in another one[5].

[4] Compiling a program is the process of transforming the source code (typically text files written in a specific programming language) into the binary code that the processor can understand and execute.

[5] In fact, it is in some cases possible to run 32-bit binary code in 64-bit processors. For instance, Intel and Intel-like processors can do it but this requires some support from the operating system.

The same compatibility problem happens at the level of the operating system. As an example, it is well known that some applications -actually most of them- compiled for Microsoft Windows^{TM}cannot be used in Apple^{TM}computers running Apple's operating system. That is because they are compiled natively for the processor-operating system pair (also known as *platform*).

To avoid having to compile the same for different settings (architecture and operating systems) some programming languages make use of Virtual Machine (VM) which is a program that simulates a complete computer and it is executed by the host operating system. This VM defines its own set of instructions, word size, and so on. Then, instead of compiling the program against the real processor and operating system specs, the program is compiled only once against the VM specs. The generated code is known as *byte-code* and has the advantage that as long the platform has a VM, exactly the same code can be executed in the computer regardless the processor and operating system types. This is the the approach taken by the Java programming language (and the Java Virtual Machine). The GMOs are a framework created by Dr. Jan Kolář for geographic information that is written in Java in order to exploit these advantages. The GMO's are described in detail in Chapters 6 and 7.

3.4.2 Multi-agent Systems (MAS)

An agent is an entity (physical or virtual) that can perceive its environment, can act into it and can communicate with other agents. According to [Ferbert, 1999], an agent is autonomous, has goals and the capacity to achieve them, and it often lives within a Multi-agent System. A MAS has been defined as a weakly coupled network of agents that interact to solve problems that are beyond their own individual capabilities [Sycara, 1998]. More formally, a MAS is a tuple $< E, O, A, R, Op, L >$, where:

- E is the environment.
- O is the set of objects included in E.
- A is the set of agents and $A \subseteq O$.
- R is the set of relationships that link objects and agents in O.

- Op is the set of operations that allow agents to perceive, produce, consume, transform and manipulate objects.
- L is the set of universal laws that express the effects of the operations to that environment.

3.4.2.1 Uses of MAS

MAS have four main areas of applicability.

- **Problem solving**: MAS can be a good alternative to centralized problem solving. As it will be shown, this is a key feature that justifies their use throughout this thesis. If the nature of the problem to be solved is distributed, then MAS fit much better than traditional approaches.
- **Multi-agent simulation**: MAS allow creating artificial domains to simulate local behaviors in the labs. They have proven to be useful in diverse fields such as social sciences, education, or biology.
- **Artificial Worlds**: MAS can be used to describe interaction mechanisms and analize their impact globally.
- **Robotics**: A robot can be seen as a MAS in which each agent carries out some specific goal. By achieving small tasks, the global objective will be completed.

3.4.2.2 Communication in a MAS

The agents in an environment can perceive information both directly, when there is an exchange of messages among agents; and indirectly, when the agent itself senses the environment. This allows us to categorize the communication in terms of information flow.

On the other hand, the relationship among the agents can also be categorized as centralized when a hierarchy exists among agents where an agent has power over other subordinated agent; or decentralized when all the agents deal with each other in equal terms. This allows classifying the types of communication as it is shown in table 3.1.

In a centralized topology, the interaction can be driven by a supervisor agent that can either directly **command** the subordinated ones or modify the environment so it indirectly **restricts** their conduct.

In a decentralized topology, the interaction can be driven by direct **conversations** among all the agents, or indirectly when agents sense the changes in the environment caused by others. The later is the typical case when agents **compete** for limited resources that have to be shared.

		Agents Relationship	
		Centralized	Decentralized
Information Flow	Direct	Command	Conversation
	Indirect	Restriction	Competition

Table 3.1: Communication categories [Grimaldo, 2008]

3.4.2.3 Coordination in MAS

Since agents act autonomously, agents need to coordinate their communications to make sense out of them. Coordination is one the most important interactions agents have. [Bergenty and Ricci, 2002] identify three types of coordination:

- **Tuple centers**, messages are posted in to a centralized container as a sort of center that processes and manages the messages in a synchronized way.
- **Interaction protocols**, that are defined by means of finite state machines. Each state of the protocol corresponds to each of the states an agent can be in through the conversation. Some agent communication languages (ACL) have been developed like, e.g. KQML or FIPA ACL. The communication with interaction protocols is composed of *speech-act* messages that are asynchronously exchanged by the agents.
- **ACL semantics** which introduces communicative actions that influence the agent's plan to the set of actions standard in ACL speech-act messages.

3.4.3 The JADE framework

The use of MAS in real applications requires a complex infrastructure to allow creating the elements in the tuple $< E, O, A, R, Op, L >$ already mentioned earlier. While the objects (other than the agents) included in an environment and the rules L defining their behavior could directly be imported from the internal data model of the host application, the agents themselves, the environment (E) they live in and allow them to interact and coexist with other objects (O), the actions (A) they can perform, the relationships (R) established, and the agent's operations (Op) are components provided by the MAS. They involve a lot of effort when trying to develop them up to a certain level of quality. Fortunately, there are software packages that provide support for them and are available to integrate in an application. JADE is one of them. In fact, it is a well-known and *de facto* standard.

JADE is written in Java and it is a product of a research project of Italia Telecom labs. JADE provides easy-to-use support for creating agents that exist in an environment and communicate asynchronously using FIPA ACL messages. With JADE user can design custom interaction protocols that define how their communication is coordinated.

Since JADE was the choice for this research, the interaction among agents is protocol oriented, i.e. it uses **Interaction protocols**. On the other hand (and speaking in MAS terms strictly), as will be described in chapters 12 to 15, the communication approach chosen falls in the category of **Command** since an agent is in charge to conduct the conversation all the agents hold (centralized agent relationship) and the flow of information is direct; i.e., via FIPA ACL messages that are sent directly to the intended recipient.

Chapter 4
Background

Civil engineering industry acknowledges the problem of the projects overhead presented in Chapter 2 [Touran, 2003, Eastman et al., 2008, Mubarak and Means, 2012, DelPico, 2004] and [FMI/CMAA, 2005, FMI/CMAA, 2006, Clients, 1997]. In fact, this problem is so common that a part of the budget is simply reserved for contingencies, i.e. a part of the resources of the project is initially allocated only to cover uncertainties or, in clear language, the unforeseen. To a lesser extent, it is also acknowledged that a seamless and fluent interoperability also has a positive impact on the performance of each and every phase of the project. Historical and traditional companies tend to underestimate this impact based on the fact that they have succeeded in projects without changing their work flow. The difficulty on estimating this impact does not help in demonstrating the beneficial effect. After all, the same project is never performed more than once. With only one sample of each it turns out to be impossible making scientific comparisons in performances. The closest to a strict evaluation possible is to take a infrastructure project divided in parts where each part is developed using different paradigms. For instance, to take a large road project in which stretches of the road are developed by different companies with noticeable different level of commitment towards interoperability.

An indication supporting the interoperability positive impact could be the project of the construction of a road in the Økern area in Oslo [Holt and Resi, 2011] that was occurring at the time this thesis was written. Vianova Plan og Traffikk AS as a company committed to foster interoperability; and Norconsult[1] as a sample of company that delivers each discipline designs in an independent manner (i.e. the engineers do not interoperate but treat the project as independent subprojects). For the time being, in the current stage of the project, the constructor has claimed for issues regarding incomplete, incorrect or incoherent designs for a sum ranging from 0.5 to 1 million Norwegian Kroner, an amount derived only for the design expenses and that does

[1] http://www.norconsult.com

not include losses like delays caused by the disputes and redesigning, which could not be estimated yet but are probably some order of magnitude bigger. From this sum, none corresponds to Vianova Plan og Traffikk. In other words, Norconsult is the only company that will have to deal with it. This gives an impression on what is the impact a good interoperability has.

Therefore, there is a need for better tools that improve or even automate the synchronization problem.

4.1 Interoperability at Data Exchange Level

Civil Infrastructure design has traditionally relied, and is still relying, on paper drawings [Eastman et al., 2008]. It has been only to very recently that the construction design is gradually switching to computer models. A large part of the information still exists only in non-digital support. This is especially true for small municipalities that cannot invest in digitalizing their data. In words of Vicente Chorques, a Civil Engineer working on public constructions in Georgia under the Strategic Development Plan of the Georgian government, "much of the information simply does not exist, and when it does, it is never ready to be used out of the box". Another example comes from Andreas Haugbotn in Norway. Quoting Andreas own words "I always ask for data in digital form. I sometimes get it in a PDF document that contains scanned maps of the area. They call that digital".

In the process of digitalizing data, the data formats that were created evolved along the history around the concept of Feature that encodes the digital representation of an object. I.e., in most of the vector-based[2] formats, Features are the atomic data structure used to represent objects of the real world. How Features are defined and physically encoded differs from format to format, but a conceptual pattern can be detected that is followed by most of them. In this pattern, the

[2] Geospatial data can be either vector or raster based. Vector-based data is more suited for representing abstract, discrete or artificial concepts, such as buildings, roads, power lines, or even schematic representations of rivers, while raster data is suitable for representing continuous variables such as light reflection and frequencies typically corresponding to satellite images, noise emissions, or Earth's surface composition to only mention a few. This thesis is specifically focused in vector data formats because, as the reader might have already noticed, the design works make use of vector data mostly.

Feature is composed of the Geometry that describes the shape of the object (and it is typically points, lines, surfaces and sometimes arcs and solids, and combinations of them) and Attributes describe the Feature's properties qualitatively or quantitatively. Perhaps, the ultimate initiative to rigorously define this pattern can be found within the "ISO/TC 211 Geographic Information/Geomatics" standard approved in 2009 [ISO/TC 211, 2009].

When a project starts, there is a data gathering phase in which information about the 'existing situation' (sic) of the area of interest is collected. This information is composed of maps of the geology and land use of the working area, the previous installations that might be there, official documentation, and more. Once collected, the challenge to be faced is how to combine the myriad of formats the data are stored in. Some examples are GML, KML, Shapefile, CityGML, IFC, DXF, SOSI in Norway, or even LiDAR to mention only a few.

Andreas Haugbotn was part of the committee for the definition of the Norwegian mapping standard called SOSI (Samordnet Opplegg for Stedfestet Informasjon, often translated to English as Systematic Organization of Spatial Information). He argues that data exchange formats have always been a problem. The lack of transparency, incomplete or closed specifications, poor extensibility problems and so on, forced the definition of the SOSI format to cover the specific needs for Norway. With Andreas Haugbotn both being part of the SOSI standard committee and also partner in this research work, it makes sense taking the SOSI format history as an example, it is possible to discern the problems. Namely, 1) that formats are defined to cover the topics they are created for and therefore when transforming from one format to another some information is lost; and, 2) that when a data format is imported into a system, the import process depends on the extent to which the programmer who implemented such importer interpreted the specifications the format used for the exchange correctly.

It is worth noticing though that the problems discussed here are not focused on SOSI's specific technical issues but rather on general ones since any data exchange format other than SOSI will also suffer from them.

It is possible to find in the literature many materials dealing with interoperability at a data exchange level. However, the problem of the interoperability has not found a satisfactory solution yet despite the many articles and books dealing with data formats that are published every year. [El-Mekawy et al., 2012], [Zhang et al., 2012], and [van den Brink et al., 2012] are some recent examples dealing with CityGML, IFC, KML, Collada or VRML.

There are two ways data can be encoded. Data are either binary or text formats. Text formats are much more human readable. That is why the open formats tend to use text encoding in order to ease support for them in computer tools. In contrast, binary formats sacrifice this readability for reducing memory and read time since they are more compact. With a lesser readability, they are also preferred when a format is wanted to be closed and hardly supported by parties other than its authors.

Text	Binary
SOSI	DWG
IFC	Shapefile
GML	DGN
CityGML	LiDAR
DXF	
KML	

Table 4.1: Examples of Text data file formats and Binary data file formats

Table 4.1 shows a classification of some formats used frequently in Civil Engineering for mapping tasks (SOSI in Norway), detailed building modelling (IFC), city modeling (CityGML), general spatial modeling (GML, KML, DXF, DWG), Geographic Information Systems (Shapefile), drawings (DWG, DGN, DXF) or laser scans (LiDAR).

SOSI is a text file format whose standard version 1.0 was first published in 1987 when CAD tools were not widely used yet. By the time the SOSI format was being defined, most of the existing data was in paper support and the amount of it was much less compared to the present days. The fact that there was little data and mostly in paper made the process of data collection too slow, costly, and even not reusable from project to project. Hence, there was a need for digitalizing the paper drawings and map data. When the authorities wanted to figure out which file format to use, they found out that there was no perfect one; not even any clear candidate. It was not because they had a weak geometry support, or could not hold the data in some, perhaps also, obfuscated way. I. e. it was not because of lack of power -there were formats that could hold it already-. But what they were not suitable for was storing the meaning of the data. The way mapping data are collected in Norway showed to be difficult to adapt to the existing formats. Attributes in the features had some particularities that made them difficult to encode in the existing formats. Moreover, those formats were -and are- proprietary. So, it was not possible to tweak them nor to figure out a comfortable way to accommodate the data into them. Starting from technical reasons like, e.g., character encoding that was able to encode particularities of the Norwegian language necessary to characterize a feature, such as ‘ø’, ‘æ’, or ‘å’, characters which were simply not supported by the existing formats and would force questionable workarounds, to more structural-related reasons. E.g. data structures to encode buildings (Bygg feature) with all the features it is composed of like the upper most roof line (Mønelinje feature), building outer limits (Takkant feature) as well as other several hundreds of specific SOSI features. And also their corresponding set of attributes with specific types which, sometimes, were Norwegian-only. Thus, existing formats were found to be too difficult to adapt or to support the needs. All these inconveniences (i.e. closed formats with no possibility to fix, poor support for concrete “local needs”...) paid off the effort to create a national standard like SOSI.

Whether it was worth or not to define a new format can be figured out by comparing Norway’s situation to other Nordic countries. When gathering data in Norway, there is a good chance of getting it in a more or less ready-to-use format. Unlike Norway, in Sweden and Finland there is no such a standard. As a consequence, each city encodes it in its particular way. It is easy to see that this is a problem each time there is a need for data.

Similarly to Norway, Denmark defined its own format in 1983. It is DSFL Dansk Selskab for Fotogrammetri og Landmåling (or Danish Association for Photogrammetry and Surveying in English) but it is less developed than SOSI.

4.1.1 The need for a paradigm switch for full interoperability at a data exchange level

Any experienced IT software engineer will argue that the problems mentioned above appear in virtually all existing data exchange formats. Data exchange remains an issue (and a big one when the support media used is still in paper). The result is that there is an uncountable myriad of different data formats coming from both private companies and public initiatives; sometimes with the same commercial goals than AutoDesk® (because, as DWG, they have also closed specifications) and sometimes aimed to increase data exchange interoperability. Examples of the first group could be Bentley MicroStation® [Bentley, 2012] or SolidWorks® [SolidWorks, 2012]. In the latter group it is possible to find initiatives like GML and its derivative CityGML [Kolbe et al., 2005] or Industrial Foundation Classes (IFC) [Adachi et al., 2003].

Since today the work of merging data is done by hand through sharing files among the project's participants, problems arise when the data exchanged are not coming from the same software package. Naturally, every design team has specific expertise which directly translates to different needs. Hence, as the interviews showed (see section 5.2), industry creates specific software packages which, in turn, have an internal, most likely incompatible, data model specific for those needs. This lack of compatibility forces the definition and creation of file formats that are used for distributing the data among all the stakeholders in the project.

The experts consulted in this research point out that AutoDesk™'s DWG tends to be the de facto standard of file format data exchange. This is due to historical reasons. AutoDesk™ (www.autodesk.com) [Dix et al., 2008] has successfully introduced its products worldwide. AutoCAD® (www.autodesk.com) [Dix et al., 2008], probably one of the most known CAD system, is one of them and it is based on this file format. Having a big portion of the market share, it forced AutoCAD® competitors to support it. In a way, this fact helped the engineers to interoperate at a data exchange level by providing a kind of *lingua franca*. But it is worth mentioning that using DWG comes at a price:

First, DWG's specifications are not public. Furthermore, AutoDesk uses this fact as a means to keep its dominant position against its competitors. Thus any non-AutoDesk implementation is the result of a very difficult and laborious process of reverse engineering. It is true that difficult does not mean impossible. There are many attempts to do it with more or less success. Possibly, the most successful independent implementation for DWG is the DWGDirect (formerly openDWG) libraries of the Open Design Alliance [OpenDesignAlliance, 2012] which by reverse-engineering provide read and write support at a high level of quality.

Finally, DWG is geared for AutoDesk's products. Inevitably, it will not have native support for other purposes than those AutoDesk's products have. For instance, imagine that we would be designing a road network in Norway. At the time this thesis is being written, there is no such a product that covers the specific Norwegian regulation on road design. Consequently, it is not reasonable to expect DWG to be able to handle it out of the box. As a general CAD-oriented format, most of the effort is put in supporting advanced geometry but not in particular expertise needs because it would introduce a complexity that is not necessary in the majority of its use cases. Thus, if such a need exists it necessarily implies the creation of another data format supporting it.

4.2 Interoperability at Distributed Behavior Level

Interoperability does not only refer to the ability of exchanging information. It has another sense which remains less explored than the pure data exchange. In collaborative works, it can also refer to the degree of the capability of a heterogeneous team to work together seamlessly. The sum of each individual efforts and skills is what shapes the strength of the team.

Civil Infrastructure projects heavily rely on this kind of interoperability. It can be seen as the glue that provides the group of engineers with the team structure required to behave as a whole. Unfortunately, in the Civil Infrastructure industry, and from a Computer Science point of view, this type of interoperability is not yet considered satisfactory. According to the interviews performed (see section 3.1 on page 29), the life cycle of the design process follows a pattern depicted in Figure 4.1. It is an iterative process in which each team regularly delivers its designs and a person or group of persons collect all those designs. This person, or set of persons, validates the designs and decides whether they are correct or not by identifying and agreeing on as many errors as possible. When all the partial errors have been detected and solved, a next iteration starts until the process is considered as finished. This error-solving problem is a normal decision making negotiation in which all teams reach an agreement regarding every issue found. The reason why this remains a manual process is twofold: (1) the tools available for each discipline may not have the same internal data model and (2) because it is simply very difficult to automate it. Unfortunately, this process can hardly guarantee that all errors will be detected. Humans are particularly well suited to detect patterns, but have big difficulties processing large amounts of data. It is particularly difficult if the data to be processed is in 2D or paper plans-based. Thus, even though the current tools make this task easier by, for instance, using realistic models instead of sets of plans, it is easy to see that there is a need for more advanced tools.

Eng. Torbjørn Tveiten and Eng. Andreas Haugbotn describe the collaborative process implemented in Vianova Plan og Trafikk, which is considered one of the most effective ones, in the "Handbook on producing design data in interdisciplinary design work" (refer to Appendix A in page 215 for a sample). This is a compendium of best practices that tries to standardize the process of creating distributed design data by "describing the procedures and methodology for the establishment and maintenance of Virtual Reality (VR) models, so-called 'engineering models' ". As expected, the handbook provides an extensive description on how the data is to be exchanged (whether a data repository is available for a given project or just file exchange is possible), and how the data is to be encoded (e.g. how a power line will be expressed, or which file format will be used).

But it also takes the standardization a step further. Firstly, several engineering disciplines are pre-identified (see Table 4.2).

Also, the members of the project and their roles are defined (see Table 4.3). Every role is assigned to a member. According to the member's skills and expertise, he takes one role or another. As can be ob-

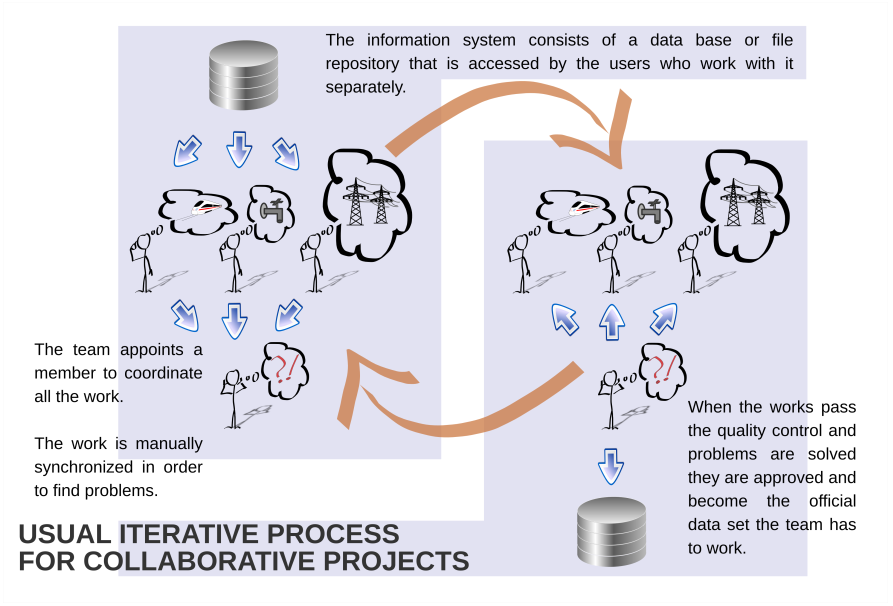

The information system consists of a data base or file repository that is accessed by the users who work with it separately.
The team appoints a member to coordinate all the work.
The work is manually synchronized in order to find problems.
When the works pass the quality control and problems are solved they are approved and become the official data set the team has to work.
USUAL ITERATIVE PROCESS FOR COLLABORATIVE PROJECTS

Fig. 4.1: Live cycle of a Collaborative Civil Infrastructure Project design.

served in Table 4.3, a member can take more than one role and vice versa, if the size of the project demands more resources for any of the roles. Tables 4.2 and 4.3 have been extracted from the sample of the Handbook in a project carried out by Vianova and other partners consisting on the development of road and railroad communications in Kolsås area, near Oslo (Norway). In this sample project Morten Simonsen takes the role of the model manager for the 3D model. It means that he is in charge of collecting the designs coming from the different disciplines and combine them into the final model that will be used for quality checking. Rune Rian has two roles Road and Markup, i. e. he draws the geometry and path of the road to be and defines the lanes it will consist of as well as the auxiliary in-going and out-going access lanes. The Structure role in this project is carried out by another company called Aas-Jakobsen and it is performed by two engineers: Jone Stangborli and Jørn Ola Sørensen. They will design bridges and other constructions supporting the road and any of its accessory installations. The Geotechnical analysis is performed by Ingunn Veimo from the Geovita company and her task consists of cataloging the type of land under the road. Either if it is rocky, sandy, etc. or whether it has

Discipline	**Description**
Road	Geometric design of the road body, requirements for material selection and construction work. Design work should follow the given manuals from the Norwegian Road Authorities and requirements provided in those based on traffic numbers/veg class.
Railing (train)	Similar to the road, but the train has much stricter requirements for horizontal and vertical geometry.
Geotechnical	Geotechnics/geology provides the road planner with knowledge of what the ground consist of and what the rock material is. This affects the design of road/rail with such as frost protection and mass replacement. During the design work the geotechnics deliver stability assessments and provide the requirements of sheet pile curtain and stabilization.
Water (supply, sewer, drainage)	Draws the new drainage system to accommodate the spill water and drainage for the new road, but also include the surrounding water situation. Tunnels also require water supply of fire water in some cases. The design work should also take care of existing facilities for water and sewage, and the projection of the new system for water and sewage.
Signs and road marking	The road role defines the design parameters of plate size, position, and content. Names of places have their own authentication levels and are not controlled by the design-workers. Road painting includes all paint marking on the road area and is given from the road class and geometry.
Landscaping/Environmental	Vegetation after digging more desc
Structural/Construction	This discipline includes everything that will be built in concrete. Construction covers aspects like bridges and retaining walls, but also smaller constructions as manholes if casted on site. An important aspect is the calculation of strength for the construction.
Electrical	Electricity for lighting, variable signs and control power to trafic control.
Architectural	Design of buildings and facilities

Table 4.2: Decomposition of Civil Engineering in disciplines according to the ViaNova Plan og Traffikk's Handbook on producing design data in interdisciplinary design work. Extracted from a project carried out at Vianova Plan og Traffikk at the Kolsås area in Bærum, Norway. (Source: Vianova Plan og Traffikk, AS)

or had some installation in the past. Water Supply and Drainage role are performed by Jon Erling Einarsen from Vianova Plan og Traffikk and Rune Johnsrud from Vianova Systems who are in charge that the canalization of the water is correctly handled. Anders Pedersen from Electro Nova and Benoni Nera from ECT are performing the Electrical role for the road and the railroad (denoted by plants in this case) respectively. The Electrical role is in charge of designing the power lines of the construction and connecting it to the general network. Magnus Greni from Grindaker and Gisle Totland from LINK mark are in the Landscaping role which ensures that the construction integrates seamlessly in the environment it is built. Aspects like wildlife or forest management are considered by this role. And last but not least, Brede Henriksen is the architect of the project.

Role	**Company**	**Person**
Project manager 3D	ViaNova Plan and Traffic	Torbjørn Tveiten
Model Manager	ViaNova Plan and Traffic	Morten Simonsen
Performing geometry road and path K22-K25	ViaNova Plan and Traffic	Rune Rian
Performing markup K22-K25	ViaNova Plan and Traffic	Rune Rian
Performing structures K22 and K23	Aas-Jakobsen	Jone Stangborli
Performing structures K24 and K25	Aas-Jakobsen	Jørn Ola Sørensen
Performing geotechnical K22-K25	Geovita	Ingunn Veimo
Performing VA and drainage K22	ViaNova Plan and Traffic	Jon Erling Einarsen
Performing VA and drainage K23-K25	Vianova Systems	Rune Johnsrud
Performing electrical K22-K25	Electro Nova	Anders Pedersen
KL-performing plants K22-K25	ECT	Benoni Nera
Performing Landscaping K22	Grindaker	Magnus Greni
Performing Landscaping K23-K25	LINK mark	Gisle Totland
Performing Architect K22-K2	Arne Henriksen Architects	Brede Henriksen

Table 4.3: Example of the structure of Engineering Roles of a road construction project in Kolsås, Norway. (Source: Vianova Plan og Traffikk, AS)

Finally, the handbook defines a “time and refresh rate” for all members to synchronize their work as well as a data exchange policy for delivering the changes.

To some extent, every Civil Infrastructure project applies partially and informally the procedures this methodology defines. The advantage of this methodology is that it formalizes the interaction in terms of the distributed work at the design process. The roles and the way the designs will be delivered are fully agreed at the project start and there are periodical and fixed meetings to align works and make decisions when required. Hence, this formalization allows Vianova Plan og Traffikk to have a competitive advantage.

4.2.1 Building Information Modeling (BIM)

The term BIM can refer to an activity (and then it stands for Building Information Model*ing*) or an object (and then it stands for Building Information Model*s*). The BIM term was introduced for first time in [Van Nederveen and Tolman, 1992]. Under BIM lies the latest trend towards teamwork interoperability in the building construction industry. With BIM there is a big effort towards the abandonment of 2D and paper drawings as the main source of information in favor of live 3D models[3] based on the assumption that these models are easier to understand and less prone to errors. A BIM system as it is defined by Eastman [Eastman et al., 2008] is a technology and an associated set of processes to produce, communicate, and analyze 'building models' that has:

i. Building Components; objects that know what they are and can be associated with computable graphical and data attributes and parametric rules.
ii. Components that include data describing how they behave. This includes parameters describing, e.g., how much tension a piece can stand but also, e.g., how much a given material costs per volume unit and so on.
iii. Capability for multiple (and also different) views of the model.
iv. Consistent and not redundant data that is represented in all views of the components.
v. Coordinated data such that all the views of a model are represented in a coordinated way.

Also, Eastman defines BIM by means of what it is not. Thus, a tool does not use BIM technology if it has (any of):

a. Viewers with only 3D data and no object attributes, i. e. no intelligence at object level.
b. Models with no support for behavior; even if they define objects, if those objects do not adjust themselves by using parametric intelligence.

[3] Notice that it does not imply that 2D or paper drawings are not allowed anymore but that they will be produced out of the 3D models instead of being *the model*.

c. Models that are composed of 2D reference files that must be combined to define the building, because it is impossible to ensure that the resulting 3D model will be feasible.
d. Models that allow changes in one view that are not automatically reflected in other views.

As can be observed, BIM has a strong focus on ensuring that the model engineers work with is always the last version of it and that this model is instantly propagated to every workplace as soon as it is changed by any of the members of the team.

The expected benefits of the general introduction of BIM in the Civil Engineering Industry are various (to a wide extend these expectations are assumed true even though it is impossible to provide accurate data) [Eastman et al., 2008]:

1. It is possible to assess the feasibility because an approximate model built into and linked to a costs database can be of a big help for fast estimation of costs in real time as the design progresses; to find out that a particular design is significantly over the budget after a considerable amount of time and effort has been expended is wasteful.
2. A schematic model prior to a detailed building model allows for a careful evaluation of the proposed design.
3. Earlier and more accurate visualizations of a design; the 3D model is designed directly instead of built up from 2D drawings.
4. Automatic low-level corrections when changes are made to the design.
5. Generate accurate and consistent 2D drawings at any stage of the design.
6. Earlier collaboration on multiple Design disciplines.
7. Easily check against the design intent.
8. Extra costs estimates during the Design phase; if a costs database is available and linked to the model, costs estimation can come real-time when changes are applied to the model. In other words, it is possible to monitor whether the project remains under the acceptable cost limits upon an unexpected change to it.
9. Discover design errors and omissions before construction (conflict detection).
10. React quickly to design or site problems.

Of special interest to this thesis are points 4 and 9 regarding the corrections that are automatically performed and the discovery of design errors and omissions. Though important, unfortunately the other points on this list are out of scope of this thesis due to limitations in time and resources. Point 4 explicitly mentions the adjective low-level to clearly delimit the intelligence capability. Low-level corrections are often exemplified by “windows that upon installed in a wall automatically create the whole in such wall” or “elbows and tees that are automatically inserted upon pipe connection creation”. This “low-level” intelligence is derived from the Building Components of properties i) and ii), and it is the approach taken by IFC (actually, IFC is very central in Eastman et al’s view of BIM) which provides with a set of standard components for which simple behaviors implementing this low-level corrections are included. These behaviors need relationships among features to be previously defined. On the other hand, point 9) claims that BIM eases the discovery of errors and omissions. This claim is also derived from the fact that all engineers are looking at the last version of the global design regardless when they do it and where they do it from. Thus, errors caused by incoherent data or outdated designs unintentionally mixed are mitigated.

As it will be shown in Chapter 10, there is a need for an intelligent mechanism that let a system to automatically weave relationships in a model that conform its knowledge in order to have powerful tools that detect errors. Relationships among features are potentially subjective because one member of a project might be interested in some of them while others can completely disregard them. Unfortunately, attempting to manually establish all the relationships in a model soon becomes too tedious because they can be virtually infinite. Plus, their usefulness are not really evident to the eyes of an engineer designing the model who is accustomed to pay most of his attention to other, more obvious, aspects like the geometry or the attributes of single features. Furthermore, there is no natural or standardized way to graphically represent those relationships so that it is easy to find out when some of them are missing or incorrect in a model.

Similarly, a method for applying high-level corrections is still missing because an error might be caused by many factors that are not necessarily supported a priori and also because, unlike the low-level corrections, high-level corrections are those that involve conflicts of interest managing and the evaluation of subjective alternatives. In other words, these corrections have much more social interactions than the low-level ones which, at least conceptually, can be solved with recipe-like pre-established procedures.

4.2.2 Multi-agent Systems

In Computer Science there is a paradigm that is very suitable for approaching the problem at hand. It is Multi-agent Systems (MAS) [Weiss, 2000]. MAS are well-suited tool for this system because they feature a distributed, autonomous (or semiautonomous), social, active, reactive and adaptive behavior that it is not present in other paradigms.

It is, however, particularly surprising that MAS, a cornerstone in this thesis, are still in their beginnings when they are applied to Civil Engineering distributed design provided their theoretical potential as well as the fact that they are a vibrant research field. Certainly, there are some works that try to model social behavior of the overall process of Civil Engineering. Examples are ADLIB [Ugwu et al., 2005] a system devoted to the design of light industrial buildings in which the the participants negotiate the characteristics of a building, or MASCOT [Ren et al., 2003] where the main endeavour was put into modeling negotiations for the distribution of risk. Other pure abstract theoretical models have also been proposed to model a complex system for collaborative design work. However, these models tend to be too abstract or too specific, or both. Too abstract in the solution proposed so it is difficult to bring them to a really implementable solution, but also too specific in the problems they deal with compared to the long spectrum of problems any Civil Infrastructure project faces. For instance, the model presented by Peña-Mora [Peña-Mora and Wang, 1998] is actually a methodology that deals with the decision of what to do with the concrete case of an old installation discovered during the excavation of a construction. The decision is negotiated among an Architect, an Engineer and a Constructor (A/E/C) by means of the concept of utility functions. As it will be seen in Chapter 11 the utility function approach is too abstract to be directly applied. Schellenbach and Denck [Schnellenbach and Denk, 2002] go a step further and provide a concrete implementation of a bidding marketplace in the A/E/C domain. Despite these works show some attempt and similarities in the way MAS systems are used in this thesis, it is hardly possible to find real products that go beyond the pure laboratory premises. In fact, to the author's knowledge this has not happened yet. It was necessary to find out why this happened. In the some cases, even the authors themselves mention difficulties, in deploying their systems [Ren and Anumba, 2004].

4.3 Summary

In Civil Infrastructure the designs are performed asynchronously by each team and it is periodically synchronized. Unfortunately, there are no tools to address the process depicted by Figure 4.1. The lack of advanced tools forces the engineers to perform their synchronization and design alignments in a manual manner. It is true that a manual process does not necessarily imply a low quality process. Considering the complexity, it can be argued that a good level of quality is already achieved. After all, designs are normally accepted by the client at the end and for this to happen a certain quality level needs to exist. However, a manual process is more sensitive to external influences, mistakes and other factors that should be kept under control in order to avoid overruns derived from design errors or omissions. In this chapter the state of the art regarding the two topics of this thesis has been presented.

Next chapter is a more in-depth study about the engineers daily work approached ethnomethodologically. Some of the conclusions extracted from that study have been already implicitly sketched in this chapter as, by definition, they are part of the state of the art. However, to the author's knowledge, there are no studies dealing with Civil Infrastructure interoperability that enjoyed a sufficient level of immersion as ethnography offers. Thus, they are presented as one of the contributions of this thesis.

Chapter 5
Ethnographic Studies

5.1 Ethnography

An ethnomethodological study was carried out in Vianova Plan og Traffikk AS to fully understand the daily work in a civil engineering company. The outcome of this study is a report that starts explaining how the study was planned, what the sampling strategy was, how the access to the individuals of the study was gained, and what the observation policies applied were. Then, the report summarizes the knowledge obtained during the field work that is relevant for the research and ends with providing the conclusions obtained.

5.1.1 Planning

The ethnographic research had a simple goal: to understand how engineers manage the collaborative projects (how they communicate, interact and decide) and the implications this activity has for designing new kind of technologies to improve interoperability. It was important to state this from the beginning to avoid losing focus.

In particular, this research is interested in identifying the problems they face for which the traditional and also the cutting-edge technologies have not provided a satisfactory solution.

5.1.2 Sampling

According to Bernard [Bernard, 1994], there are two categories of sampling: probabilistic and non-probabilistic sampling.

The probability sampling is based on the induction or generalization from research samples to a target population. The degree of accuracy is measured using probabilistic estimations and values. The samples are randomly selected and normally require a rather big sample size.

The non-probabilistic sampling demands less sampling size but deeper immersion. Most of the research studies are carried out using non-probabilistic sampling. Green [Green, 2001] demonstrates that smaller non-randomly selected samples can produce the same results as large-scale survey research for 1/100 of the costs of that for probability.

A non-probabilistic sampling strategy was selected in this study because, as it will be explained in the next section, this researcher was in direct contact, spatially and temporarily, with the engineers. Since studying a concrete set of individuals could result in overfitting[1] the research to such concrete case, some interviewing to external professionals not belonging to the same community under the ethnographic study were also carried out. The aim is to have a sort of "group of control" to validate the conclusions of the research in a more general manner. Otherwise, it could be argued that the conclusions achieved were tightly tailored in excess to the studied group.

5.1.3 Gaining Access

Gaining access is usually the most difficult task for an ethnographer. But the best information flow comes from the closest relationship with the studied subjects. While in the origins of Ethnography this involved establishing friendship with the members of the society, nowadays this has shifted to a more contractual approach. Today, everything needs to be agreed, specified and signed by both the ethnographer and the participant parties. This research did not require to sign any contract or agreement since it was carried out by one of the employees. So, issues like ethical, copyright or other kind of rights were avoided automatically. In addition, the researcher was a new employee without any kind of contact with the civil engineering world and the company. Therefore, the observation did not suffer from any previously, potentially biased, point of view.

[1] "Overfitting" as in the statistical, data mining or artificial intelligence sense; referring to the fact approximating too much to concrete samples might compromise generality.

5.1.4 Observation

Observation was conducted during the first weeks of the research period. During this time, an *observer-participant* role was taken when accompanying the engineers in their work. In other words, it was preferred to be as unobtrusive as possible. Many reasons exist for this. Especially, the fact that the meetings were normally held in Norwegian which is not a language the researcher speaks, and by not being a Civil Engineer the researcher was not competent enough for taking an active role in the meetings. Fortunately, the language problem could be easily solved given the close involvement this research had with Engineer Andreas Haugbotn who was always willing to help in those moments it was not possible to follow what was being observed.

Initially the goal of the observations was simply getting familiar with the way engineers work. Very soon it became clear that the most valuable source of information was the regular meetings the engineers have. In those meetings they talk about what they have done, what they are planning to do next, and is the time for synchronizing. With these information exchanges it is very easy to detect the problems they face and how they face them. Hence, the observation task soon focused on those meetings.

5.1.5 Field work

The study starts at the beginning of a project when it is contracted and follows with a period of time in which it is studied how the engineers do in their working settings. It distinguishes several phases that are explained below that approximately correspond with the initial phases of Data Gathering and Design of the Civil Engineering process (introduced earlier, see figure 1.1 in page 12). In order to avoid word clashing with the previous use of the term “phase”, the term Job will be used within this section instead. So, the first two Jobs correspond to the Data Gathering phase while the last two are Job performed in the Design phase. As it can be expected, Construction and Maintenance phases were not covered in depth by this ethnographic study. Special attention is paid to the job with the biggest body of work: The design and the planning, that is, when the engineers start designing and creating objects that don't exist after the initial data is gathered and processed.

Job 1.- Gathering the data.

In the fall of 2010, an engineering project consisting on widening the E6 Omkjøringveg Nord road has been granted to the company that won a public bid. The goal was to increase the road traffic capacity. Participating in the bidding consists of delivering a proposal with a relatively low, but still credible, level of detail that includes a description and a study of the costs. The authorities responsible of the bidding select the proposal that fits best amongst all the proposals. The selection process is not important for us since it is external to the organization.

The first task is to prepare the data that the project is going to be based on. This consists of collecting digital terrain models and digital maps of the existing constructions (other roads, houses, buildings, etc.) and installations (power lines, gas pipes, etc.). Although this is normally performed by more than one person the work is not very structured. There is a list of data to be collected that is more or less standard and fixed. For example, collecting the power line maps is a task that implies asking the authorities about the official maps or, if they don't have them, to the electricity companies that have infrastructure in the area. Who takes this responsibility obeys more to practical reasons (for instance if there is a member of the team who has done it before, or knows somebody inside the other organizations, etc.) rather than to structured ones.

Job 2.- Adapting the data.

Unfortunately, data often comes incomplete and incompatible. It may consist of digital maps, plans (paper), or simply reports that have to be processed into a common data model. The data model to be used is basically a matter of tradition within the company. At some point, probably in past projects, it was decided to acquire licenses for a given set of software tools which implicitly defined the data format to be used. There is a general tendency to use AutoCAD's de facto standard format DWG files because it is a well-known CAD file format which has roots back to the 80's [Dix et al., 2008]. Even though it is not an official standard it became a de facto standard at some point. Other formats widely used in Vianova are SOSI and Quadri (Vianova's proprietary format)[2].

So, all the data is compiled into CAD files in such a way that it is possible to combine them more or less easily.

Job 3.- Design and plan: a multidisciplinary work.

At this point the project gets structured in a rather flat hierarchy with only two levels of responsibility. A Project Manager (PM) is nominated who is in charge of making high level decisions. One step below in the hierarchy are the working teams (see figure 5.1). Each team specializes in one discipline. Nine teams are identified: Landscaping, Geologists, Road, Road Marking/Traffic Signs, Railing, Electricity, Architectural, Structural/Construction, and Water/Sewer. Members of the teams are sorted according to the roles (see also table 4.3). An engineer with expertise in electricity can take a role in Traffic Signs and in Electricity, i. e. a person may take more than one role. This is typical when the roles are somehow "less important". The Road engineering team is likely to be composed by several fully-dedicated engineers as the Road discipline is central in a road construction project.

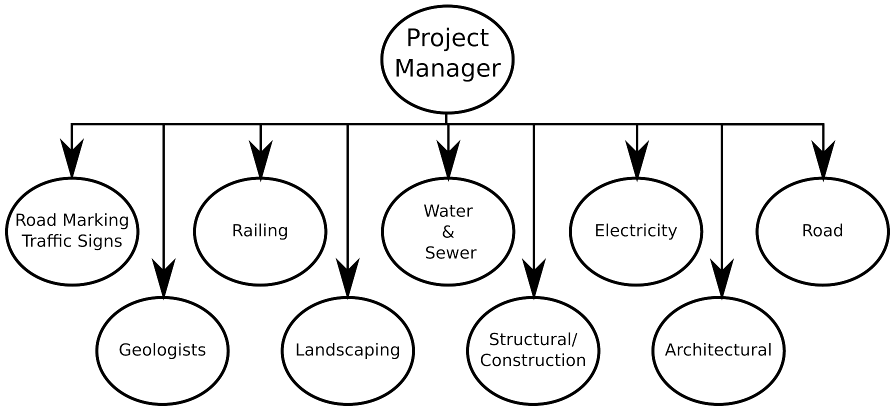


Fig. 5.1: Hierarchy of Civil Engineering design disciplines.

There are no formal rules to classify the discipline's order in importance. As engineers say, "it is culturally assumed or common-sense

[2] Unfortunately, due to copyright issues, it was not possible to find a reference to Quadri's specifications that could be safely included in this thesis

and efficiency-driven"; i.e., a sort of the-most-costly-discipline-first-like rule. For instance, The Traffic Signs team adapts their work to what the Road team decides and so do the Electricity. Landscaping can leverage in the decisions of Road team although this is solely to the Road's decision to accept their directions. Very few directions are taken as "red lines" that can't be crossed. An example of this could be when the Road team is designing a road in a bird nesting protected area. But in most of the cases, Landscaping directions are taken as mere recommendations. If there is a bridge to be constructed, the Bridge team has a strong word as bridges have big constraints due to costs and construction constraints in where they can be built. Since the road must be connected to the bridge, Road and Bridge teams need to completely align their work.

In general, the teams work separately with some inter-team interaction. Engineers within the same team share impressions, talk about possible ways to solve local problems, issue opinions. This interaction is facilitated by the spatial proximity. Team members tend to arrange their work places next to each other in order to speed-up the informal communication. This lack of barriers does not apply to inter-teams work. Even within the same room, some barriers exist intrinsic to the nature of the design work. Since the project involves several disciplines it burdens the work with different data models that need to be combined. Unlike intra-team interaction where sharing knowledge and opinions that easily map to a concrete function call in the tool being used, inter-team interaction requires an agreement to adapt every model in a coherent way. This is one of the most critical task. During the project life the teams meet twice a week in a meeting room to align their works in order to avoid deviations. When doing so, they discover what they call sometimes collisions and sometimes clashes. Since the difference between them is subtle and does not change the fact that both represent a problem to be solved, we will use collision indistinctly.

A collision pops up when something that was correct in any of the, for instance, road models separately becomes incorrect when the models are merged into one. For instance, one can think that a sewer conduction under a road designed by Water and Sewer team becomes a collision because the Electricity engineers installed a power line under it. In other words Water and Sewer team collides with the intentions of another team. It is a collision because there is a rule which specifically says that sewer will always be on the lowest level in order to, in case of leakage, avoid contaminating anything. Before merging both models, everything was right in each of them since the conduction and the power line were not coexisting yet together. But the model resulting of the combination of them two obviously had a problem.

These collisions are discovered in the model by navigating it in a Virtual Reality environment showcase on a big screen while everybody pays close attention. Vianova Plan og Traffikk has rooms especially prepared for this task with a big table where engineers meet. They attend the meetings equipped with their notes and designs. The room has a big screen where the complete 3D model is shown and presented. Engineers expose their opinion, clarify doubts and find problems. This manual process is normally enough but sometimes a collision is not detected and remains all the way in the project's design lifetime. On the other hand, when the collision is detected teams discuss a workaround that suits everybody's needs [3].

When a collision happens between teams with lower importance (like Traffic Signs, Water and Sewer...) the solution is normally delegated to their own discretion as long as it does not affect the works performed by more important teams (Road, Bridge...). In any case, they need to report if this change has any impact in the budget.

If a collision happens between at least one primary team, then all the teams have to reach an agreement as the solution probably affects every team and has a sensible impact on the budget.

The solutions are generated as a result of a bargaining of proposals between the different disciplines. The Project Manager acts as a judge that arbitrates the negotiation and watches over the fulfillment of the project's objectives and the budget. When any of those two aspects are compromised the Project Manager decides what the measures taken for all contingencies are.

This process iterates as the project evolves until the work is approved both by the Project Manager and the clients (most likely the public administration). Consistency checks are performed to ensure the project is not deviating from what is expected. Besides the design conflicts, issues like the material supply chain are also considered.

[3] This paragraph is a very crucial finding that will be the foundation of the approach of the system proposed in Part II of this thesis.

These include checking that the costs of the materials are still between the expected thresholds and they can be available by the time they will be needed. This is important because the Civil Infrastructure projects are long-term lived and aspects like international prices of raw material may vary substantially; or the supplier providing them cannot deliver on the time it was agreed due to some particular unexpected reason. If necessary, countermeasures are taken to overcome these problems. The Design phase spanned from December 2010 until May of 2011.

Job 4.- Closing and delivering.

When the project problems have been filtered out to a level in which it is acceptable, the project is closed. A report describing all the works done and the models of the constructions are delivered to the authorities. The idea is that for the next time some works are going to be done in the area this data can be reused.

From now on, the construction work is carried out by the contractor. It may happen, and in fact it does, that the project is not 100% correct and the contractor cannot follow it literally due to some unexpected conditions. As an extreme example, the geological cartography could had have some inaccuracies and the land that was supposed to be rock and strong enough to support the foundations is rock only the first 2 or 3 meters and underneath it is only sand. This could compromise the structure. In such cases (and of course after cleaning responsibilities) the normal way to proceed is to create a whole new project. On the other hand, if the problem is not really important, the planning company can decide to assist the contractor to fix it. This last situation is quite frequent, actually. And, in fact, it is calculated in the costs of the project. They compute the material costs, the salaries of the workers, the machinery rental or purchasing, etc... and also uncertainties. The "uncertainties" are a part of the budget: a reserve for contingencies that has no fixed destination but helps protecting the project viability from situations like the explained above.

5.1.6 Reflections on the work nature

Jobs 1 and 2 have a strong focus on creating the working environment in which the engineers will design afterwards. An initial model of the area is created and a system for the data distribution is agreed upon so that the engineers have an as-easy-as-possible access to the data. The reasons that influence the system used are basically “to use the tools that the companies are more familiar with”. Thus, the data is converted and edited to the known formats and stored in certain previously agreed places. It is assumed that the data is not complete and that it will need to be preprocessed each time it needs to be used in a different part of the project. This causes a burden in the project due to the interoperability problems derived of data exchange. The problem of the data format is extensively covered in Part I of this thesis.

Job 3 is a pure creative work. Enumerating all the activities performed in this aspect in a detailed manner would require a lot of pages, if such a thing could be even possible. Because there aren’t two identical places on Earth, it is probably impossible to find two identical projects. Therefore, when the tasks are defined and assigned to team members, they are specified in a high level or even informal way. The tasks take the form of “we are going to build a roundabout to remove this crossing because several car crashes were reported”, and then the teams contribute with their know-how to design it. The sum of all the know-hows builds up a distributed cognition that is bigger than putting individual cognitions separately. The organization structure is almost flat where every team makes their own decisions and others rely on their competence. Only when an issue appears they ask for help from the Project Manager for a workaround.

However, there are cases in which it is possible to detect patterns. These cases refer to situations that have occurred already in the past. Consider the following example: an illuminated road is being widened. When making the road surface bigger, a light pole is standing on the road. Several solutions are possible; for example, relocate them so they move with the road, install new models of light pole because the cost is going to be similar, or whatever other option. The engineers decide what to do. The way they could proceed with the remaining light poles could be the same or could be “we do the same for all the light poles that are now in the middle”. The association with “do the same as last time” with the actual tasks is straightforward for humans, but is extremely difficult to capture formally so that it can be automated. Anyway, it is a desirable feature that would save manual work while at the same time would allow producing better tools for the engineers if it were possible to be implemented. Part II of this thesis deals with how a system could replicate the process of detecting and solving design conflicts.

5.2 Interviewing

As previously described, interviews were conducted to contrast the conclusions extracted. The interviews were designed to be informal and in a relaxed ambient to let the interviewed feel comfortable. The interviews consisted of short questions which only aim to drive the interview to ensure the relevant topics are covered. With short questions, the risk of transmitting implicit information that potentially shapes the answer is minimized. In the interviews the questions were intrinsically the same even though they might have been formulated differently. In fact, the structure of the interviews is the same. After a small introduction as a warm up, they converge and focus on the interoperability aspects. Naturally, since there are no written questionnaires, some wording differences appear. Nevertheless, they did not prevent the researcher to cover the topics he was interested in. Finally, it has to be said that the interviews were held in Catalan and Spanish and translated afterwards to English by the researcher. Since both are languages the researcher is native in, the amount of information that could be lost in the translation can be considered negligible.

A complete transcription of the interviews can be found in Appendix B (page 225). The conclusions on the interviews are exposed below.

5.2.1 Conclusions on Interview #1: Vicente Chorques

Several weaknesses can be detected from the information kindly provided by Vicente. Basically all the errors come from the works that are made in a manual fashion.

In the pre-computer era, work has been documented in formats that are hard to maintain and to study.

1. Paper documents and plans are hard to understand since it is always a subjective part in the way the things are made that differs from person to person. So this may be a source of problems (misunderstandings, personal interpretations).
2. Conflict detection is purely a manual task.

3. Material documentation gets lost in the amount of files stored in the system with a rather primitive structure which could make it difficult to remember where they were.

In the computer era:

1. The complexity of the projects has to do with the amount of models that can be applied to each element. Not all the models are supported by one software solution but there are solutions specialized in one concrete subset and they export their results to AutoCAD files.
2. Each team in the project is in charge of a subset of tasks that use a supporting software tool that may differ from the other tools used by other teams.
3. They exchange works by means of AutoCAD files stored in a drive in the network.
4. Conflict detection and solving is still a manual work and relies on the competence of the team members.

5.2.2 *Conclusions on Interview # 2: Raúl Néstor Rodríguez Fajardo*

It has to be mentioned that some parts of this interview are omitted. They are the parts where we talked about the budget calculation of the projects with contingency costs. Raúl never worked in this and his experience and his knowledge regarding this was, admittedly, not extensive enough. His wish was not to include them and conclusions in this topic are not considered. Below are the conclusions extracted from this interview:

- There is a computer infrastructure, but it is mainly intended to give support for file sending among different spatially-located offices. It could be avoided if it would be only one office.
- Even though they use a relatively small set of software tools, they still experience problems with data exchange. Surprisingly, these problems appear even within the same package (AutoCAD) when different versions are used by two different members of the team.

- The validation heavily relies in good human relationships and good manager skills.
- Raúl works in a very hierarchical company where higher rank employees keep strong control over lower ones. Even then conflicts are negotiated by suggesting and evaluating alternatives. So, in essence, the working system remains similar to Vicente's with the only difference of who is allowed to negotiate. That is expectable if we notice that his company is owned by Santiago Calatrava, a well-known architect who bases his business on his very particular architectural style which is easily recognizable by anybody.

Part I
Interoperability at Data Exchange Level

Chapter 6
Motivation

The qualitative research (i.e. ethnographic study) pointed out that data exchange is still an obstacle. In a project, engineers need to send and receive designs frequently. In fact, it is desirable that the data exchange happens transparent- and immediately when there is a need for it. Unfortunately, this is not the case and exchanging data is an extra burden for the project. In a typical project several teams collaborate. Each team specializes in one aspect of the project (e. g. water supply, structures, geology,) and produces one part of the model. The most difficult part is to merge all the partial models into one. In order to have a deep insight of how the data is exchanged it is necessary to know how it can be organized.

Data containing geometric information is known as spatial data. If the spatial data is also referenced to the Earth they are said to be geospatial data. Models in Civil Infrastructure are composed of geospatial data. Normally, the smallest identifiable object in those models is a Feature and geospatial data models are represented by sets of Features. In most of the cases, the word Feature is explicitly mentioned by the data format specification and it is obvious that they are the basic data structure. In other cases, normally older data formats, it is not explicitly mentioned as the unit of information. However, it is easy to identify the concept in all the models. The Feature pattern is followed by the vast majority of file formats. IFC and CityGML are good examples. But there are many others: ESRI™'s shape file, AutoCAD®'s DXF and Bentley Systems™'s DGN, Google's KML, PostGIS, MySQL spatial or Oracle Spatial. A conceptual evolution of the data structure can also be observed and presented in this chapter.

6.1 Models composed of plain Features

Whether the concept of Feature is explicitly defined or not, it can be observed from the early data formats such as AutoCAD®'s DXF or ESRI™'s shape file. In these data formats, a complete model is a vector of the instances of the basic data structure Feature or simply a vector of Features. A Feature is used to represent an object in the real world. The earliest and classic definition of a Feature contains a Geometry and sets of Attributes[1]. The Geometry is used to define the feature shape. Some systems support non-spatial objects by allowing Features with no geometry or by defining empty geometries. The Attributes are properties with a name, a value and a type. Figure 6.1 depicts the structure of the plain Features as in the classic definition of it. The circles represent an instance of a Feature, and the table shows their structure. The Attributes are arranged in columns. Each column is given a name and a type corresponding to those of the Attributes, and each *i*-th cell (row) of the column is given a value which corresponds to the value such an Attribute gets for the *i*-th Feature. As it can be guessed, a dataset like the one shown in figure 6.1 is composed of *n* Features that have one Geometry[2] and *m* Attributes.

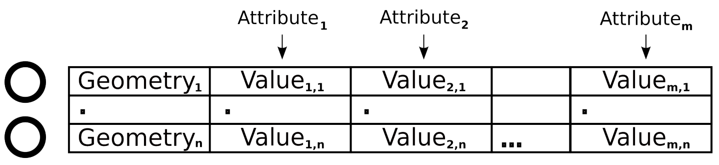


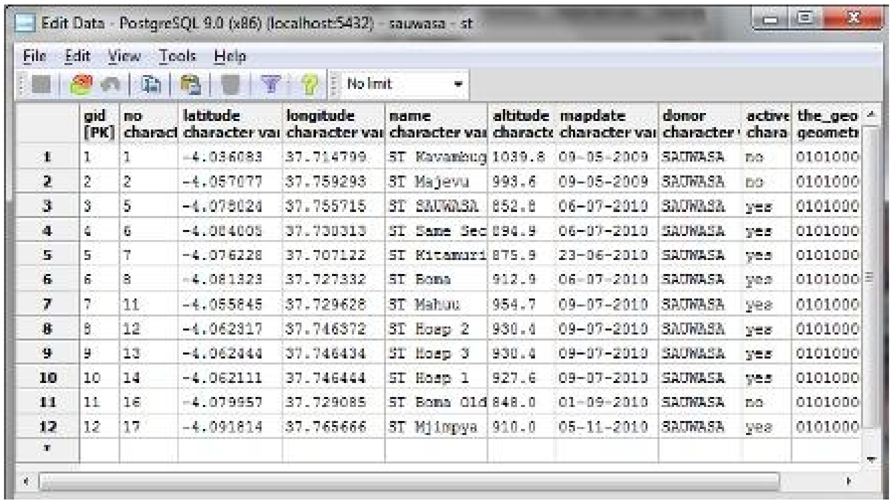
Fig. 6.1: Traditional "plain" data model.

A Feature is, then, a tuple $F = [G, A]$ containing a Geometry, a set of Attributes. Since the Attributes are defined by the name, their type and their value, Attributes are in turn a tuple $A = [AN, AT, AV]$. Combining both expressions, a Feature is formally defined by a tuple $F = [G, AN, AT, AV]$. As an example figure 6.2 shows a screenshot of a data set from a spatial database shown in gvSIG. This data set has

[1] In fact, the Geometry of a Feature is sometimes regarded a special Attribute with a specific type that is used to encode the Geometry.

[2] Though it is technically possible that a Feature has more than one Geometry, it is very rare to find plain data formats that use them.

12 features that are composed of 10 attributes. The header of the table show the names of each attribute and the types of each Attribute can be guessed from the values shown. For instance, the first attribute is named “gid” and is of type “integer”, the fifth attribute is “name” and naturally it is of type “string”, the ninth attribute “active” is of type “boolean” and so on.



Edit Data - PostgreSQL 9.0 (x86) (localhost:5432) - sauwasa - st

	gid [PK]	no charact	latitude character vai	longitude character vai	name character vai	altitude characte	mapdate character vai	donor character	active chara	the_geo geometr
1	1	1	-4.036083	37.714799	ST Kavambug	1039.8	09-05-2009	SAUWASA	no	0101000
2	2	2	-4.057077	37.759293	ST Majevu	993.6	09-05-2009	SAUWASA	no	0101000
3	3	5	-4.078024	37.755715	ST SAUWASA	852.8	06-07-2010	SAUWASA	yes	0101000
4	4	6	-4.084005	37.730313	ST Same Sec	894.9	06-07-2010	SAUWASA	yes	0101000
5	5	7	-4.076228	37.707122	ST Kitamuri	875.9	23-06-2010	SAUWASA	yes	0101000
6	6	8	-4.081323	37.727332	ST Boma	912.9	06-07-2010	SAUWASA	yes	0101000
7	7	11	-4.055845	37.729628	ST Mahuu	954.7	09-07-2010	SAUWASA	yes	0101000
8	8	12	-4.062317	37.746372	ST Hosp 2	930.4	09-07-2010	SAUWASA	yes	0101000
9	9	13	-4.062444	37.746434	ST Hosp 3	930.4	09-07-2010	SAUWASA	yes	0101000
10	10	14	-4.062111	37.746444	ST Hosp 1	927.6	09-07-2010	SAUWASA	yes	0101000
11	11	16	-4.079957	37.729085	ST Boma Old	848.0	01-09-2010	SAUWASA	no	0101000
12	12	17	-4.091814	37.765666	ST Mjimpya	910.0	05-11-2010	SAUWASA	yes	0101000
*										

Fig. 6.2: A “plain” data model of a PostGIS spatial database shown in gvSIG. Source: the gvSIG documentation (http://www.gvsig.org). Creative Commons attribution.

With this pattern, it is possible to define objects. For instance, a place mark feature describing a light pole, could have a symbolic representation as a point as geometry and attributes describing its, say, wattage, operation time, or any other of interest.

In this plain structure, there is no built-in mechanism to associate a Feature with others in the same model. The only association allowed is grouping the Features of the same type, i. e. with same attribute set, in a container structure normally called layer[3] that is independent and unaware of other layers in the same model. The use of layers imposes some limitations. The feature belongs to one and only one layer and

[3] The name layer is an analogy to the old method of drawing in semi-transparent papers. Designs were represented in drawings where each drawing focuses in one concrete subject of the model. By adding or removing a drawing (or a layer) it was possible to see or hide specific parts of the model when it was necessary. Artistic graphic software packages like Adobe Photoshop^{TM} or GIMP also exploit the layer concept.

it is the layer who is in charge of defining the type of the features it contains. Moreover, in GIS systems each layer can contain only one type of Feature to allow map algebra operations. Hence it is a normal practice that the definition of the feature type resides in the layer and the Feature is not really aware of which type of objects in real world it is representing.

6.2 From plain to hierarchical models

Choosing IFC and CityGML as the examples of the interoperable formats with public specifications was not casual. Besides the fact that the specifications are public and well-known, they also share another characteristic that makes them worth special mentioning. In contrast to the "plain" data structure used in traditional formats like shape files, DGN, DXF, or spatial databases (see Figure 6.2), IFC and CityGML data models present an object-oriented tree-like structure approach (see figure 6.3) in which a set of relationships can be established among the features contained in the model. Because the structure of a Feature is defined by its type the Feature *also* needs to define its type. This is a tautology in object-oriented structures that does not happen in plain structures where the structure of the feature is normally defined by its container (or layer as in GIS). Thus, a Feature in object-oriented is a tuple $F' = [F, T]$. A direct consequence is that the dataset cannot be represented in a matrix like in figure 6.2 anymore and a tree is normally used.

The Features in an object-oriented structure follow hierarchies that, as in Object-Oriented Programming, allow a Feature to inherit properties that have been defined for the Features it derives from. Also, being Object-Oriented, some structural design patterns like Composing [Riehle, 1997] Features of other Features are possible.

Moving from plain data models to Object-Oriented ones is a trend that can be observed in the recent years as it allows reusing effort while introduces higher levels of abstraction on the information represented by the data. In the case of CityGML, the model is built up from the basic data type Feature which is composed of its geometry and its attributes. The generic type Feature is then subclassed by more specific and potentially more complex types. An example of a "complex" type is the Building feature which is a subclass of Feature.

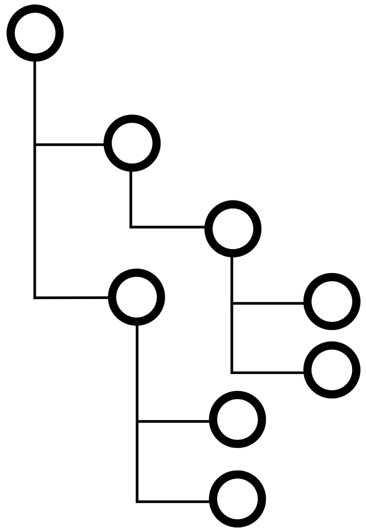
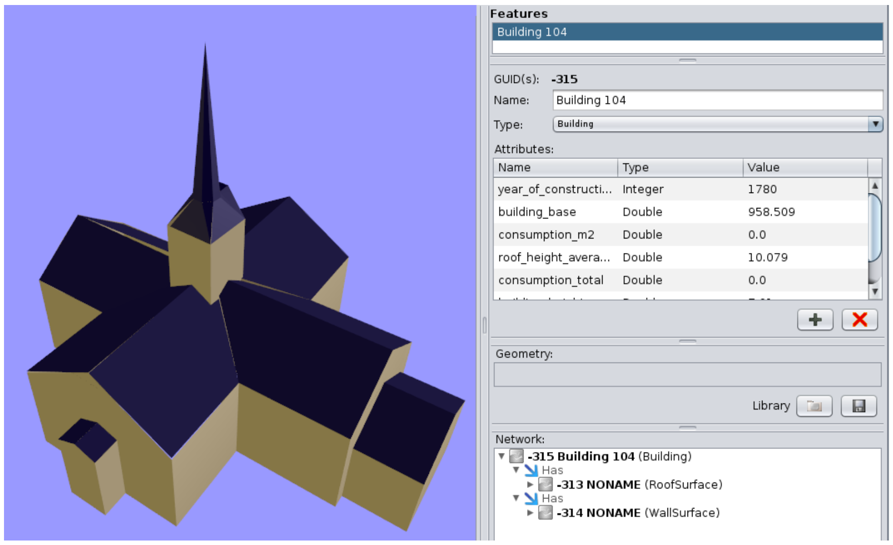


Fig. 6.3: Object-Oriented data model extends the plain Feature concept by adding built-in feature type definition.

Since, Building inherits from Feature, it has geometry and attributes. A Building is also composed of the "simple" RoofSurface and WallSurface features, which have geometry and attributes as well (see figure 6.4).

An Object Oriented data model has the clear advantage of facilitating higher levels of abstraction than pure plain data models. First, the data is intrinsically typed. This means that a, say, Building feature *is* a Building unlike the plain approach where there is no Building *per se* but instead a Feature perhaps with a previously agreed attribute that defines itself as a building. However, in Object-Oriented structure, inheritance defines that any subclass of Building like Library, for instance, *is* a Building as well. It implies that if, for instance, a Building in a given data model has a land registry (cadaster) number then a Library will automatically have it without explicitly defining it. This allows focusing on the definition of the Library feature specifics -like e.g. amount of books- while support for inherited properties is obtained for free. This is, of course, possible to achieve with a plain data



Features

Building 104

GUID(s): **-315**

Name: Building 104

Type: Building

Attributes:

Name	Type	Value
year_of_constructi...	Integer	1780
building_base	Double	958.509
consumption_m2	Double	0.0
roof_height_avera...	Double	10.079
consumption_total	Double	0.0

Geometry:

Library

Network:

```
▼ -315 Building 104 (Building)
    ▼ Has
        ▶ -313 NONAME (RoofSurface)
    ▼ Has
        ▶ -314 NONAME (WallSurface)
```

Fig. 6.4: CityGML object oriented definition of Building. A Building feature with several attributes that is composed of RoofSurfaces (blue) and WallSurface (brown) which are in turn features as well. Screenshot of the Infraworld prototype.

model as well. But it soon becomes overwhelming when the amount of properties to be stored is [not necessarily so] big. A consequence of reduced complexity[4] is that higher level of interoperability can be achieved more easily because the simpler a system is, the simpler it is to operate.

6.3 From closed to extensible data models

Another characteristic of both IFC and CityGML is their ability to be extended. In fact, CityGML is a super set of GML that is extended by providing specific XSD schemas ([Roy and Ramanujan, 2001] and [Sperberg-McQueen and Thomson, 2000]) designed for representing 3D models of cities. CityGML's extensibility mechanism, in turn inherited from GML, allows non-*official* extensions from particular vendors. Thus, those vendors need to provide their own XSD schemas in order to allow applications consuming the data to understand the data being delivered.

[4] Complexity in the sense of level of difficulty when dealing with the data model.

Enabling data format extensibility is a phenomenon that shows how difficult the definition of exchange data format specification can be. Very often, data that has been generated for a purpose is afterwards manipulated and transformed for other purposes. For instance, a road network of a city created by the local authorities to manage the traffic can be adapted by a public transport entity in order to manage the buses of the service lines. Obviously, the buses, as any other kind of traffic, will use the streets to transport the passengers from one place to another. It is clear, though, that both models will partially differ in the information included. While the public transport entity will probably have an overlay network on top of the road network defining the service lines and the transfer nodes, most probably, the initial road network will lack this information.

For cases like the one above, without an extension mechanism, the way to achieve other usages of the data necessarily implies that specific data formats will be used. In other words, the overlay network of the bus service will be created in the public transport entity's own format. This is the option that is normally chosen. It has the advantage that the owner of the format has full control of its definition. The disadvantage is that yet another data format is created which is not directly usable by others. The extension mechanism of formats like CityGML or IFC attempts to find a tradeoff between the flexibility of having a format that meets specific needs while keeping the capability of being reused by others by defining the specifications of such mechanism. So, enabling any system to be able to access to data that is not defined in the, say, core data format is a matter of providing the system with support for the extension mechanism that each format standardizes (.xsd files in the case of GML/CityGML).

6.4 Beyond exchange data format specification

While theory says that data is the same everywhere it is used, in practice very often the same data behaves differently when it is used in different environments. Unfortunately, a well-defined specification for a data exchange format does not guarantee that all the implementations of that standard will support it with no errors. A proof for this is easily obtained by creating a text document in a so-well-known Open Document Text (.odt) format and then open it in several software packages supporting them. Very soon, discrepancies about how they are rendered on the different packages will be detected. Another example is when we surf the Web with different browsers. We find pages that work well in one browser but not in others. And this happens even though the specifications for any standard the web is composed of (HTML, JavaScript, etc..) are very clear. This problem seems to appear everywhere and for every format. Sometimes it is due to the fact that, after all, data needs to be consumed by a program which is the image of how its creator interprets it; some other times, the creator of the consumer program is simply not skilled enough to create a good quality product; some other times it shows that the specification was not so good as it was expected to be when it was designed; other times the owners of a format deliberately give poor support for competing formats in order to keep their position in the market; or, most probably, a combination of all these issues and others. A very representative example of how difficult to avoid these discrepancies can be found in Raúl Rodríguez Fajardo interview. As he comments (see Appendix A's section 0.1 on page 234) they happen even among AutoCAD® versions. AutoCAD® is a product of AutoDesk™ which is the owner of the DWG format. It is very illustrative that not even the owner of the specifications of a format is able to seamlessly deal with the different versions of its own format in its own products.

The evolution of data formats has helped to increase the data interoperability. However, as it has been shown in the previous paragraph, the problem still remains unsolved. When a model is shared by using exchange data formats- to which data is exported and imported from-, we can only hope that the importers and exporters have been implemented correctly so that no information is lost on its way from its source to its destination. Achieving a high quality importers and exporters is not an easy task and demands vast amounts of resources invested in programming and testing. Even then, nothing guarantees that the exchange is perfect. In next chapter, a new paradigm for data sharing created by Dr. Jan Kolář is presented that solves all these problems. It is called Geographic Managed Objects (GMO) [Kolar, 2007]. It allows full interoperability and guarantees that no information is lost. Based in the concept of Virtual Machines (VM), GMOs exploit the property of write-once run anywhere to enable the integral transmission and reception of data in all the client applications that consume them. Features in GMO are object-oriented structures that contain attributes in their data fields. In addition, they are also allowed to transport behavior in their methods as in any program developed in an object-oriented programming language. Thus, thanks to the VM that execute them, it is guaranteed that they will behave exactly the same way regardless of the application that is consuming them and independently of what the internal data model used by that application is.

Chapter 7
[Article] Improving Interoperability of 3D Geographic Features via Geographic Managed Objects

Jaume Domínguez Faus, Wan Wen
jaume@land.aau.dk. wanw@3dgi.dk. Centre for 3D GeoInformation, Aalborg University. Fibigerstræde 11, 9220 Aalborg, Denmark +45 9940 3699

[This article] Appears in: *International Journal of 3-D Information Modeling (IJ3DIM)* 1.1 (2012): 43-54. January 2012 doi:10.4018/ij3dim.2012010103

Abstract This paper presents a method to improve the interoperability of 3D geospatial features from the perspective of data representation, with an emphasis on the interoperability of advanced functions associated with 3D models, such as thematic functions, interaction capabilities, and dynamics. The central concept of the proposed method is to express geospatial features in the form of managed objects, which carry both functions and data of 3D geospatial features in an object oriented and platform-independent manner with the support from virtual machine mechanism. Unlike any conventional data representation method that maintains the data and functionality separately; this method allows the features to carry their own executable functions by themselves in order to achieve the interoperability on a feature's level. A use case is also provided in this paper to demonstrate the presented method. In this use case, different applications can simply fetch the 3D features and run their functions without the need to know what they can do beforehand.

Key words: Interoperability, Managed Object, 3D, Geographic, Data Representation, Behaviors

7.1 INTRODUCTION

A geographic feature refers to an existing entity that is on or near the surface of the Earth (i.e. the geographic domain). Goodchild [Goodchild, 2001] defines geographic as “a subset or specialization of spatial, which by extension refers to any spatiotemporal frame, and any spatial resolution, and also includes non-Cartesian spaces”. Thus, geographic features are also called geospatial features, which are specialized spatial features that carry the specific nature of geographic information [Goodchild, 2001].

7.1.1 Representation of Geospatial Features

The scope of geographic features is extensive; it encompasses any disciplines in the geographic space: geography, meteorology, transportation, energy, criminology, sociology, archaeology, and many more [Goodchild et al., 2007]. This diversity actually brings difficulties in the digital representation of geographic features. The same geographic entity is usually differently defined in different paradigms for the purpose of depicting different spatial and temporal scenarios, and there is no uniform set of identities that reflect heterogeneous representational models.

Therefore, some researchers suggest that the representation of features is application-dependent. [Lehan, 1986] defines a feature as a “physical entity that is recognized in the user’s definition of reality ’and reality as’ the characteristics of a region that are significant to the user of a specific classification of spatial information”. By this definition, “what makes a feature a feature is a system’s users and usages” [Langrana, 1992] (page 49).

In such context, we can say that the number of representation forms is as large as the number of ways that humans perceive the world [Yuan et al., 2004]. However, in order to enable the interoperability, it is also required to let the representation forms be understandable by other systems ([Fisher and Unwin, 2005], [Bishr, 1998]). This requirement is then conventionally fulfilled by standardization, which extracts representation templates that covers most of the needs in a common field of application.

In the field of 3D GIS, the geometry of a feature is normally taken as a basic part of the feature representation. Therefore, we can parameterize a standardized representation of a geographic feature as the following tuple:

$$F = [G, T, V] \tag{7.1}$$

where;

- G is the geometry (if any) defining the shape and the spatial occupation of the feature,
- T is the attribute schema of a specific feature type according to a predefined standard,
- V is the set of values (which are not necessarily numeric) assigned to the attributes.

Since the feature is depicted with strict respect to the T, T in turn also defines the range of values valid for V. Different systems must fully understand the data representation before they can run any functions regarding the F, and this requirement can be really hard when diverse applications are involved.

7.1.2 The Concept of Managed Objects

In contrast to the conventional data representation methods as shown in Eq. 7.1, we propose to use managed objects (MOs) to represent geographic features. A managed object (MO) is defined as a pure Object-Oriented (OO) and a platform-independent binary representation of a real world entity.

An MO carries both the executable behaviors and data. It is said to be managed because it is defined within the paradigm of managed code (MC) concept, which uses the underlying virtual machine (VM) [Smith and Nair, 2005] to manage the code that defines it. Since the code is loaded, executed and unloaded by a VM, it becomes independent from any hardware, operating system or external data sources.

The only mandatory requirement is that there is a virtual machine (VM) running on top of the host computer. It is therefore reasonable to think of the term ‘managed’ as a way to achieve interoperability under the ‘management’ of a VM. The idea of MC has been studied and applied in many technologies, such as Smallworld MAGIK, Java, .NET, Apple’s universal binary and more. However it has not effectively been explored for the representation and implementation of geographic features. At the Centre for 3D GeoInformation (3DGI), Aalborg University, an active research aiming at introducing MC to the digital earth domain has been carried out since 2006. It is worth to mention this paper is largely built up on top of all the previous research at 3DGI.

7.1.3 A Real Life Use Case

The introduction to our approach begins with a real life use case. The city of Drammen, Norway, is involved in an ambitious project whose main goal is to promote zero-emission electricity consumption. This consists of promoting the use of renewable energy and the optimization of energy consumed by heating. In this project, we were asked to simulate the consumption of electricity for each building. The electricity consumption per building is obtained with two methods. The first is the value provided by the electricity company and the second is calculated according to a scale that defines the degree of thermal insulation. This insulation scale ranges from Class A for the best insulated to Class G for the worst. Thus, in the second method the electricity consumption of a building is determined according to numeric tables containing values for the consumption factors for every day of the year. These factors define the kWh consumed per unit of area (i.e. square metres). Then, the consumption can be approximated calculated by multiplying the area by the factor given the building class. To visually simulate the electricity consumption, we decided to let the buildings change their colours according to the electricity they consume on a given day of the year.

In the use case, the MO method is adopted aiming at fulfilling the two following tasks: 1. The first task was to represent 3D buildings that contain the functionality of calculating electricity consumption. 2. The other task was to enable those 3D buildings to work in three heterogeneous geo-visualization client applications: 1) Virtual Globe[1], 2) Novapoint Virtual Map[2] and 3) gvSIG[3]. In other words, those MO-based buildings should ‘know’ how to change their colors according to how much electricity they consume, and they should also be interoperable so that their functions can run on different systems, which have no previous knowledge of what functions are performed by the buildings.

The results in this use case were successful. Figure 7.1 shows one MO-based building displayed by three testing client applications, which have different rendering engines. The definition of the building includes...

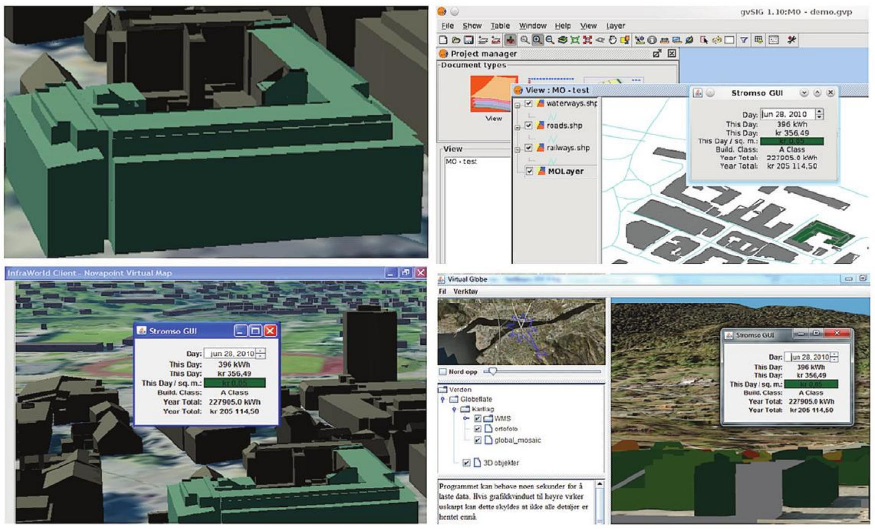
- the method for calculating the electricity consumption on a given date,
- the method that associates the value for the consumption with a color,
- the method that generates the dialog from which the user can select the date
- the method that stimulates the object each time the date is changed.

None of the client applications was required to implement any special function to change the appearance of the building; they only execute the functions contained by buildings.

7.2 METHODOLOGY

The methodology in general involves representing Geospatial Features by Managed Objects. Given the evolution of the geographic systems, we have come across an overwhelming number of new features and phenomena for which we need suitable representation. Consequently, some challenges have risen related to the diversity of representation, system extensibility, system reusability, and functionality

[1] http://www.virtual-globe.info
[2] http://www.viasys.com/vm
[3] http://www.gvsig.org



Stromso GUI

Day: Jun 28, 2010
This Day: 396 kWh
This Day: kr 356,49
This Day / sq. m.: kr 0,05
Build. Class: A Class
Year Total: 227905.0 kWh
Year Total: kr 205 114,50

Programmet kan behøve noen sekunder for å laste data. Hvis grafikkvinduet til høyre virker uskarpt kan dette skyldes at ikke alle detaljer er hentet ennå

Fig. 7.1: One building of the use case (top left): the building is displayed by gvSIG (top right), Virtual Map (bottom left) and Virtual Globe (bottom right), together with the dialog box. ©2011, Asociación gvSIG, Norkart Geoservice AS and Geovekst, Vianova System Finland. Used with permission.

exchange; in other words, interoperability challenges. The number of ways that concretely implement features is potentially so large that interoperability among heterogeneous systems is still a common and open issue.

In this research, we proposed exploiting the MC concept to improve interoperability by implementing the geographic features by means of MO. A MO that represents a geospatial feature is then noted as a Geospatial Managed Object (GMO), which is created by pre-installing (compiling) a piece of Object-Oriented source code, which contains the description of a geospatial feature in terms of both attributes and behaviors (i.e. fields and methods in OO terminology), into an intermediate representation known as byte-code [Dahm, 1998]). Since the VM provides inherent interoperability to the byte-code form, each GMO is like an interoperable black box whose contents are freely customizable.

A GMO can be formally defined as a tuple:

$$F' = [G, T', V', M] \tag{7.2}$$

Where:

- G is the geometry (if any) defining the shape and the spatial occupation of the feature
- T' is the feature type defined by a class definition ('class' as defined by the OO paradigm) instead of by a standard,
- V' is the set of value fields ('fields' as the data members of a 'class' in the OO paradigm) assigned to an feature instance,
- M is the set of methods ('methods' as the functional members of a 'class' in the OO paradigm).

The theoretical benefits of this approach can be summarized in 3 dimensions, as interoperability at feature level, semantic heterogeneity and extensibility:

7.2.1 Interoperability at Feature Level

Interoperability is the ability to exchange information reliably and consistently between different software applications. For a user, it is the ability to utilize information in his/her application from another project participant on a different system & discipline and vice versa. The conventional approach to achieve interoperability is to standardize the exchange data formats, such as CityGML [Kolbe et al., 2005], [Basanow et al., 2008] and IFC. By predefining interoperable schemes or templates for common feature types, this kind of approach achieves interoperability on the feature (type) level. In other words, if a feature instance is not defined according to a standard feature type, it simply cannot be utilized by other applications. However, the GMO method provides an interoperable representation for each feature itself. Every instance can have its own specific definition, independent of any other standards. This capability is allowed by the following two key aspects:

7.2.1.1 MC runtime environments and OO mechanism

The support for data exchange between systems is an intrinsic characteristic of MC runtime environments (normally achieved through the serialization of objects to the byte-code). Like using any other data source, a system needs to know how to ‘load’ the GMOs in order to utilize them. However, the OO mechanism provides a distinct advantage for loading GMOs. Instead of having to know the definitions of all GMOs, a system only needs to know the super-class definition, and all the GMOs that inherit from the super-class are then automatically usable. In a usual case, there is only one super-class that is applicable for a whole field. For example, all GMOs share the same abstract super-class, which specifies that every GMO should have its extent in spatial and temporal dimensions, etc. However, it is important to be aware that this characteristic could become a security risk if the GMO is not used with caution. It is then suggested to have a security manager system to prevent the execution of malicious code in the system consuming the GMO.

7.2.1.2 Self-contained functions

GMOs are created by compiling code into byte-code that is executable by a virtual machine. In this process, the functions defining the behaviors are included into the GMO. Since both data and the executable functions are encapsulated inside the GMOs, they can be considered autonomous. Compared to the conventional systems where any use of the geographic data has to be implemented by the system consuming them, a GMO-based system can plug functions packed in the features themselves within the set M (Eq. 7.2). The utilization of the GMOs depends much less on software designs. Two systems can share both data and functions, without any prior knowledge about how to use the data. The development of feature contents can be completely separated from the development of the systems using these contents by allowing the GMOs to manage their own related functions.

7.2.2 Semantic Heterogeneity

In terms of the semantic interoperability, the ontology research usually plays an important role in standardizing the common semantic terms in a specific field. Conventional standards (e.g. CityGML [Kolbe et al., 2005], IFC [Adachi et al., 2003], etc.) normally embed significant ontology studies, which are worth to follow [Hunter, 2003], [Kashyap and Sheth, 1997], [Harvey et al., 1999]. Regarding GMOs, no data standard is necessarily linked to the definition of GMOs. However, in order to use the semantic interoperability embedded in other data standards GMOs can be defined strictly according to certain standards. Actually, GMOs that follows different data standards can coexist in the same platform. Therefore, the GMO method allows heterogeneous semantics that address different spatial coverage, various social or environmental aspects or specific analyses to cooperate in a comprehensive information system. One can also create GMOs by parsing data defined in other data formats, which allows better communication with existing data in conventional formats.

7.2.3 Extensibility

Extensibility is important because software systems are long-lived and are subject to user's demands for new features and added capability [Cook and Churcher, 2003]. Unfettered extensibility usually demands that function implementations are independent from their input and data structures. With smart deployment of GMOs, the evolution of feature attributes or functions can be associated with the definition of the GMOs themselves. Since GMOs are self-defined, we prevent challenges of system extensibility by keeping functions regarding the feature evolution inside GMOs; while in conventional systems a change in the definition of the GF implies a change in every system using it, a change in the GMO does not necessarily enforce any system re-implementation.

7.2.4 A Conceptual Pattern for representing geospatial features as GMOs

To apply the GMO method, a conceptual pattern for representing geospatial features as GMOs is proposed in order to highlight the data organization and feature representation in geospatial, temporal and application dimensions on a conceptual level. The pattern consists of 4 elements:

7.2.5 Context Information

The context of a feature defines the basic means to georeference a feature and it facilitates the data management and manipulation. With the context information, one can quickly retrieve a certain GMO without having to load and run the object. According to the presumption that every geospatial feature has its spatial and temporal extents [Massey, 1999], [Mennis et al., 2000], [MacEachren et al., 1999], it is natural to tag the temporal and spatial reference to every geospatial feature. If needed by the applications, other attributes can also be used as the context information, such as type, size, etc., for the purpose of fast data retrieving. To tag the context information into every GMO is achieved via the OO inheritance mechanism, which allows a superclass to pass common attributes and operations down to all of its subclasses. Therefore, only the most essential and common attributes are welcome to appear in the context information, in order to avoid inappropriate or redundant information. In this paper, we suggest only the temporal and spatial references, which temporally refer to the life of a geographic feature into the whole universal timeline, and spatially refer the location and spatial occupation to the whole earth.

7.2.6 Spatial, Thematic and Temporal Properties

This pattern also follows the data representation conventions in the GIS field, which turn to describe geospatial features via their spatial, thematic, and temporal properties [Berry, 1964], [Mennis et al., 2000], [Peuquet and Duan, 1995], [Usery, 1996], [Worboys, 1994]. Via the self-contained functions, none of the spatial, thematic or temporal properties of a GMO has to be static. Different properties can be interrelated and interact with each other, as well as the visual properties, and this can support more advanced representation and visualization. A GMO can freely change its location or spatial occupation without losing its identity. Spatially continuous phenomena with undetermined boundaries, such as a piece of terrain, can be modeled as identical features. Therefore we can say that the spatial properties, which conventionally are the identical essence of a geospatial feature in most geographic information systems, lose their privilege in this pattern. For some features, it is not efficient to carry all the properties inside the GMO themselves. For instance, the spatial properties of a terrain model may include thousands polygons; to carry all the related coordinates can be too onerous for a single GMO. In such cases, it is suggested to link the GMO definition to other data sources, and use the behaviors to retrieve the required external data.

7.2.7 Visual Properties

For any 3D geographic information systems, representing the visual properties is an essential part. In this pattern, visual properties refer to attributes that indicate how the geometry and appearance will be constructed. The geometry of the feature also describes the feature's spatial occupancy - the boundaries. The capability of dynamically handling visual properties of the features is significant in this pattern. To achieve dynamic visualization, visual properties can be linked to other thematic related properties or functions of the GMO, so that the visual effects can change accordingly. One can also build up a direct or indirect link between visual properties and interactive interfaces, so that dynamic visualization can be triggered by user interaction.

7.2.8 Behaviors

The conventional data models usually depict geospatial features solely via their attribute data; 'behaviors' in their models are pure tools that manage data associations for the purpose of data management. The GMO method views the behaviors as some essential aspects of the reality, especially when taking into account of the dynamics. Therefore, it is proposed to encapsulate both the data and functions as a whole to provide a nature description of geospatial features. It is worth noticing that it is not necessary to always bind behaviors and data together. A GMO can include only the behaviors parts, so that common behaviors can be defined as tool GMOs shared among different features. With the interoperability guaranteed by VM, the implementation of functions is freely-customizable.

7.3 IMPLEMENTATION

The following provides the details of the implementation along with the data and components used in the implementation.

7.3.1 The Initial State

The initial data in the implementation included; i.) SOSI [4] map file for extracting the geometry of the buildings: the SOSI format is widely used in Norway for defining alignments and other features mostly by means of lines in 3D and ii.) Electricity consumption data: the electricity consumption information is listed in two different Microsoft Excel sheets: one specifying the area and the insulation category of each building and another with the real consumption of each building. The electricity consumption files were provided by the Drammen municipality and by the electricity company Hafslund. The GMO methods were implemented in Java. The VM environment for managing GMO was Java Runtime Environment 1.6. The purposed of preprocessing was to extract the geometry of each building from the SOSI file and associate the electricity consumption information to each building. The inputs were; The SOSI map file and the electricity consumption data, and the outputs of this stage was a file in CityGML format containing both the geometry of buildings and their consumption information. The geometry of the buildings consisted of sets of multi-surfaces defining their boundaries as defined in the CityGML 1.0 standard.

[4] http://www.statkart.no/nor/SOSI/SOSI_English/

7.3.2 Test clients

The experiment was carried out using three existing software packages as the clients for testing the interoperability of GMOs.These packages were:

- Novapoint Virtual Map, supplied by Vianova Systems Finland AS[5]; this is a 3D CAD environment written in C++ and .NET for the Windows operating systems. It is meant for designing and managing civil infrastructure projects. The 3D rendering engine is based on OpenGL and accessed by a proprietary implementation of a scene graph.
- VirtualGlobe, supplied by Norkart Geoservice AS (Norway)[6]; it is a 3D globe similar to Google Earth[7], which can be executed as a desktop application or embedded in a web page as an applet. It is written in Java, it uses Aviatrix[8] scene graph as the internal 3D rendering engine.
- gvSIG, supplied by Iver Tecnologas de la Informacin, SA (Spain)[9]; it is an open source desktop 2D GIS written in Java that is focused on the Spatial Data Infrastructures[10] and allows handling multiple geospatial data sources into the same workspace.

7.3.3 Testing

The test results indicated that

- The three client applications were able to retrieve and load GMOs from the server.
- The three client applications were able to visualize GMOs in their correct place. Thus it can be argued that the GMOs can satisfy the 3D visualization and georeferencing requirements in three different geovisualization clients.

[5] http://www.vianova.fi
[6] http://www.nkgs.no
[7] http://earth.google.com
[8] http://aviatrix3d.j3d.org/
[9] http://www.iver.es/
[10] http://inspire.jrc.ec.europa.eu/

- The three client applications were able to run the functions embedded in the same GMO. Thus it can be argued that the self-contained GMO functions can run on heterogeneous systems without prior knowledge of those functions.
- The three client applications can support the change of colors. The color values were determined by the building GMOs themselves, while the rendering engine on the client side is in charge of rendering the colors. Since the rendering engines are different in the respective test clients, we can test if the GMOs can interact with the visualization functions of the clients (see section 7.4).
- The three client applications were able to change the behaviors of GMOs via plug-ins (see section 7.4), which can test if heterogeneous client applications can also interact with GMOs by changing the class definitions.

7.3.4 Implementing the Methods

This implementation covers various stages which are elaborated in the following.

7.3.4.1 Creation of the Building GMOs

Inheritance: Like any GMOs, the Building class definition used to represent the buildings starts from extending the GMO super-class. This class specifies three main types on the context information. The first type is an ObjectID primarily used for direct referencing or retrieving in databases or over network. The second type is a SpatialIndex constructed from the approximate geo-location of its associated feature. This spatial index benefits a fast spatial query. The third type is the temporal information that records the start and end points of the feature's life span. The super-class also specifies a common function called goMain(), which is the entry point that allows the target platform to interact with the GMOs. Since the goMain() method is embedded in all GMOs, it can be directly utilized by any client in order to access to specific behaviors of the objects. This method takes two parameters: an event object sent by a client to the GMO when the object is activated (for example a mouse click event on the object in a viewer client application), and an object of type Mediator (explained in section 7.3.4.3) that is provided by the client application and represents the action to be performed by the application when the GMO finishes reacting to the stimulus.

Thematic Information: Besides the intrinsic properties of a GMO, the Building class also included 1) a field for the building name, 2) a field to hold the values for the electricity consumption of the building provided by the electricity company in kWh, 3) a field defining the official estimation of the consumption per square metre in kWh and 4) field holding the area in square metres. The building name is a string; the electricity consumption, and the official estimation of the consumption are arrays of 365 real numbers (one for each day of the year) and the area is a real number. The class definition also included the four methods (or behaviors) aforementioned.

7.3.4.2 The Plugins

Plugins are designed to hold complex operations that are not strictly within the feature definitions. Any concrete plug-in is said to be a plugin when it implements an interfaced called PluginIF. A single feature type can apply multiple plugins, as well as a single plugin can suit multiple feature types. In response to a stimulus, the plug-in to be used is determined by the returned value of the goMain() method. For this use case, two plug-ins were created to calculate the electricity consumption. The first simply returned the value of the consumption according to the electricity company for a given day; the second returned a value calculated by multiplying the value of the area by the consumption factor for a given day. The purpose was to exemplify how to change the behavior of a GMO.

7.3.4.3 Creation of the Interface between Objects and the Clients

Mediators: The Mediators are meant to provide sophisticated interaction. But unlike plug-ins, Mediators are meant to be closely bound with particular clients rather than the GMOs. A Mediator provides a gateway for any GMO to access the client application. In our concrete case, three Mediators were implemented (one Mediator for each client application). They were in charge of notifying the clients about the need to refresh their viewers when the building decided to change color. Since each client has a completely different rendering engine, the "refresh" notification slightly varies from one client to another.

7.3.4.4 Creation of a Simple Server as a Data Source

A very simple though conceptually valid server was created to hold the definitions of the GMO classes and the GMOs. The server provided a basic support for querying the features when a client opened a connection. Querying the objects consisted in serializing and sending 1) the class definitions required by the clients and 2) the instances of the objects representing the buildings from the server to the clients through the network using the built-in serialization feature of the JVM. The reason to create a server was to ensure that clients do not know anything about the Building class nor the buildings GMOs themselves. Initially, this information is located in the server only.

7.3.4.5 Adding Support for GMOs in the Client Applications

It was necessary to add a driver in each client in order to download the building objects and class definitions from the server and load them into the clients. In the particular case of Novapoint Virtual Map which is a C++/.NET application, an additional effort was necessary consisting of implementing a mechanism to instantiate a JVM runtime that would execute the byte-code from the already existing .NET code. The process of loading the objects consisted of deserializing the objects queried to the server (via the serialization mechanism of the JVM) and then loading and integrating them to each client's feature model. It is important to highlight that, based on the OO inheritance, this integration process only needs to be done once for the super-class of GMOs. Once done, any subclasses such as the class Building become automatically supported.

7.4 RESULTS

7.4.1 Portrayal of Results

The proposed method was successfully tested on the three clients. Figure 7.1 shows screenshots of the three clients running the same building GMO. According to the spatial context embedded in the GMO, the building is located in the right position in all three clients. The buildings in the simulation were also able to calculate their consumption and consequently change their own color according to the consumption value in the three different clients, which proves that the self-contained functions can be recognized and the mediators work fine for re-rendering the colors of the buildings. As mentioned above, the method for calculating electricity is embedded in the first plug-in (Section 7.3.4.2), which consists of returning a value from a list of 365 values given an index expressing the day or the year. Once it was checked that all the buildings were behaving in the same way in every client and the color change was correct, a test was carried out to change the method for calculating electricity consumption. Then the first plug-in was replaced by the second and more complex one in the Building class definition at the server. After this change and reloading the buildings were able to calculate the consumption with the second method. Even though the clients were not changed to handle the new behaviors, it could be noticed that the buildings were now giving different consumption values (due to the different calculation method applied) for the same date than those before. Visually, the buildings appeared in different colors than before showing that the behavior was effectively changed.

7.4.2 Evaluation of Results

The fact that the building GMOs can change the colors according to their self-contained functions in all three client-applications shows that the interoperability has been achieved at a feature level (i.e. not only at a feature-type level). It proves that the characteristics of the MC technology are applicable in representing 3D geographic features.

The fact that it could be possible to change the behavior of the objects shows that information systems based on the GMO method can be easier to maintain than those based in classic systems. When that at some point, an electricity company decides to publish the electricity consumption of the buildings on the Internet, a system like the one described in this use case could be easily adapted to show the exact consumption in real time; simply by substituting the current plug-in by another one that takes the real-time values from the Internet. In fact, there are no conceptual differences between replacing only one plug-in by another and substituting the old building class by a new one as long as the pattern presented is followed. This effect could also be achieved by providing a completely new definition of the Building class. In fact, this change would only occur on the server but not in the clients, while any conventional systems would probably require the (re)implementation of a tool to retrieve this data in each client.

7.5 CONCLUSIONS

This paper described the concept of Managed Objects applied to representing geospatial features and how it was put into practice. The main objective was to demonstrate the use of GMO method as a way to obtain high interoperability between geospatial systems. Interoperability is conventionally addressed by means of data format standardization. We propose to extend its concept to include sharing not only data but also functionality. In other words, the GMO method can allow users to utilize information in different systems by providing both the data and the function regarding the use of data in an interoperable way. When a system receives an external GMO, it can use the functions carried by the GMO to run the data; and neither prior knowledge about the object nor prior system implementation would be required.

A good way to test the interoperability is to use the GMOs in different client applications. The three clients used in the use case are distinct enough to test the interoperability allowed by the GMO concept. For instance, even though gvSIG has 3D globe and 3D planar support, we tested our method in the pure 2D GIS part, whose rendering pipeline is completely different to the other 3D viewers. Between

Virtual Globe and Novapoint Virtual Map, the scene graphs for drawing the view are also completely incompatible. In addition, Novapoint Virtual Map is not even a Java application. However, based on the interoperability achieved on the GMO method, all these platform differences could be overcome.

It has been proven that it is possible for heterogeneous systems to run GMOs without knowing what kind of objects they are or what they can do, as long they follow the pattern presented in this paper. It could be noticed that supporting GMOs in other applications than those presented in this paper only requires to: 1) write a driver module for the input of the objects which will be reused for every type of GMO (including those objects that do not exist yet), and 2) implement a very simple Mediator that executes the desired reaction of the client (the "refresh view" function in our case). It is worth noticing that since the mediator is provided by the client and passed to the GMO, one can think of having different Mediators to handle different types of events as a mean to make the system scalable from the clients point of view.

This use case was simple but sufficient to test the GMO method. The GMO method is in fact intended to be used in more advanced situations. Because the behaviors of a GMO are executable code generated by a piece of source code, what they can do is mostly limited by the programmer's imagination rather than technical restrictions. This characteristic can be exploited to let geographic features contain functionality to solve a given problem and export this functionality everywhere they are used.

7.6 ACKNOWLEDGEMENTS

We would like to express our gratitude to Jan Kolar, Ville Herva, Runne Aasgaard, Jan Erik Hoel, and Erik Kjems for their enormous work on designing and developing the software packages used without which this paper could not have been possible.

Chapter 8
Other use cases for GMO

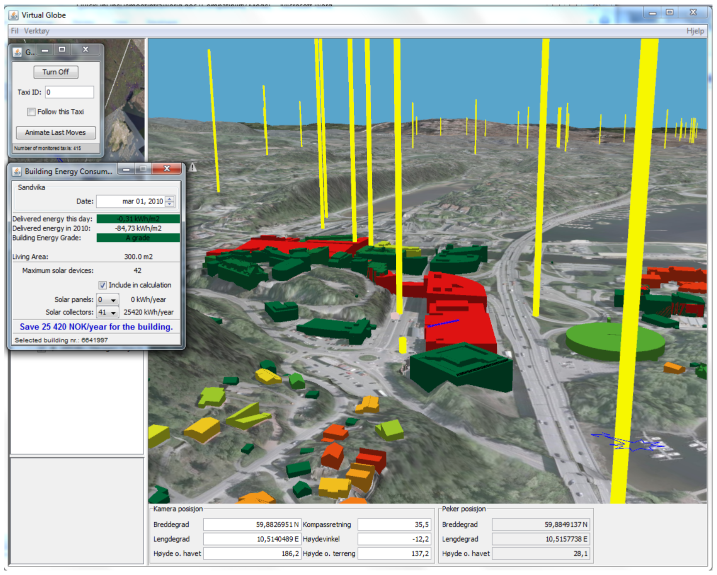
As part of the Infraworld research project this PhD was associated with, GMO's were applied in more situations to test their power and flexibility. Some of the goals were to test their applicability to environments where models require live interaction and moving objects in a model that is being operated by several users in a distributed manner. Here, two use cases are presented: a monitoring system for taxi fleets and a simple scenario with smart features that, being aware of their environment, perform simple conflict solving. In both cases, the systems benefits from the same properties explained in Chapter 7. Namely: complete data exchange interoperability and distributable functionality across any client application with GMO support.

8.1 Taxi fleet monitoring

In this use case, the fleet of the Bærum taxi company was implemented through GMO's. The company operates the taxis with license in this district to the West of Oslo, in Norway. Bærum taxi company has more than 400 taxis and maintains a web service that publishes the current position of each car. The positions are real-time updated every 30 seconds with the values given by GPS receivers installed in the cars and transmitted via GPRS to the central server.

Each taxi is implemented as a GMO that regularly updates its position by pulling the taxi current location from the taxi company's web service. As it can be seen in Figure 8.1, the taxis are implemented as very tall yellow bars for convenience so that when zooming out in 3D applications such as Virtual Globe[1] [Butler, 2006], the bars will still be visible even if the position of the viewer is several kilometers over the sea level. It has to be noticed, though, that nothing prevents the GMO to be implemented with a more realistic car-shaped geometry

[1] http://www.virtual-globe.info



Taxi ID: 0

Number of monitored taxis: 415

Sandvika

Date: mar 01, 2010

Delivered energy this day: -0,31 kWh/m2
Delivered energy in 2010: -84,73 kWh/m2
Building Energy Grade: A grade

Living Area: 300.0 m2

Maximum solar devices: 42

[✓] Include in calculation

Solar panels: 0 0 kWh/year
Solar collectors: 41 25420 kWh/year

Save 25 420 NOK/year for the building.

Selected building nr.: 6641997

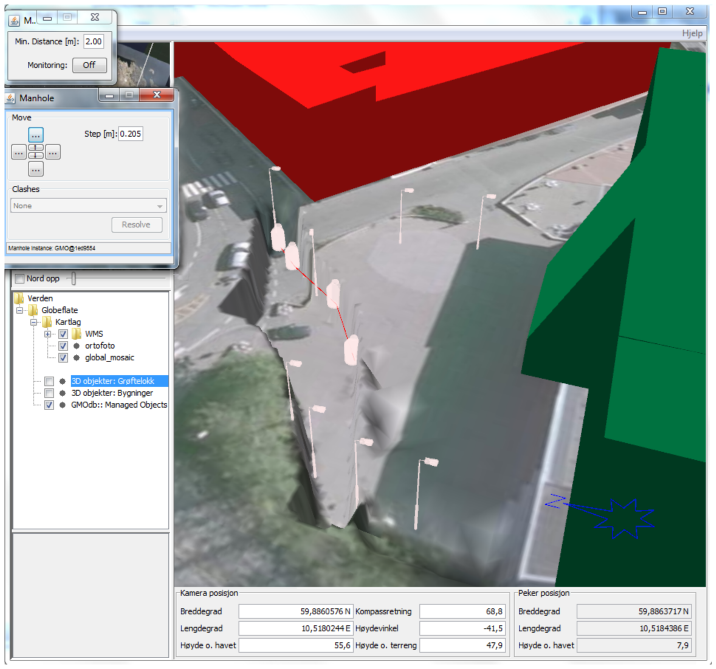
Breddegrad	59,8826951 N	Kompassretning	35,5	Breddegrad	59,8849137 N
Lengdegrad	10,5140489 E	Høydevinkel	-12,2	Lengdegrad	10,5157738 E
Høyde o. havet	186,2	Høyde o. terreng	137,2	Høyde o. havet	28,1

Fig. 8.1: Screenshot of Virtual Globe with a model containing GMO's representing taxis as long yellow bars. The bars move live according to the actual position of the taxi.

if that would be a requirement of a commercial application. Actually, thanks to the capability of carrying data, GMO are enabled to define visual information about themselves such as the typical 3D graphics ones like appearance, textures, and the like. Again it was demonstrated that client applications only need to implement a small module that connects the GMO with the client's event handling system. The reason is that the client application needs to repaint itself when a GMO notifies that the current view is outdated because the object has changed. The same applies in the other way around: click events and similar are generated by the application and, thanks to the module, those signals are sent to the GMO to stimulate the behavior they implement. The design and development time necessary to implement these GMO was approximately 3 normal working weeks.

8.2 Simple conflict solving

In this use case, GMO's were tested as a tool to solve a simple design conflict. There were two types of GMO. One implementing realistic manholes and another one implementing realistic light poles (see figure 8.2. As a matter of curiosity, each light pole GMO is composed of 5000+ triangles (which shows the graphical level of quality that can be achieved with GMOs). The functionality included in the GMOs consisted in detecting when a light pole was clashing with a manhole or any of its connections. When a conflict was detected, the user interface provided by the GMO allows the user to execute another built-in functionality that relocates the light pole in another free place by translating it a certain and preprogrammed distance.



Min. Distance [m]: 2.00
Monitoring: Off
Step [m]: 0.205
Clashes: None
Manhole Instance: GMO@1ed9554

Kamera posisjon
Breddegrad 59,8860576 N
Lengdegrad 10,5180244 E
Høyde o. havet 55,6
Kompassretning 68,8
Høydevinkel -41,5
Høyde o. terreng 47,9

Peker posisjon
Breddegrad 59,8863717 N
Lengdegrad 10,5184386 E
Høyde o. havet 7,9

Fig. 8.2: Screenshot of Virtual Globe with a model containing light poles and manholes as GMO's that are able to detect when a manhole or its connection with other manholes is overlapping a light pole. In front of a conflict, the corresponding GMO will relocate itself to another non-conflicting location.

This use case demonstrated that the GMO's capability of transmitting behavior can be exploited to provide a mechanism that extends all clients with the capability of solving conflicts. This is a big step ahead. Eventually, if the way conflict is solved (i.e. relocating the light pole) changes due to changes in the system requirements, then only the GMO's need to be changed. By providing a new definition of their methods it is possible to update their behavior to meet the new needs. For instance, it could be decided that instead of relocating the light poles, it is preferred to remove them. Then, the only thing that has to be done is to implement that deletion behavior in the light pole GMO's method[2].

Unfortunately, there is one important downside of the GMO approach regarding the conflict solving. It assumes that a conflict has a fixed way of solving it. While this can be true in some cases, it is not in general. In real life a conflict is not always solved the same way. Actually, is the context in which it occurs what drives the decision of how to solve it. Considering that Civil Infrastructure projects are executed in open environments, it is easy to realize that, in general, the context of the problems varies. Thus, unless the conflict to be solved is very specific and a solution for it is very clear, GMOs, despite their clear interoperability power, are still not sufficient for solving complex conflicts requiring higher level of decisions. Part II of this thesis deals with these situations. It is built upon the recognition that, as will be explained, Civil Infrastructure projects are *situated* (meaning that they are context-dependent and context-variant) and have a social-collaborative component that cannot be covered satisfactory just by defining new data formats.

[2] Actually, any GMO could implement that behavior, not only the light poles. For instance, it can be the manhole who is programmed to remove a light pole if that light pole is *disturbing* it

Part II
Interoperability at Distributed Behavior Level

Chapter 9
Motivation

The first part of this thesis dealt with the data exchange aspect of interoperability. This section deals with the other aspect of the interoperability in Civil Infrastructure Design: the Interoperability at a Distributed Behavior[1] Level. Thus, issues regarding the data exchange are set aside. The main focus is now shifted to how the project, as a living entity in itself, manages the sum of the individual initiatives of its members to build up what has been called the **distributed cognition** [Hutchins and Lintern, 1995].

It is assumed that the exchange of information is of key importance in collaborative team work. It seems, however, that facilitating file (or data) exchange is the only way that has been considered when it comes to improve interoperability among the project participants. Either by making smarter data formats, powerful repositories, or -as proposed in Part I of this Thesis- with the experimental Geographic Managed Objects (GMO's), the attention is mostly put in the data (bytes) transferred. After the ethnographic research, it remained a feeling of that the engineers are stuck in the thinking that the only thing that can be improved is making better transport of the data itself. Instinctively, the only limitations in the current systems the engineers use that they can identify are those coming from the computer system. A solution for a better organization of the data in repositories can be the use of BIM-like servers [Eastman et al., 2008] which allow centralizing the designs and other documents of a project for easier access to them. BIM servers can also allow defining access permission to the files stored so users can access only to the designs they are authorized to. BIM servers also provide a *versioning* mechanism that can

[1] As the reader might have noticed, the meaning of the word "distributed" slightly differs from what it is normally understood as something that is to be spread out. Instead, it refers to a situation in which a group of things, members, services, or any other concept is already spread out, maybe even strategically, in order to be exploited as a whole. This is the meaning that the word "distributed" takes in terms like "distributed systems" and that is probably very familiar to computer scientists but could sound strange to others. However, it is a perfectly valid one. For instance, the Internet is a distributed system which is already there and it was not distributed by anybody in particular.

be exploited to keep track of the evolution of the project. Obviously, a BIM server is an implementation of what the engineers would imagine as an improvement for their work. However, the design work is not only performed by the computers and a set of software packages, computer networks, data storages and data formats. The design work is performed by a system composed by these tools for sure, but also by the persons, the engineers, who work on it. Without the engineers the system is simply a dead inoperative apparel. The tools are operated by humans who use their experience and skills to manipulate them.

Sometimes, there are tasks in the design work for which there is no computer tool available. In those situations, human have no other choice than their good old manual procedures to perform those tasks. That is the case of the process of solving design problems. Certainly, there are computer tools that detect and solve problems. But the problems they solve are normally very specific (e.g. structure calculus, pipe network pressure calculus, etc.). There is no general-purpose tool for solving problems. However, a project will have problems. That is simply a fact that is pointed out by the ethnographic research presented in Chapter 5. Since there is no general tool for solving them, the problem solving task is performed manually by humans. The question is how those problems are addressed currently and whether it is possible to treat them in some more advanced way.

During the ethnographic research, several classic problems that can occur in a project along the project different phases (see figure 9.1) could be registered.

Recalling that the most common problems engineers identify (refer to the ethnographic studies and the interviews in Chapter 5) with the phase they typically occur are:

1. Missing or incompatible data (Data Gathering Phase); designing a project is to imagine what it can be based on what it is now (the existing situation). Defining what it is now is a process of the Data Gathering phase that consists of collecting the available data from relevant entities (authorities, water supply companies, electricity companies, etc.). The data needs to be adapted to be homogeneous to the project. As already mentioned in Part I of this thesis, the data adapting entails data exchange problems. In addition, usually some data needs to be created because it does not exist yet.

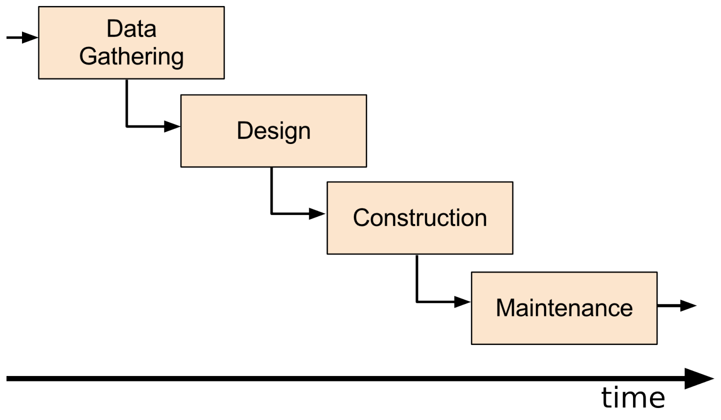


Fig. 9.1: The distinct phases composing a project. Ideally, the phases occur sequentially. In practice, there are overlaps in time where a phase is approaching completion and the next one has already started. In these cases, both phases evolve in parallel in which the workload of the previous decreases while the workload of the next increases.

2. Design conflicts (Design Phase); since a project is not designed by a single person but by groups of them working in parallel and not necessarily connected, a partial design can be in conflict with other group's design or interest. For instance, an installation that cannot be in the middle of a road, a light pole that has been placed too close to a manhole, etc. It must be kept in mind that engineers work with only partial designs of things that do not exist yet. It is very often that these designs need to be validated and, in case of conflict, parties involved in such a conflict need to agree on how to solve it. As mentioned in previous chapters, today this is a manual process still.
3. Changes in the supply chain (Design Phase and Construction Phase); refers to the situations in which the materials needed for the infrastructure are not supplied as they were expected and that forces the project to be redesigned. Some reasons exist for this:
    a) Raw materials' prices differ from what it was believed by any external reason. E. g. supplier can not longer manage the materials at the accorded price, international prices for steel, copper or anything are changing, transport costs change, etc.
    b) Materials cannot be delivered at the time they were agreed. For instance, the factory producing the materials can not provide them at the time it was expected. This is particularly frequent

for special pieces of structures like buildings or domes that are not mass-produced.

4. Unexpected existing situation (Design Phase and Construction Phase); this happens when the area to be constructed turned out to be different to what it was supposed to be according to the the geologists in the Data Gathering phase. Since the land that was going to hold the infrastructure no longer corresponds to the type and shape it was considered for the infrastructure, the design needs to be reconsidered. Some examples are:
   a) A terrain initially considered rocky that turns sandy and too soft when the machinery starts movements of land.
   b) Unregistered installation or construction in the available data that is discovered once construction works have started (e. g. archaeological ruins, ancient factory installations, etc.)
   c) Less probable but also possible is that the terrain can change as a result of floods, avalanches, landslides, volcano eruptions, and other natural disasters.
5. Client modifications to the project (Design Phase and Construction Phase); once the project is designed and delivered, the client (typically the authorities) can accept it or propose modifications. Ideally, these modifications occur before the Construction phase starts so that only the plans need to be changed. If the Construction phase is already ongoing, it might occur that some partially built parts such as structures need to be recalculated and rebuilt.
6. Wrong budget that leads to project reconsideration (Construction Phase); when a design shows unreasonable budget estimation and some of the members of the project -typically the contractor- complain arguing that the project is not realistic compared to what the available resources are. When this situation occurs, a new complete design needs to be done, or in the worst case, it can lead to the project's cancellation.
7. Construction works synchronization (Design Phase and Construction Phase); frequently a big infrastructure is executed stretch-by-stretch. If the schedule is not followed by some of the stretches then subsequent stretches might need to be delayed. In that case, some management decisions need to be taken in order to thwart project's global delays.

This thesis will circumscribe to problems in the Design Conflicts category since they are the goal of this project. Problems in other phases have a big component of political and work organization which fall out of the scope of this thesis[2].

9.1 When (current) technology does not help

The first thing to be wondered is "why do we still solve Design Conflicts manually?". This question was asked to the engineers. Their answer was the same in every case (in their respective mother tongues): "Well, how else do you want to do it?". What this skepticism actually acknowledges is that current technology has difficulties in tackling this problem.

First, the multidisciplinary nature of Civil Infrastructure projects derives in a wide range of software tools used by the engineers, most of them with their corresponding data models. It must be recognized that each different tool will introduce a layer of complexity that needs to be overcome. This issue is already dealt in Part I of this thesis by suggesting the use of Managed Objects that virtually remove the data exchange barriers plus providing with a mechanism to distribute behaviours.

But also, there is the singularity of every project. Unlike automated production where tasks are repetitive and can be planned into an algorithm, projects tend to be an ad hoc and one-shot work. In Civil Infrastructure, this trait is taken to the extreme because projects are executed in uncontrolled and completely irregular environments. So, there is a very high chance that what has been designed for a project cannot be used for another. In other words, very unlikely a road can be copied and pasted from one project to another, simply because the existing situation (land, road network to connect it to, etc.) in both places will be, in general, different. Many uncountable variables also apply like different weather conditions that force using one strategy (decisions) or another: different material supplier, new regulations on environment, materials to be used, etc.; the project has distinct partners with

[2] For instance, by turning the project phases into less sequential and more iterative some of the problems derived from unexpected existing situation could be avoided. Whether this is possible or interesting would be a PhD thesis in itself more in the area of the Industrial Engineering, though.

other goals, skills, etc.; a particular situation that turns the "natural next step" of a process into an "unreasonable decision". The reader can try to imagine possible situations that can change the course of design decisions. The list is virtually infinite.

The traditional approach in the industry towards improving design tools is to enrich the model with every parameter involved in the decision-making process. If the tools are really ambitious, they would even include the corresponding procedure that from a set of rules, perhaps specifically tailored for a project, it is able to automate some concrete decisions about the design. This approach has proven to be very good in controlled environments. In fact, when the procedures for a given design conflict are clear, it is a very reasonable one. An example using this principle is, as we saw in Chapter 8, to use the Geographic Managed Objects (GMO) concept to implement such procedures and allow every user to take advantage of them transparently. If the procedure eventually changes, then it is only a matter of updating the corresponding GMO to implement it and it will automatically behave exactly the same way everywhere it is used. Unfortunately, because of the virtually infinite factors that affect which decision is the adequate among the, again, virtually infinite set of alternatives of a conflict, this technique cannot be generalized. Hence the engineers skepticism and their preference for the manual way to solve Design Conflicts.

In next sections, the theories that can be used to treat this problem in an alternative manner will be revisited and, gradually, an alternative approach will be developed and proposed.

9.2 The 'Situated Nature' of Civil Infrastructure projects

A process that depends on the conditions it is carried out, the people that is involved, the civic structures, individual views, the spatial and casual context, and the interaction relationships is known to have a 'situated nature'. This is the central term of a field of study called Computer Supported Collaborative Work (CSCW) systems. CSCW is originated in 1984 from a workshop organized by Irene Greif and Paul Cashman [Greif, 1988]. In short, CSCW focuses on the challenge of designing systems that are social-aware. It pays special attention to the role of computers in the distributed work. Along its history, the CSCW field has agreed that problems with a situated nature turn out very difficult to be addressed with the traditional analytical way. Actually, CSCW guidelines are sometimes against what is considered good practice in software engineering, i.e. requirements gathering, analysis, abstraction, design, implement and test of algorithmic procedures [Scott et al., 2003] [Fitzpatrick, 1998]. In contrast, CSCW acknowledges that a situated process has much more portion of social than computer science. That is so because there is an overwhelming amount of possible situations that the abstraction of a system turns out to be 'wicked' [Fitzpatrick, 1998], as defined in the term 'wicked problem' [Rittel and Webber, 1973].

On the other hand, in order for a system to be possible, an abstraction of this system needs to be made. Here is where we encounter the CSCW wicked problem paradox: "the challenge for designers is that, while the CSCW problem domain may be wicked, they must ultimately design and build systems that are sufficiently bounded and specified that they can be built as software. Finding the right abstractions that can account for a wicked problem domain but also guide design of tame systems is a key issue in meeting this challenge" [Fitzpatrick, 1998].

While CSCW might seem disappointing due to its lack of clear structure -in fact, it became a highly fragmented field [Schmidt, 2009]- and maturity -the word 'paradox' is frequently used in the CSCW literature- because of its young history, it is a good tool to recognize when a problem belongs to the type of problems it deals with. So, the foundations on how to address them become much clearer. According to CSCW, Civil Infrastructure projects, as a collaborative work, need to be addressed with much more focus on the social structures and relationships among users than the pure analytical approach that is traditionally taken by industry. Acknowledging this is a cornerstone for this thesis, since it unlocks barriers that seemed to be insuperable. However, other obstacles need to be overcome.

9.3 The "Programmer's Dictatorship" problem. [3]

Design conflict solving is a matter of finding a good alternative to a given conflict. It is the process by which engineers detect that something is causing the design to be wrong from a global perspective (even though it might be correct from some individual point of view) and propose an alternative design that satisfies the global requirements. This is something that some software application can certainly do. For instance, one could write an algorithm that detects when a light pole has been put in a location that conflicts with an existing manhole as shown in Chapter 8. It would only require an easy geometry and topology analysis and maybe some rule checking too. One could even add the functionality to automatically fix this kind of conflicts by placing the light pole in an alternative location calculated by the computer. In that case, the programmer of such functionality imposes how the design conflict is solved to the users of the system. That is what the concept of *The Programmer's Dictatorship* refers to, i.e. the fact that a software behaves in the way it has been ordered to (programmed) when it was written at the beginning. That is the approach taken by probably all current software solutions for design. However, that is a problem as it is described below.

The Programmer's Dictatorship approach would make much sense in the light poles and manholes example above if the project had "officially" pre-agreed that is the light pole what needs to be relocated. However, given the strong situated nature of projects, the Programmer's Dictatorship becomes too inflexible. That is because, except for some specific exceptions, it is not possible to define a pre-established way to solve a conflict. In other words, why relocating the light pole is the valid solution for that conflict and none else like, for instance, removing it, or relocating the manhole instead? Moreover, even if relocating was the solution, why placing it in position X and not in position Y instead? To the author's opinion, these two questions summarize the critics engineers have against current solutions. It means that

[3] I want to give the due credit to Dr. Jaume Garcia i Segarra from Universitat Jaume I regarding the term "the Programmer's Dictatorship". We both worked around this concept independently in our respective works until we realized they were in some parts very related but differing in their point of view: from the Economics (his) and Computer Science (mine). However, he is the actual coiner of the term.

there is a low chance to implement a procedure for solving a conflict that will be a silver bullet.

So, when it comes to conflict solving, traditional approaches (i. e. writing algorithms to solve the conflict) for creating solvers soon show their unsuitability. So it does the behavior -read algorithms, procedures- capability of the GMO's because, even though it introduces the ability to distribute and substitute behavior at wish, it still takes the traditional "pre-programmed" approach. i. e. The Programmer's Dictatorship approach. Nevertheless, a conflict might be solved in many different ways depending on the situation at hand. Many reasons can apply for which by no ways they can be known at the time the GMO is designed. The only response to this complexity the GMO can give is to make a more complex method. However, it could be easily asserted that the new version of the method will become obsolete virtually as soon as it is released.

9.4 Multi-disciplinary implies many different meanings

The situated nature of Civil Infrastructure projects is not the only challenge; there is also the multi-disciplinary nature of them. The reason why a project is divided in discipline teams is because different fields of expertise are required. By definition, each expertise will give a different perspective to the project (after all, that is why they are part of it). The consequence of this is that an object in the model can no longer be associated with one and only one meaning but instead it can mean multiple things. For instance, if we look at the light pole example, for a sewer designer designing the manholes, a light pole is no more than an obstacle that cannot share the same space than any of his objects. Thus, so long the light pole does not overlap on anything it is more than correct. But for the electrician designer, it means that a cable to power it up needs to be installed too, and in turn that this cable is providing the right wattage and voltage. Failing on any of those requirements means that there is a problem that can turn into a conflict if, say, the cable somehow interferes in some sewerage installation or any other third discipline's design involved in the project.

The problem of multiple interpretations of a model is, consequently, a very important and nuclear one. As mentioned earlier, today this problem is solved only indirectly with the use of specialized applications that provide the particular tools to manipulate the model in the way each expert needs. And this drives to the consequent interoperability issues that have been already addressed.

9.5 What then?

It seems that the current state of the things creates a situation in which it is difficult to evolve towards more advanced automatic design conflict detection and solving unless the problem and systems are revised completely from its foundations. Recall that the problem to be solved is the process of solving design conflicts because it causes money loss due to the many errors that happen. The question now is how to do it. If traditional analytical and pre-programmed approaches find so many difficulties to solve it so that humans still do it manually, then it is worth studying how this manual approach is and whether it is possible to imitate it by a system according to the tools Computer Science can provide. In the next chapter an approach to the concept of design conflict is given from the semantic point of view together with a formalization of it. In Chapter 10, the formalization of the conflicts is exploited to propose a system that is capable to detect them. A conflict appears because different intentions or goals collide. In Chapter 11, negotiation is proposed as a mechanism to solve conflicts and the theoretical mathematical model is studied and expanded in order to make it applicable to a system. In Chapter 12, these concepts are implemented in a system demonstrating their applicability. Chapters 13 and 14 show how the system can be evolved in order to exploit the negotiation mechanism for conflict solving as a way to create adaptive software that evolves with the projects and gains competence that it was not pre-programmed for, and consequently, responding to the challenge of the situated nature of civil infrastructure projects. Chapter 15 is an article published in Proceedings of the Autonomous Agents and Multi-Agent Systems 2012 Conference that shows the Infraworld framework, a working system based on the research presented in the previous chapters. Finally, Chapter 16 is a discussion and comparison of other related works.

Chapter 10
Conflict Detection

10.1 Detecting *semantic* problems.

During the development of a project undesirable design problems occur. The quality of the project depends to a big extent on these problems. If a problem remains undiscovered until it is too late, the project can become very expensive or even unfeasible. Conversely, if a problem is detected immediately, the negative effects effects on the project can be very limited or negligible. Then, it is desirable that the design is continuously checked. If a computer system that monitors the design in order to detect problems exists, it implies that a formal notation has been defined such that the computer can interpret it. The challenge is that a problem in the design might not be obvious to every engineer. Sometimes, a given situation is a problem only for some engineer's particular eyes but it is not for the others. An example could be a power line with less capacity than the power demanded by the installation it is serving. Probably, most of the engineers will not notice this because in principle, they primarily care about it not overlapping other parts of the design. Most likely, it is the electricity designer who discovers the problem because such power line has a meaning for him that is not necessarily the same meaning that other engineers give to it. Therefore, a computer system that can use that formal notation to let the engineers define their particular point of view is required so that a computer can process it to capture the meaning of a model. In other words: the semantics for each of the particular points of view.

Semantics are composed of things and the relationships among those things[1]. The study of semantics as a branch of philosophy roots down to the Ancient Greece. Parmenides is often referred as one of the first to propose the characterization of the fundamental nature of existence which later shaped the Western Philosophy. By the simple fact of existing, things *are* something, *have* attributes and establish

[1] the term "thing" must be read off as "thing in general", i.e. a concept, rather than some "particular thing".

relationships with others. Aristotle called this study the "primordial philosophy" in its classic work *Metaphysics* and his disciples renamed it to "metaphysics". Metaphysics involved the process of identifying and classifying things into concepts and weave all the relationships that can associate a concept to another. For the philosophers, this process could happen consciously (through meditation) but also unconsciously.

In 1613, Rodolfo Goclenius used the latinized term "Ontology" in his work *Lexicon philosophicum, quo tanquam clave philosophiae fores aperiuntur* [Goclenius, 1964]. Later, Jean Le Clerc was the first in making technical use of Ontology [Le Clerc, 1736].

In its process to be used as a tool, Ontology is tailored according to how it is used. They can be used as a kind of structured dictionary of terms in a field that, in addition, also defines the relationships among them[2]. Ontology is more often used in the sense of building "an Ontology for", i.e. as a concrete use case of the philosophic meaning of Ontology. This derives in the situation that there is no clear way to define it. In words of [Bubenko Jr, 2007] "the term 'ontology' seems to have as many definitions as there are ontologists' ", a problem that other authors tend to acknowledge (e.g. [Spyns et al., 2002] and [Yu, 1997]).

Even though it is not a widely used term, Ontologies are utilized even if there is not a real awareness about it. Maybe, the most paradigmatic case is the Object-Oriented modelling. Object-Oriented modelling is an Ontology in itself since it classifies objects in concepts (the class) and allows establishing relationships among them (inherits, generalizes, aggregates). The reason why it makes sense to call it Object-Oriented, or rather not to call it Ontology, is because in Object-Oriented the set of relationships is limited (since the focus is paid on encapsulation of data and, maybe, operations so that the objects can be easily reused). Unlike Object-Oriented, an Ontology allows an arbitrary set of relationships (see figure 10.1 where curved arrows represent these arbitrary relationships in addition to those provided by Object-Oriented ones) among which those classic in the Object-Oriented approach are, of course, allowed. This is the definition this

[2] In fact, it is possible to find ontologies for almost anything. For instance, Enterprise Modelling [Fox and Gruninger, 1998], Medicine [Rector and Horrocks, 1997], etc.

thesis takes for Ontology[3]. Recall that, in Part I of this thesis, the Object-Oriented approach to create Geospatial models was presented as an evolution from 'plain' models (see figure 6.1 on page 72) to Object-Oriented models (see figure 6.3 on page 75). The Ontology approach can be seen as another step in the evolution of models presented in Chapter 6. The evolution starts from plain models in which the Features were formalized as tuples of Geometry, Attribute Names, Attribute Types and Attribute Values: $F = [G, AN, AT, AV]$. The next step in the evolution was the Object-Oriented data models that introduced the built-in feature type. The Features were defined as a tuple that, in addition, had the feature type: $F' = [F, T]$. Hence, following the same approach, a Feature in an Ontological data model that defines its relationships with other Features, is then formalized as: $F'' = [F', R]$ where R is the set of relationships.

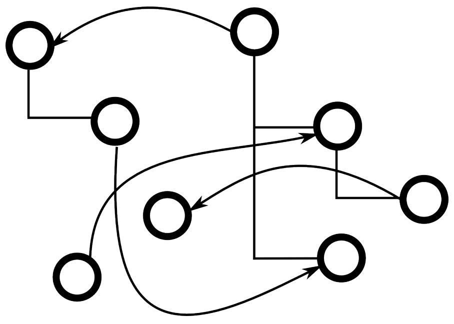


Fig. 10.1: Ontological approach for data models.

One of the consequences of the ontology's ability to define arbitrary sets of relationships is that an object can actually be regarded as multiple meanings (types). Consequently, a Feature can be of type A, and type B, and more, at the same time[4]. Hence, belonging to more than one type a Feature can capture any viewer's point of view "label-

[3] I.e., they are used in the sense of a mechanism to define semantic-rich computer models, rather than the pure philosophical way of cataloguing concepts and relationships.

[4] While it could be argued that some multiple inheritance-enabled Object-Oriented models could allow this too, what makes Ontologies especially appropriate is that in Object-Oriented models an instance is set to a type upon its creation and sticks to that during all its existence. In contrast, in an ontology, an instance of a type can be created before it is set to that type. Though advisable, the type does not even need to be known in order to make the instance usable for other purposes. Then, compared to Object-Oriented, Ontologies provide a flexibility that the former cannot.

ing" it with as many types as necessary. Moreover, a Feature can also be other things in the ontology other than a Feature at the same time. As it will be shown in next section, this capability will be exploited to detect when a Feature is also a [part of a] Conflict and which other Feature's it is conflicting with.

Another reason why using ontologies is appropriate is because computer reasoners can analyze them and infer information that remained undiscovered or hidden. Furthermore, reasoners can be extended with rules that create information that was not present.

In this thesis, OWL[5]([Bao et al., 2009], [McGuinness et al., 2004]) and SWRL[6] [O'Connor et al., 2005] languages are used to allow defining different points of view the members of a Civil Infrastructure project may have. OWL is a formal language that allows specifying Ontologies as the term is used in this thesis. OWL language is chosen because it is considered well-known, standard and XML-based which makes it very easy to process. SWRL language is used because it integrates rules in an OWL ontology very well. The combination of them covers sufficiently the needs to build (or use) reasoners.

10.2 Ontology-oriented data models for distributed-collaborative environments

It has to be stressed that one of the challenges of detecting semantic problems in multidisciplinar designs is that the semantics are subjective. Every discipline is focused on a type of semantics that it is not necessarily important for others. Thus, a system aiming to capture the subjective semantics and detect the problems derived from them needs to be able to manage potentially all the semantics required by the disciplines while its complexity remains acceptable. In Computer Science, it is a common practice to separate complex systems in different layers of abstraction in order to keep their complexity under manageable levels. The idea behind the layers of abstraction is that a lowest layer is in charge of dealing with complexities closest to the machine;

[5] OWL stands for Web Ontology Language. The order of the two first letters was swapped to form the word "owl" that name the bird that is traditionally used as the symbol for Philosophy and associated with wisdom.

[6] Semantic Web Rule Language.

the second lowest layer deals with other complexities that are closer to the problem being considered while does not need to struggle with problems already solved by the lowest layer; then the third layer does the same and so on so that different levels of abstraction are added as needed until the problem is solved. This approach allows that complex problems are addressed with relatively simple programs. Almost any complex system is built like that (e.g. operating systems, computer networks, etc.). The architecture for building ontology-oriented models for distributed- collaborative environments proposed by this thesis also follows this layered approach.

The lowest level of abstraction is to have a "common language" that will let a reasoner to understand the basic concepts. This layer is implemented by means of a Core ontology. In this ontology, the basic concepts are defined. And the basic concepts are those that encode the information required. Namely:

- *Feature*, as in the classical definition of it, it represents the minimum amount of information that makes sense by itself in a model. It represents the objects contained in the model and it is composed of geometry, attributes, and relationships with other features.
- *Attribute*, is a property of a *Feature* and it is used to characterize it in some aspect. For instance, a road *Feature* might have a "capacity" attribute which holds the amount of traffic the road can hold.
- *Geometry*, as usual, defines the shape of a Feature.
- *Relationship*, defines a relationship of an instance of a *Feature* with another instance or instances of *Feature*.
- *Problem*. The goal of the system is being able to detect issues in the design. The concept of Problem represents those issues. It is simply something that is not correct and can be solved.
- *Conflict*, because the data model is geared towards distributed-collaborative design environments, a *Conflict* refers to a situation that appears correct for a given member but it is seen as a *Problem* by another member's eyes. In other words, it is a *Problem* in a design that other participant in the project than such designer detects because it clashes with his intentions.

A *Feature* may or may not have *Geometry*, *Relationship*(s) or even *Attribute*(s). But *Geometries*, *Attributes* and *Relationships* are associated with a *Feature* and cannot exist without the *Feature* they are attached to.

Once the Core ontology is completed and the basic concepts are defined, it is possible to proceed introducing higher levels of abstraction required by each discipline. Then, higher layers will introduce more meaningful concepts that are of interest of each discipline involved in the project design. In the architecture proposed, the term *Profile* is used as a short for these design disciplines. Figure 10.2 depicts schematically the layered structure of the architecture proposed.

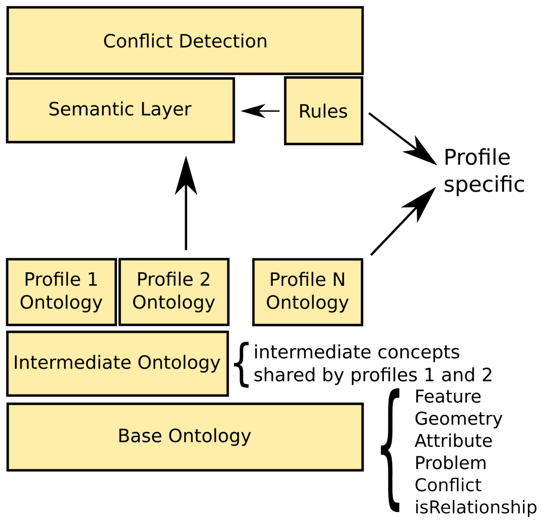


Fig. 10.2: Semantic Knowledge Architecture with shared ontologies

It is worth mentioning that intermediate and shared ontologies among profiles are also accepted. Thus situations like the one in figure 10.2 where several profiles benefit from an intermediate ontology defining concepts shared by both are perfectly possible.

Finally, the definition of the semantics is completed with the specification of SWRL rules. Used in combination, the conflict detection is performed by a reasoner which transverses the model and evaluates the Features and their components against the ontologies and rules.

SWRL rules follow the grammar shown in figure 10.3:

As it can be seen, an SWRL rule consists of an antecedent and a consequent. Both antecedent and consequent of a rule are composed

$$\begin{aligned}
\mathit{Rule} &\leftarrow \mathit{Antecedent\ RightArrow\ Consequent}\\
\mathit{Antecedent} &\leftarrow \mathit{ListOfTerms}\\
\mathit{Consequent} &\leftarrow \mathit{ListOfTerms}\\
\mathit{ListOfTerms} &\leftarrow \mathit{Term} \mid \mathit{Term} \wedge \mathit{ListOfTerms}\\
\mathit{Term} &\leftarrow \mathit{TermName} \mid \mathit{Namespace\ Colon\ TermName\ ListOfTerms}\\
\mathit{ListOfArguments} &\leftarrow \lambda \mid \mathit{Argument} \mid \mathit{Argument\ Comma\ ListOfArguments}\\
\mathit{Argument} &\leftarrow \mathit{boolean} \mid \mathit{string} \mid \mathit{number} \mid \mathit{Identifier}\\
\mathit{Identifier} &\leftarrow \mathit{questionMark\ identifierName}
\end{aligned}$$

Fig. 10.3: SWRL Rule grammar.

of atomic assertion terms. The assertion terms in the antecedent side of the rule compose a series of logic AND-like operations that are evaluated sequentially. If all assertions pass, then the antecedent resolves to true. When a rule's antecedent resolves to true then the consequent becomes true as well. That is to say, the atomic assertions the consequent is composed of also become true.

Terms have the namespace separated from the term's name by a colon (':') that corresponds to the ontology namespace. If the namespace is skipped, it is assumed it refers to a concept defined in the Core Ontology. Like in many different programming languages, the namespace prevents name clashing if two different terms happen to have the same name but come from different ontologies or have different purpose. A term might take a different amount of arguments, which in turn can be the classical boolean, number and string constants or an identifier which is preceded by a question mark ('?') symbol. Constants are simply a one-size list of values with the value defined by itself. Identifiers behave as containers that have values they have been previously assigned. A term takes arguments and can read values from constants and identifiers. A term can also write values to identifiers. Chaining terms with their respective arguments is how information is passed along the rule as it is executed.

This mechanism can be used to infer knowledge that has been hidden in the model. For instance, consider the following model of a city land use shown in figure 10.4 composed of a River (blue polygon), Roads (black lines), Parcels (light blue, green and red polygons) and Buildings (polygons with orange shell)[7]. River, Road and Parcel are

[7] For the sake of simplicity and understanding this model is an oversimplification of the use case described in next chapter.

types of Features that have been defined in some Profile-specific ontology to represent a Profile-specific concept.

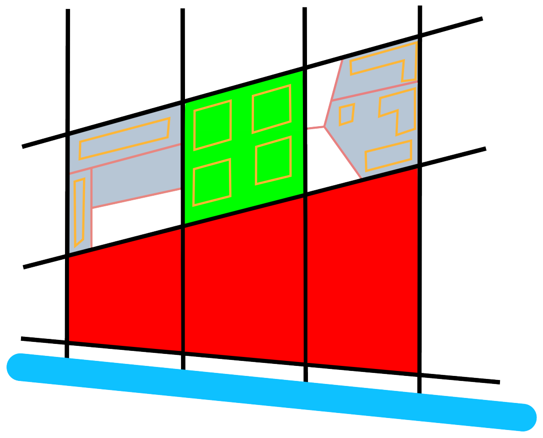


Fig. 10.4: Schematic representation of a city with roads and parcels

Given this model, the rule in figure 10.5 infers which road is associated with each parcel on that model. The association is inferred based on proximity criterion evaluated by the term. Step by that is how this rule gets executed.

$$iwcatalog : Parcel(?p) \wedge iwcore : hasGeometry(?p, ?pg) \wedge \\ swrla : closestRoad(?pg, ?r) \rightarrow isServedByRoad(?p, ?r) \wedge \\ iwroleroad : roadServesTo(?r, ?p)$$

Fig. 10.5: Example of SWRL rule

1. $iwcatalog : Parcel(?p)$, transverses the model and puts in identifier p all the Features that are Parcel. For this, it is required that the type Parcel is defined in any of the ontologies as a subclass of Feature.
2. $iwcore : hasGeometry(?p, ?pg)$, puts in pg the associated Geometry of the Parcel feature in p. Since identifier p contains Parcel as values, this term is executed for each Parcel in it.

3. $swrla : closestRoad(?pg, ?r)$, is a **user-defined operation** that computes euclidean distance between a Geometry and another Feature and puts in the identifier *r*.
4. $\rightarrow$, if any identifier on the consequent was not initialized or it is empty, then the antecedent is resolved to false and any temporal knowledge created is discarded. The rule stops executing here. Conversely, if all the identifiers have been initialized and have at least one value, then there is at least one combination that causes the antecedent to resolve to true. For every such cases, the consequent is executed but in this case, the terms are used to state facts instead of evaluating them.
5. $iwroleroad : isServedByRoad(?p, ?r)$, this term becomes a fact. And the fact is that the current combination of Parcel (p) and Road (r) have a relationship "*isServedByRoad*". This assumes that some ontology has defined the "*isServedByRoad*" concept as a subclass of "*isRelationship*".
6. $iwroleroad : roadServesTo(?r, ?p)$, similarly, this term creates the inverse fact stating that the given Road (r) is serving the given Parcel (p).
7. Finally, the facts obtained in the consequent become part of the model. In this case, new relationships that were not present have been inferred.

The mechanism can also be exploited in combination with other rules to find issues of interest in the model. For instance, a Conflict. Consider the rule in figure 10.6 as an example of how can they be used to detect Conflicts.

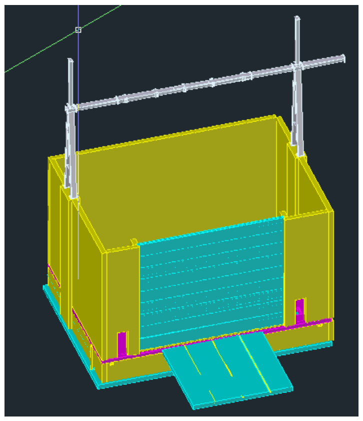
$$\begin{aligned} iwcatalog : Road(?r)\wedge \\ iwcatalog : Building(?b)\wedge \\ iwroleroad : isServedByRoad(?p, ?r)\wedge \\ swrla : isRoadExhausted(?r)\wedge \\ iwrolebuilding : isLocatedAt(?b, ?p) \rightarrow iwroleroad : RoadExhaustedConflict(?r, ?b) \end{aligned}$$

Fig. 10.6: Rule for detecting overloaded roads

This rule uses the knowledge inferred in the previous rule to find the Parcels being served by each Road in combination of the term defined as $isRoadExhausted(?r)$ to infer what roads are holding more traffic than they should and the $isLocatedAt(?b, ?p)$ term which detects what buildings are located in those parcels. If after evaluating the antecedent, symbols r and p have values, then a $RoadExhaustedConflict$ (defined in iwroleroad ontology) has been detected.

10.3 Structuring Conflicts

Now that it is possible to detect conflicting situations by applying a reasoner to the ontological model a good question is how to arrange the Conflicts in order to be processed in the future. The most basic approach is to just put all the conflicts detected by the reasoner in a list and solve them one by one sequentially. Consider the example in figure 10.7 that shows a design conflict between two hypothetical Profiles. As it can be seen in the magnified image (figure 10.7b) the bolts in the green structure do not fit in the holes that were prepared for them in the gray structure. This could be captured as a list of three conflicts of type, say, $BoldDoesNotFitConflict$. So, correcting the model is just a matter of solve each of the three conflicts.


a) Sample of a conflicting situation

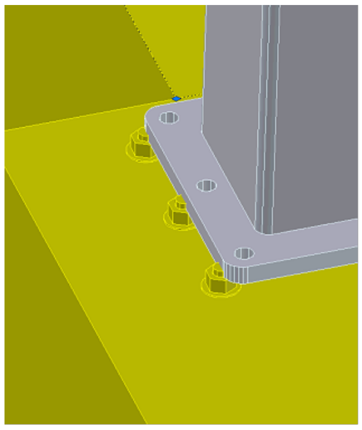

b) Same conflicting situation magnified

Fig. 10.7: Example of a design conflict.

Conflicts could also be organized in a tree structure and take advantage of the reasoner to detect potentially more meaningful conflicts. For instance, it would be perfectly possible to consider that what happens in the situation depicted in figure 10.7 is not [only] three different $BoldDoesNotFitConflict$ but that is the gray structure what is misplaced. After all, it is not only one bolt that is not fitting but all of them. If a conflict called, say, $StructureDoesNotFitConflict$ is defined by one Profile's ontology, it could exist a rule that under certain conditions (e.g., all the bolts are misplaced in the same way) promotes a set of $BoltDoesNotFitConflict$ into a $StructureDoesNotFitConflict$. Then, when correcting the model, only one conflict would need to be solved instead of three sequential ones with the corresponding saving of effort.

To conclude this chapter, it has to be said that the next step is correcting the conflicts that have been detected. The conflict solving is covered in the next chapter by means of considering the mathematical formalization of the mechanism already used manually by engineers, the negotiation. The key aspect is that, being formal, it can be replicated in a computer system and provides some interesting properties that will be explained.

Chapter 11
Negotiation As A Mechanism To Solve Conflicts

In real life, the resolution of a conflict in a distributed collaborative work environment is not a trivial process. When conducting a negotiation, the human actors internally evaluate the state of the situation and according to each individual knowledge, preferences, and interests; they decide what their limits to concessions are. All these considerations occur internally in the actor's mind, maybe even unconsciously. Given the situated nature of Civil Engineering projects, it is virtually impossible to capture all these considerations. Hence, high-level conflict solving remains an unsolved problem. As an example, it is not unusual that when the route of a road is being decided, the selected route is not the shortest or the least costly -as common sense could suggest- but the safest or the one that preserves some particular wildlife in the area it is to be built. However, it is very unusual to find Computer Aided Engineering (CAE[1]) applications that even consider this kind of conditions. Now, CAE applications typically focus on non-situated aspects like structure calculus, road slope and curvature, etc. The reason is that, as already mentioned, the variables to consider are so many, some of them are instinctive or even unknown *a priori* that it turns out to be a utopia to capture them in a traditional software package.

Because it is not possible to capture them, the possibilities of creating tools that can solve high-level conflicts in a general way are seriously reduced. The natural approach to solve problems in the software industry is by means of writing algorithms that define recipes to solve a problem. However, it is too difficult to write algorithms, if not impossible, for which the input is unknown. Moreover, because of the situated nature of Civil Infrastructure projects, how a conflict is solved depends on the concrete situation. Thus, because the conflict problem to be solved is not known, an algorithm to solve it can hardly be writ-

[1] CAE applications are a newer generation of CAD systems that enhance the standard drawing-only capabilities with discipline-specific engineering capabilities. For instance a CAD system that is geared to design aircrafts and, in addition to the drawing, it is capable of computing the distribution of forces along a wind can be classified as an aeronautics CAE system.

ten by any programmer no matter how many civil engineers assists him with their expertise and experience in his work. Consequently, any traditional algorithm will inevitably suffer from the Programmer's Dictatorship (in this case, a programmer and a civil engineer's) problem as described in section 9.3.

In real life when a non-trivial conflict appears, the people involved negotiate an acceptable solution. Negotiation emerges as the only known mechanism that is capable to perform high-level decisions. Unfortunately, this does not take the current status any further since it is how engineers solve high-level conflicts in Civil Infrastructure projects already. It has not been yet possible to capture this process in a computer system. Since a conflict does not repeat exactly (and due to these differences, an algorithm solution is not feasible) engineers can only negotiate it manually.

However, the formalization of the Bargaining Problem [Nash, 1950] allows us to consider the negotiation in a methodological way. The Bargaining theory provides a normative tool to solve a bargaining problem which elegantly avoids the Programmer's Dictatorship problem. These properties qualify Bargaining Theory as the key that can unlock the barriers that have prevented the advent of more advanced software tools so far. That is why, it is proposed by this thesis. A short introduction of the Bargaining Theory is given as it roots the approach followed by thesis. It is not, however, the purpose of this thesis to give a detailed coverage of the Bargaining Theory. A deep exposition can be found in [Nash, 1950] and [Thomson, 1994]. For a deep insight, the reader is asked to read them.

11.1 Introduction to Bargaining Theory

The bargaining process is a mechanism where agents[2] solve a conflict by reaching an agreement. There are different bargaining types based on the way the agents behave. It is possible to model competitive behavior where selfish agents might try to mislead other agents in order get better individual payoff; strategic behavior in which agents can create alliances with other agents if that increases the outcome; or

[2] "agent" as in the Nash's definition.

collaborative-cooperative behavior in which the agents work together for a common benefit. According to the ethnographic research this is the behavior that engineers have. That is why this thesis centers in collaborative-collaborative bargaining.

According to Nash's formalization, in a bargaining process there are two or more agents trying to reach an agreement on how to distribute their utility of a given conflicting situation. Every possible solution for a conflict renders an amount of utility that measures the satisfaction of an agent. The bigger the resulting utility is the more satisfied an agent becomes. Thus, bargaining agents try to agree in a solution that satisfies them as much as possible. For a given conflict, its solution is a choice on a set of feasible solutions. If the agents cannot reach an agreement, they will get the pay-off of the disagreement point that, by definition and for convenience, always belongs to the set of feasible solutions. The disagreement point represents the best external option that each agent has without reaching agreement on the set of feasible solutions. In other words, if no agreement is reached, the agents obtain the pay-off they would take without negotiating, i.e. the payoff of an unsolved conflict[3].

A bargaining problem is a mapping $f\colon \mathfrak{R}^n \to \mathfrak{R}$ that takes a set of parameters representing a hypothetical agreement situation and yields a value representing the degree of satisfaction for such situation and where n corresponds to the number of agents involved[4].

Figure 11.1 shows a bargaining problem involving two agents, the simplest case, graphically. Hence it is shown for $\mathfrak{R}^2$. It could also be shown for $\mathfrak{R}^3$. In that case, it would look like a 3D plot. However, for $\mathfrak{R}^n$ with $n > 3$ agents, it cannot be graphically represented in an intuitive way.

Each agent's utility is represented by an axis in the Euclidean space. The disagreement d point is where agents get the disagreement payoffs. This it is not a desirable solution, or formally, it is not an *efficient solution*. Conversely, the point that maximizes all agents' potential utility is known as the *utopia* and it is represented by the point $m(s)$.

Now, Nash agents define the utility they will obtain in front of a hypothetical agreement situations (i.e. their level of satisfaction asso-

[3] Although this is the normative definition, for the sake of simplicity this thesis the disagreement point is considered to have no benefits for the agents along this thesis.

[4] Thus, for instance, a negotiation among three agents is a subset of the three-dimensional space.

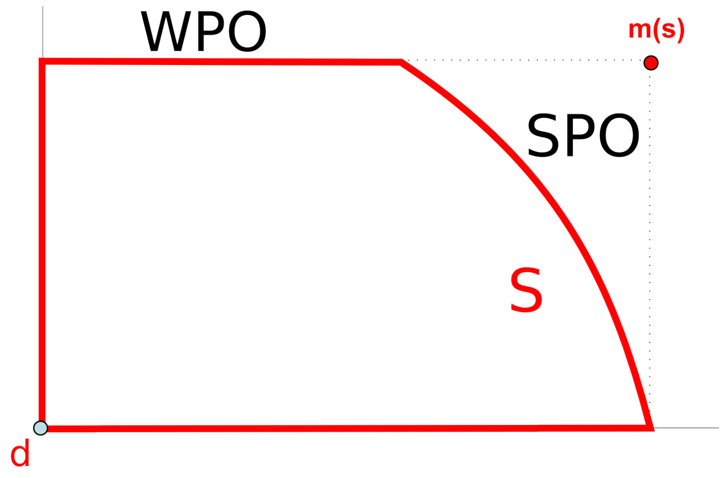


Fig. 11.1: A basic Bargaining Problem between two agents. Utility of Agent 1 and 2 are represented by the abscissa (x) and ordinate (y) axes respectively.

ciated with that agreement) by means of a utility function u that takes an alternative as input and returns a value of utility. From every agent's utility function, Nash builds up a Bargaining Problem in an abstract mathematical manner. The set of hypothetical situations is known as the Set of Feasible Solutions, or briefly, the set of Solutions and it is denoted by S. Thus, the bargaining problem is formally a pair (S,d) where S is a subset of the n-dimensional Euclidean space, and d is the disagreement point of S.

Let $\sum$ be the class of problems such that S is:

i) convex; if two points $x,y \in S$ the convex combination between those points also belongs to S,
ii) comprehensive; (S,d) is *d-comprehensive* (meaning that if $x \in S$ and $x \geq y \geq d$ [i.e., y lies between x and d], then necessarily $y \in S$),
iii) closed; S contains its boundary,
iv) there is at least one point of S strictly dominating d.

In the bargaining problem, agents try to find a solution that is as much efficient as possible. Considering this formalization the problem is solved as follows:

If $m(s) \in S$, i.e. the *utopia* is a feasible solution, then it is clear that this is an efficient solution that is desirable for all agents in the bargaining problem as it satisfies every agent's maximum expectations in the negotiation. If, as in the example shown in figure 11.1, the utopia is not a feasible solution, i.e. $m(s) \notin S$, then Nash shows that the efficient solution is a point lying in the frontier of S (except, of course, the d point since by definition of *disagreement*, it is not efficient), also known as the set of optimal solutions.

Within the set of optimal solutions, i.e. the frontier of S, two different situations can occur:

1. if when moving along the frontier an agent can increase its utility without affecting the other agents' utility, then it is said that this is a region of *Weak Pareto Optimality* (WPO) (see figure 11.1). Naturally a WPO region is a segment orthogonal to the agent's axis of the problem. Or, conversely,
2. if when moving along the frontier every gain of utility of an agent comes at a cost in the utility of other agents, then it is said that this is a region of *Strong Pareto Optimality* (SPO) (see figure 11.1). A SPO region can be a straight line or a curve depending on the problem. When the SPO is straight it typically models problem where the bargaining consists on distributing a resource and when that resource is consumable and limited. Otherwise, the SPO region is typically a curve.

11.1.1 An example of bargaining between two friends

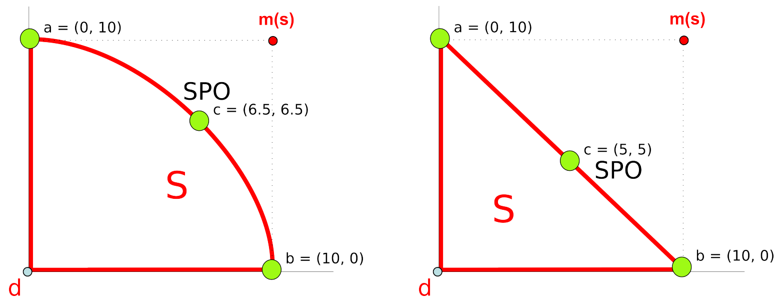
Consider the following example in order to better understand the concept of utility.

Two friends meet in a bar a Friday after work and have to decide what to do with two pints of beer. In a problem like this, they typically decide to have one beer each. This sounds very reasonable and, unless they are not so good friends as they think, both will be satisfied with such a distribution. That is because the utility of having one beer each is bigger than one friend having two and the other having none.

Now imagine that the situation is a bit different. Instead of meeting in a bar after work, the two friends happen to be the only two survivors of a shipwreck on a tiny island with no fresh water nor supplies and an unclear chance to be rescued in the following days. In a situation like that, it is not that obvious that the two friends agree on having one beer each as there is a considerable chance of death by dehydration before they get rescued. For sure, having two beers instead of one can determine their survival.

This simplified example illustrates what the utility is. While there is a tendency to associate utility with money (benefits/costs), utility refers to a more abstract concept (which of course can involve money too). From the bargaining problem point of view, the difference of these two situations is that while the first case negotiates how the two friends can enjoy most of the evening, the latter case negotiates a scarce and consumable resource. In other words, the two situations have different goals: maximize happiness (through the beer but happiness anyway) vs. maximize pints of liquid (beer in itself). Figure 11.2 shows graphically the two problems. The problem of sharing the beers in a bar is depicted in figure 11.2a. The SPO is curved because it is assumed that, as it can be imagined, in combination both friends enjoy the evening more if bath have beers compared to the case where one friend does not drink or he does significantly less than the other. Figure 11.2b depicts the other case. In this problem, the two friends are trying to survive. Assuming that the two friends will consume the liquid at the same speed so that the same amount increases the survival time equally, it is easy to see that the total amount of survival time remains constant regardless how it is distributed. Let's now see how this is mathematically analyzed. In the problem on figure 11.2a, the point c can be a more efficient alternative than a and b because the total utility obtained is $6.5 + 6.5 = 13$. This is more than the total utility that would be obtained either in a or b, that is 10. In this case, agents have more global happiness if they decide to share their drinks. In problem on figure 11.2b, however, the global utility is constant (10) regardless what agents decide to do. This is typically the case of negotiations about how to distribute a consumable resource.

Now, recall that the solution is a point that lies in the frontier. But the frontier can have many points. Which solution (i.e. which point in the frontier) is picked will depend on the policy followed. In essence, it means that it has to be known what is considered an *efficient solution*. Is it the most fair? The most globally profitable? Are the actors interested in maximizing gains or in, on the contrary, minimizing losses? In section 11.2 a set of algorithms is presented that consider



a) Friends meeting on Friday b) Shipwrecked "friends"

Fig. 11.2: The concept of utility and Strong Pareto Optimality

these kinds of characteristics, formally called *axioms*. They are some of the most known, interesting or easier to apply[5].

11.2 The Bargaining Efficient Solutions

The theory of the axiomatic bargaining is a powerful tool from the normative point of view in the sense that if a bargaining problem is well defined and their axioms are clear, then there is a complete certainty that, if there exists a theoretical characterization that matches the problem's definition, it will be applicable and it will select the most efficient solution (according to the axiom's characterization of efficiency) among all the ones in the frontier of S.

Thus, the results provided by the Bargaining Theory consist of a characterization of the solutions; more precisely, "in isolating the minimum amount of logic axioms that are as weak as possible that demonstrate the existence and *unicity* of a solution", i.e. the minimal set of axioms that reduces the set of solutions down to one: the efficient solution. This means that the goodness of a solution basically resides in the goodness of the axioms that characterize it. For the sake of comprehensibility, the solutions provided later in this section are discussed qualitatively. For an in-depth, axiomatic and technical description of characterizations of the efficient solutions, the reader can refer to [Thomson, 1994] for efficient solutions prior to 1994, and

[5] As a life research field, Bargaining Theory has, of course, others that are not considered in this thesis due to lack of time in studying their applicability.

to [Dubra, 2001] for the Weighted Kalai-Smorodinsky or to (Garcia-Segarra et al 2012) for the Weighted Losses and Weighted Proportional Losses.

Thus, depending on what is the kind of negotiation at hand (the axioms), there are different definitions for finding the efficient solution of a bargaining problem. Some possible bargaining efficient solutions are listed here and graphically represented in figure 11.3:

1. Egalitarian [Kalai, 1977]; also known as Equal Gains Solution, is the maximal point in the bargaining set S of equal coordinates. This point is defined by the equation 11.1.

   $$E_i(S) = E_j(S), \forall i, j \tag{11.1}$$

   This solution distributes gains equally for each agent in the bargaining. Conceptually, it equalizes gains from the disagreement point by assigning the max point of equal coordinates that belong to the set S.
2. Nash [Nash, 1950]; is the maximisation of the product of all agents' utilities. The algorithm that finds this point is defined by equation11.2

   $$N(S) = argmax\{\prod_{i=1}^{n}(u_i - d_i) \mid u \in S\} \tag{11.2}$$

3. Kalai-Smorodinsky [Kalai and Smorodinsky, 1975]; Is the maximal point of the set S on the segment connecting the origin (disagreement point d) with the utopia point ($m(s)$).

   $$KS^{\alpha}(S) = argmax\{k \in \Re_+ \mid \frac{u_i}{u_1} = \frac{m_i(S)}{m_1(S)}, u_1 = k, \forall i, j \in \mathfrak{N}\} \tag{11.3}$$

   It gives each agent a solution that is proportional to its own utopia. This solution was proposed to avoid situations in which one agent is damaged when the set of the solutions improves. Some definitions have an unexpected behavior when the set S grows. The unexpected behavior consists in that an actor can actually get less pay-off even when there is actually more to be distributed. The authors of this solution did not find this behavior reasonable as they considered that if there is more to distribute, everybody should, at least get the same than when there was less. This solution does not show that behavior (eq. 11.3).

4. Utilitarian [Moulin, 1983]; is the maximisation of the sum of all utilities (eq. 11.4).

$$U(S) = argmax\{\sum_{i=1}^{n} u_i \mid u \in S\} \tag{11.4}$$

It maximizes the sum of the utilities of the agents. Its philosophy is to search the maximum amount of utility for the maximum amount of agents.

5. Equal losses [Chun, 1988]; is "the point on the Pareto frontier (the frontier except d) where all agents make the same concessions" (eq. 11.5).

$$EL(S) = max\{x \in S \mid m_i(S) - u_i = m_j(S) - u_j, \forall i, j \in \mathfrak{N}\} \tag{11.5}$$

It takes a dual approach to Egalitarian but focusing on losses instead of in gains. Hence, it distributes losses in respect to the utopia point equally for each agent in the bargaining.

6. Dictatorial [Arrow, 1950]; which corresponds with the maximal dictator agent's coordinate that belongs to the bargaining set (eq. 11.6).

$$D^i(S) = argmax\{u_i \mid u_j = 0, \forall j \neq i\} \tag{11.6}$$

This solution simply maximizes the utility of one agent (the dictator). There are actually two flavors of this solution: 1) the Dictatorship that maximizes its utility without considering other agents benefits and 2) the Lexicographic Dictatorship that also maximizes its utility but allows other agents to increase their own so long the Dictator's utility is not affected.

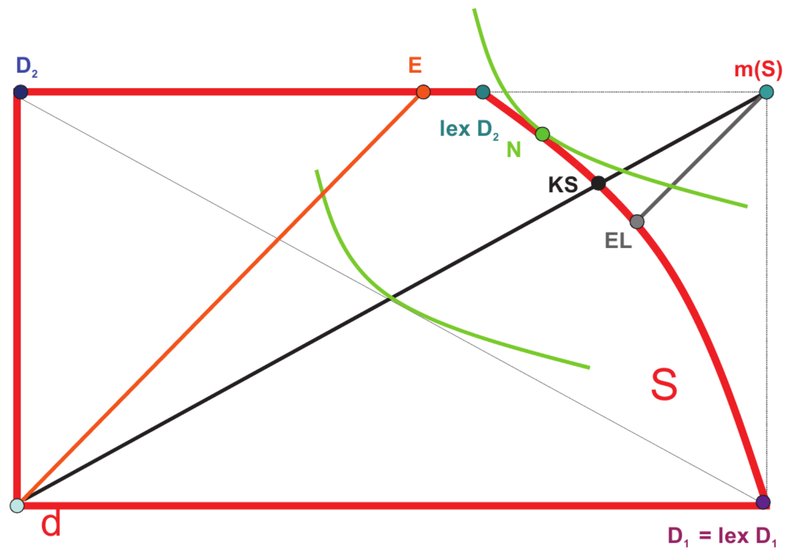


Fig. 11.3: Graphical representaiton of the symmetric bargaining solutions for two agents. The different efficient solutions are depicted by the points in the frontier: Dictator when actor 2 is the dictator (point D_2), Egalitarian (point E), Lexicographical Dictator when actor 2 is the Dictator (lex D_2), Nash (N), Kalai-Smorodinsky (KS), Equal Losses (EL), Dictator when dictator is actor 1 (D_1) and Lexicographical Dictator when dictator is agent 1 (lex D_1) (used with the permission of Jaume Garcia-Segarra).

11.3 Asymmetric Bargaining Power

Another aspect that can be considered in a negotiation is the bargaining power of the agents. In general, a negotiation is performed by agents who do not necessarily influence the result of it. It will depend on the hierarchy established in the negotiation. This is largely accepted and known as *asymmetric bargaining power*. A typical example in Civil Infrastructure for this case is when a project manager and an engineer are negotiating something regarding the project. The engineer may influence the result of the negotiation, but the manager's opinion is more relevant. Moreover, a third agent like the politician (the customer) can have the last word and the decision being made is what he or she says regardless what others say.

The bargaining power is modeled by means of a parameter $\alpha \in [0,1]$, i.e. a factor of relativity, defined for each agent such that when

$\alpha_i = 0$ the i-th agent has no bargaining power, and when $\alpha_i = 1$ the i-th agent has all the bargaining power, becoming a dictator agent. The sum of all the bargaining powers is always 1. In other words, $\sum_{i \in \mathrm{n}} \alpha_i = 1$, where is the bargaining power of the *i*-th agent. Given this definition for α , the equations above can be rewritten to support asymmetric bargaining power as follows:

1. Proportional (Weighted Egalitarian) [Kalai, 1977] (eq. 11.7.

$$E^{\alpha}(S) = argmax\{k \in \Re_+ \mid \frac{u_i}{u_1} = \frac{\alpha_i}{\alpha_1}, u_1 = k, \forall i, j \in \mathfrak{N}\} \quad (11.7)$$

2. Weighted Nash [Harsanyi and Selten, 1972] (eq. 11.8).

$$N^{\alpha}(S) = argmax\{\prod_{i=1}^{n}(u_i - d_i)^{\alpha_i} \mid u \in S\} \quad (11.8)$$

3. Weighted Kalai-Smorodinsky [Thomson, 1994] (eq. 11.9).

$$KS^{\alpha}(S) = argmax\{k \in \Re_+ \mid \frac{u_i}{u_1} = \frac{\alpha_i m_i(S)}{\alpha_1 m_1(S)}, u_1 = k, \forall i, j \in \mathfrak{N}\} \quad (11.9)$$

and, Dubra's [Dubra, 2001] case for 2 agents (eq. 11.10).

$$lKS^{\alpha}(S) = \{u \in \Re^2_+ \mid u \geq KS^{\alpha}(S)\} \cap SPO(S) \quad (11.10)$$

4. Weighted Utilitarian (eq. 11.11).

$$U^{\alpha}(S) = argmax\{\sum_{i \in \mathfrak{N}}(\alpha_i u_i), u \in S\} \quad (11.11)$$

5. Weighted Equal Losses [Segarra and Vilar, 2011a] (eq. 11.12).

$$EL^{\alpha}(S) = argmax\{u \in S \mid \alpha_j(m_j(S)) - u_j) = \alpha_n(m_n(S) - u_n), \forall i \in \mathfrak{N}\} \quad (11.12)$$

6. Weighted Proportional Losses [Segarra and Vilar, 2011b] (eq. 11.13).

$$PL^{\alpha}(S) = argmax\{u \in S \mid \alpha_i(1 - \frac{u_i}{m_i(S)}) = \alpha_n(1 - \frac{u_n}{m_n(S)}), \forall i \in \mathfrak{N}\} \tag{11.13}$$

7. Weighted Dictatorial. Notice that Dictatorial is asymmetric by definition, since the dictator agent has all the bargaining power. Thus both the symmetric and asymmetric versions are actually the same (i.e., eq. 11.6).

Figure 11.4 illustrates them graphically for the case of two agents bargaining (i.e., a problem in $\mathfrak{R}^2$):

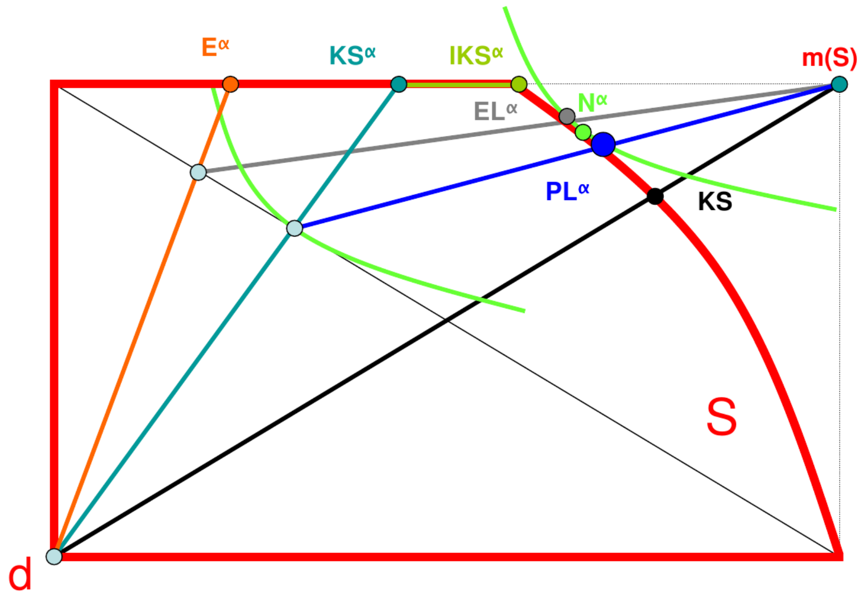


Fig. 11.4: Graphical representaiton of the asymmetric bargaining solutions for two agents. The different efficient solutions are depicted by the points in the frontier: Weighted Egalitarian (point E^{α}), Weighted Kalai-Smorodinsky (KS^{α}), Dubra's (lKS^{α}), Weighted Equal Losses (EL^{α}), Weighted Nash (N^{α}), Weighted Proportional Losses (PL^{α}) (used with the permission of Jaume Garcia-Segarra).

11.4 The problem of utility functions

Conceptually, utility functions are an instinctively elegant and intuitive way to develop a mathematical model for negotiation because they capture the subtleties of each agent in the negotiation in a mathematical expression that returns the level of satisfaction the agent has in response to a possible solution. Since the set S is defined by the agents' utility functions, it is a requirement that agents define them. The big downside is, as any intuition-driven measure, it can easily become near to the impossible to provide the analytical expression for them. In other words, it is not realistic to expect somebody to describe his tastes in a mathematical formula about an arbitrary topic unless this topic is so simplistic that it arguably deserves consideration. For instance, it could be feasible to ask somebody to provide a formula that describes his utility in choosing a piece of bread among all the varieties of bread. This formula could take aspects like price, type of corn and maybe a few other simple attributes by associating numerical values to each possible value for the attribute and accumulating them. But, is this applicable to other more complex scenarios like those in Civil Infrastructure? That is why utility functions are strongly criticized as they are. It could be suggested that statistics could help here. We could imagine having sets of individuals as the population under study where in front of a choice situation they were asked to make a decision. Then, registering and analyzing the results in order to find correlations, a regression or similar that could discover a hidden formula. While this would follow a scientific method in its pure and genuine essence, unfortunately, it seems far too much idealistic. The reason, again, is that of the situated nature of the Civil Infrastructure projects. Given a design conflict leading to a negotiation process, what are the utility functions that define the set of possible solutions? Due to the situated nature, they will depend on each concrete conflict. Even the set of parameters of the hypothetical utility functions will most probably vary from each conflict to one another (and so will the utility functions) since they refer to different -potentially incompatible- situations. Next section considers these issues and adapts the pure Bargaining Theory to Civil Infrastructure conditions. The current situation is pushed forward to a more practical and applicable point closer to what the Civil Infrastructure needs. So, the foundations on which Part II of this thesis is based are grounded.

11.5 Building a Computer Model for Solving Civil Infrastructure Design Conflicts

This thesis focuses on the negotiation of Design Conflicts. According to the ethnographic research described in Chapter 5, engineers most often negotiate in a cooperative-collaborative way[6]. As in any other environment, all styles of negotiation are potentially possible in Civil Infrastructure depending of the project's concrete settings and the situation at hand. For instance, some of the engineers planning a project can enjoy more power (the α parameter, see section Asymmetric Bargaining Power) than the others if the latters belong to companies that have been subcontracted by the formers. Another example of asymmetric bargaining power would be the case when the decisions of a given discipline have more impact in the overall project than another. In a case like that it is natural to think that engineers in the critical discipline will have more power in accordance to their level of reponsibility. Actually, any combination can occur. Fortunately, by applying Bargaining Theory the method for negotiating remains the same. That is, agents describe their utility and, depending on the bargaining solution chosen, the result is computed. Therefore, a conceptually valid solution for a concrete type of cooperative-collaborative negotiation can be used to model a system without any loss of generality. The rest of the method would remain unchanged. In other words, a system implementing one of the Bargaining Theory solutions for a given setting is still generalizable to any kind of cooperative-collaborative negotiation focusing on some other aspects (e.g. equally-distribution of losses among participants) because it just needs to choose the algorithm that decides which the result of the negotiation is among those ones already modeled in Bargaining Theory[7]. So, in case other types of negotiation were to be supported by a hypothetical computer system, then the task to extend such system would consist of finding the suitable solution among the available efficient solutions that fits into the new type of negotiations according to their properties (or axioms

[6] There are other approaches that model other types of negotiations such as cooperative-competitive, strategic, etc. Since the ethnographic research pointed out that they are not the most common case, they have been left out of the scope of this thesis

[7] supported by this fact and for the sake of applicability, in the experiments carried out in this thesis only Nash and Utilitarian solution are used to deal with a certain type of negotiation (as it will be shown). It has to be noticed, however, that this does not prevent to use any of the others.

as they are formally referred in the literature; also in the previous sections).

Some differences would exist in the scope of an application in respect to the canonical bargaining theory [Thomson, 1994]:

First, agents have a common goal: the development of the project. Then, the aspects regarding the distributive fairness are put aside since the common goal presses the agents towards the cooperation. Consequently, the properties related to monotonicity[8] are less relevant.

Another important feature, as mentioned above, is that given the situated nature of Civil Engineering projects, it is difficult to capture the set of solutions S and their boundaries. In the Civil Infrastructure, domain the problem of the utility functions becomes unacceptably serious since only a small portion of the virtually infinite set of conflicting situations can provide solutions. Thus, if defining a utility function for a concrete problem (i.e. a concrete conflicting situation like, for instance deciding what to do with a light pole that is too close to a manhole) is difficult, then asking for utility functions for any kind of conflict that might appear on the Design Phase, is simply impossible. A system aiming to be general cannot accept this limitation whatsoever.

11.6 Using Alternative sets instead of utility functions

A possible way to overcome the serious problem of how to define utility functions for a given conflict is to *emulate* that abstract -or too abstract- set with a set of more concrete samples as a means of building the set of solutions S. These concrete samples will be a set of possible concrete alternatives to a conflicting situation. Technically, S will not be a set of solutions anymore but a set of alternatives. But this difference is more philosophical than practical in the sense that the bargaining mechanism is essentially the same. That is, instead of asking each agent's utility function, they can be approximated by asking

[8] Monotonicity means that if we have a solution for an initial set of utility and this set of utility grows, then the new solution on the new set must be greater or equal than the previous one. In other words, if a pie is to be shared by agents and the agents agree on how much each gets then if the pie becomes bigger they will get at least as much pie as they got before the pie grew. It is very convincing to argue that, if the feasible set increases, the payoff of no agent should decrease [Segarra and Vilar, 2011b] (see also Kalai-Smorodinsky's efficient solution at equation 11.3).

each agent for alternatives to the given conflict. In other words, before the negotiation actually starts, each agent is asked “how would you, as an individual, solve this conflict? can you suggest alternatives for it?”. Then, all the concrete alternatives are firstly collected and afterwards, in a second round, all agents can be asked to express preferences on every alternative suggested. Then, their preferences on concrete alternatives are used to solve the conflict.

When every agent expresses its preferences (i.e. the utility a given alternative renders to the agent) by giving a score within a known interval, a set of points in the n-dimensional Euclidean space is obtained. Formally, a bargaining solution now becomes a mapping $\mathfrak{R}^n \rightarrow \mathfrak{R}$ between a set of alternatives and the outcome of the solution. For practical reasons -and also because it is a condition to use Nash’s solution-, the preference values need to belong to a known and scaled set. For the sake of comprehensibility, this thesis defines this scale as the interval $[0,10]$ as it is an often used scale to valuate things. However, the scale to be used can be an arbitrary one so long it belongs to $\mathfrak{R}_+$ since it is a requirement of some of the solutions (see section 11.2) that the utility $u \geq 0$. Nevertheless, such a constraint is not very restrictive since the bargaining solutions used in this work satisfy *scale invariance* or *translation invariance* (which implies scale invariance). Thus, any initial situation can be easily transformed to satisfy the need for a scaled and positive values for the preferences[9].

These concepts are the theoretical foundations for the system that is proposed in the next chapter (the chapter is actually a published article). The system is a real implementation of this approach in a computer system. This system is implementing a simple CAE tool that simulates those used by the enginners (naturally, without the same powerful options since its only goal is to test the ideas exposed in this and the previous chapter). This application is part of a distributed system developed on purpose in which there is a simple BIM server with the capability to keep a central model and which accepts concurrent modifications. Engineers can perform designs cooperatively working against the BIM server by means of instances of the prototype application. The interaction among the pieces is performed by agents as they

[9] For instance, when the resources being allocated are tasks, it is more natural to express preferences by means of negative utilities (i.e. costs of the task), in such cases the problem can be easily flipped into positive utilities by applying a negative scale of, for instance -1.

are defined by Wooldridge. It means that the discourse of the thesis will now gradually move from Economic Science (the science of the decision) to Computer Science and Artificial Intelligence. The concepts exposed in this chapter are used and extended with others like communication protocols among a society of agents that allow these ideas to be implemented in a real tool. The society of agents mimics the "society of engineers working collaboratively" that has been modeled based on the findings derived from the ethnographic research. In this society, three types of agents are utilized. There is one Coordinator agent representing a model manager who is in charge of conducting the communication protocol, briefly sketched in this chapter, coordinating the other agents in a negotiation to solve design conflicts. The Coordinator agent is also in charge of deciding what will the solution be among the available ones according to the preferences expressed by the engineers, by using the concepts presented in this chapter regarding efficient solutions. There are also Validator agents which explore the model the engineers are working on in parallel. Validator agents exploit the concepts exposed in Chapter 10, namely the ontologies and rules, so that they know what a conflict is according to the discipline they represent. Finally, there are the Negotiator agents which enable the human engineers to interact with the Coordinator agent by implementing the negotiation protocol. Each engineer has one Validator and one Negotiator agent. Together, they constitute what is called a Profile showing how a complete system exploiting these concepts can be.

Chapter 12
[Article] Multiagent System for Detecting and Solving Design-time Conflicts in Civil Infrastructure

Jaume Domínguez Faus[1], Francisco Grimaldo[2], Fernando Barber[2]
[This article] Appears in: Trends in Practical Applications of Agents and Multiagent Systems issue of *Advances in Intelligent and Soft Computing* (formerly known as "*Advances in Soft Computing*"), 154, Springer-Verlag. 2012.

Abstract One typical source of problems in the Civil Infrastructure domain is the distributed and collaborative nature of the projects in which different profiles of engineers contribute with designs devoted to the interest of their field of expertise. Thus, situations in which there are different conflicts of interests are quite common. A conflict refers to a situation in which the actions of an engineer collide with the interests of other engineers. In this paper, we present a multi-agent system[1] that, thanks to the use of ontologies and rules on those ontologies, is able to detect profile-specific conflict situations and solve them according to the preferences of the parties involved in the conflict. The conflict solving is based on the Multi Agent Resource Allocation (MARA) theory. The system is applied to a real use case of an urban development where both the road network and the buildings are designed.

Jaume Domínguez Faus
Centre for 3D GeoInformation, Aalborg University. 9220 Aalborg, Denmark +45 9940 3699
e-mail: jaume@land.aau.dk
Francisco Grimaldo, Fernando Barber
Departament d'Informàtica, Universitat de València, Av. Vicent Andrés Estellés s/n, (Burjassot) València, Spain 46100
e-mail: francisco.grimaldo@uv.es, fernando.barber@uv.es

[1] "agent" as defined by Wooldridge

12.1 Introduction and Related work

Interoperability is a very often addressed term when enumerating the problems of distributed systems. A system may refer to a set of computers working together, but also to a set of humans that put their effort to push in the direction of solving a complex problem. Similarly, it can also refer to a hybrid set of humans and computers. In the same way that communication becomes difficult between two people speaking different languages, communication is difficult when dealing with systems relying on data for modeling a problem to solve. This happens because data only describes things and, as any description, they can be interpreted in many ways. As an intensive data consumer, the infrastructure domain is not immune to the interoperability problem. Any infrastructure project as, for instance, a road construction involves lots of disciplines ranging from land-use to security regulations, with noise emission regulations, road tracing, water drainage and many others in between.

The general tendency is to have -when possible- a data model for each discipline that is used by specific software packages to assist the daily engineer's life and split the project works in discipline-experts teams. Problems arise when all the works done by different teams have to be put together. Issues like design clashes, synchronization problems, exceeded budgets, conflicts of interests appear, etc. as a consequence of the decentralized way of working with heterogeneous data models. The detection and solving of such problems is still a prominent manual work and some of them might remain undetected when this process is finished. Once the construction starts, the consequences of mistakes or suboptimal design cause that the infrastructure cost increases a 5-10% of the total budget in average [Eastman et al., 2008].

Most of the efforts done so far have focused on avoiding the collisions by improving interoperability among different data models. It has not been, however, until recently when the conflict-solving has gained attention. This paper presents a new multiagent-based approach for detecting and solving design-time conflicts in the Civil Infrastructure domain.

12.2 State of the Art

Currently, the Civil Infrastructure software industry focuses on the creation of models that integrate more and more aspects of design disciplines in order to increase interoperability. Perhaps, the most advanced results of these efforts are the most successful standard files (such as CityGML or IFC [Kolbe et al., 2005], and AutoCAD's DWG) or the Building Information Model (BIM) servers as defined by [Eastman et al., 2008] which combine CAD models with management spreadsheets and other documents to provide an integral project life-cycle management. However, as a distributed collaborative work, it has to deal with conflicts that inevitably appear when sub-designs of a project are merged.

Multi-agent Systems (MAS) have been suggested to aid in Civil Infrastructure projects. It is possible to find examples of MAS that control construction equipment like the system described by Zhang, Hammad, and Bahnassi where sets of cranes align their movements to transport materials from one location to another in the construction area without crashing [Ren and Anumba, 2004]. It is also possible to find examples of models 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 by the owners and interested companies apply for it [Schnellenbach and Denk, 2002]; [Udeaja and Tah, 2001] focus on the construction material supply chain that is managed collaboratively. Within the Construction phase the main focus has been the formalization of expert knowledge and the negotiation mechanisms to solve unexpected situations. [Peña-Mora and Wang, 1998] presented a Game Theory approach to solve various conflict situations between the Architect/Designer/Constructor settings when each agent competes for reducing the impact of unexpected situations in its own interest. More recently Shen et al. studied the applicability of cognitive maps MAS in collaborative-competitive working on construction projects (CWCP) in which these maps are used to parameterize the agent's beliefs within the MAS negotiation [Xue et al., 2010].

Nevertheless, to the authors' knowledge there is a gap that has not yet been considered satisfactory: the negotiation between designer expertises in the Design phase of the project. Despite it is possible to find some problem-specific works like [Anumba et al., 2002], the situated nature of this collaborative work makes the problem of abstraction of a system to be wicked [Fitzpatrick, 1998]. However, this abstraction is necessary to capture the negotiation as a design conflict-solver in the software packages normally used by the engineers in their daily work. Therefore, more research is needed in this field to allow MAS to be a real integral option.

The use of ontologies has been proposed as a means to give sense and semantics to the data in several contexts. In geospatial and civil infrastructure information, ontologies are not widely used. Although it is possible to envisage ontological structures in some data models (e.g. CityGML) they are hardly used in a formal and explicit manner.

Thus, we propose the use of ontologies to support automatic conflict detection and of the Multi Agent Resource Allocation (MARA) [Chevaleyre et al., 2006] for its solving at a semantic level. In section 12.3.1 we present the ontological approach we propose to represent the world semantics, and the rules that are used to detect conflicts. Further below, in section 12.3.2 the negotiation mechanism used to solve conflicts is introduced. Due to the specific and situated nature of the projects, a methodology like this needs to be able to learn how to solve conflicts that have not been typified yet. For the sake of simplicity, the learning process is not discussed in this paper but is part of the ongoing research. Finally, in section 12.4 we describe a use case in which the system was applied in order to illustrate its usefulness.

12.3 Architecture of the System

The multi-agent system we propose for detecting and solving design-time conflicts in the Civil Infrastructure domain is depicted in figure 12.1. It follows a distributed architecture approach allowing the engineers of different profiles to design, through their client interfaces, a common BIM model that is stored in the server. This collaborative work is carried out with the assistance of a set of agents: the Validators, the Negotiators and the Coordinator.

Validator agents are in charge of semantically detecting conflicts and errors within the model by using the ontological knowledge of each field of expertise. In turn, Negotiator agents aim at solving conflicts by expressing the preferences of the engineers in a negotiation protocol that is initiated by the Coordinator under conflict notification. Following, we review the details about these agents, which have been implemented as part of an agent society in JADE [Bellifemine et al., 2007].

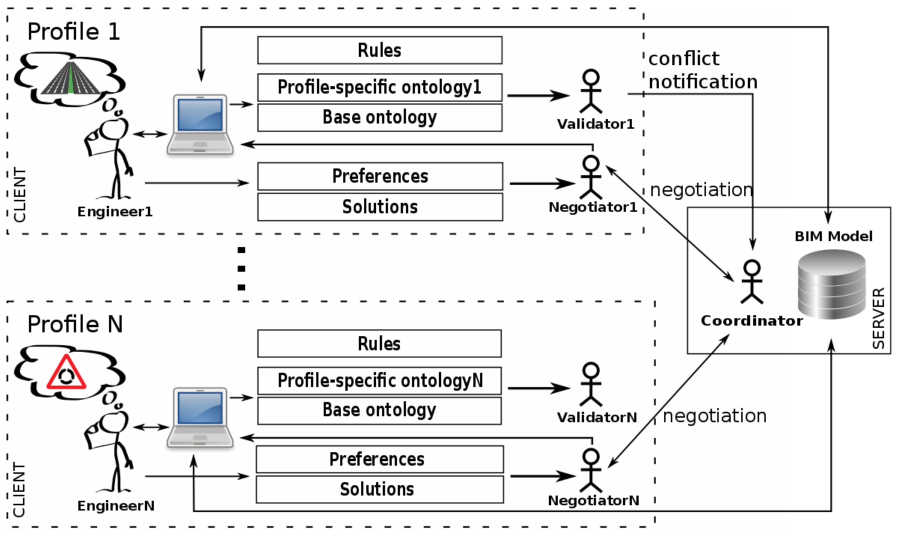
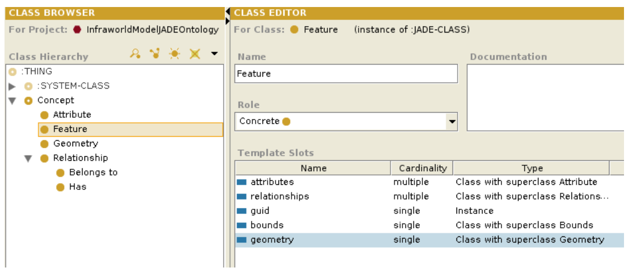


Fig. 12.1: Overview of the system

12.3.1 *Semantic Conflict Detection*

We propose using OWL [Bao et al., 2009] ontologies for the semantic abstraction of the data beyond the pure classical attribute/value pair. As shown in figure 12.1, our ontologies are structured in layers in which each layer provides an extra level of abstraction. At the lowest level, the Base Ontology defines the basic concepts needed by any geospatial data model. Figure 12.2 shows an extract of the classes defined in the Base Ontology. The class `Feature` refers to the most basic object that traditionally forms geospatial data models such as GIS or CAD systems. A `Feature` is composed of a `Geometry` and of a set of `Attributes` defined by its name, its type, and its value. `Features` can be related to each other through the generic relation `hasRelationship` and its inverse relation `isRelationshipOf`. This pattern has proven to be flexible and suitable for many uses. Besides, by using inheritance, classes can be arranged in a hierarchy (e.g. `Conflict` and `Error` are particular types of `Problems`). Therefore, this ontology acts as the first layer of abstraction allowing the creation of Profile-specific ontologies on top of it.


CLASS BROWSER
For Project: InfraworldModelJADEOntology
Class Hierarchy
- :THING
- :SYSTEM-CLASS
- Concept
  - Attribute
  - Feature
  - Geometry
  - Relationship
    - Belongs to
    - Has

CLASS EDITOR
For Class: Feature (instance of :JADE-CLASS)
Name: Feature
Documentation
Role: Concrete
Template Slots

Name	Cardinality	Type
attributes	multiple	Class with superclass Attribute
relationships	multiple	Class with superclass Relations...
guid	single	Instance
bounds	single	Class with superclass Bounds
geometry	single	Class with superclass Geometry

Fig. 12.2: Extract of the Base Ontology shown in Protégé ontology editor[Protégé, 2006]

Profile-specific Ontologies are meant to define the concepts of interest for each profile. These concepts can be specific `Features` providing particular properties and/or semantic meaning (for instance, a `Building` or a `Parcel`) and also specific relationships defining how certain `Features` relate to each other (e.g. `isLocatedAt` relates a `Building` with the `Parcel` where it is placed). This second level of ontologies allows separating the categorization of the different interests involved in civil infrastructure projects in order to ease the management of the knowledge. Note, however, that this does not necessarily prevent a concept to be shared among different profiles in case several profiles need it.

As the project progresses, the different engineers include new designs or edit the existing ones and the model changes continuously. In this dynamic context, the Validator agent automatically assists in the correctness of the model as a whole by periodically checking a set of rules defined for the profile. We propose using SWRL [O'Connor et al., 2005] rules as a way of supporting the semantic consistence and ontological reasoning. These rules are ontological expressions with an antecedent and a consequent that allows the Validator agent to detect and infer problematic situations. The problems encountered are categorized between `Errors` or `Conflicts`, according to the classes defined in the Base Ontology. `Errors` are situations in which the model is not correct due to missing or wrong values in the `Feature`'s properties and, thus, they are notified and solved manually by the engineer through its client interface. On the other hand, `Conflicts` capture the situations where the designers' interests collide and they are solved through the negotiation protocol explained next.

12.3.2 Conflict Solving Protocol

We propose using a Multi-Agent Resource Allocation (MARA) approach to analyze the possible alternatives that solve the `Conflicts`. The MARA [Chevaleyre et al., 2006] model provides agents with a general mechanism to make socially acceptable decisions. In this kind of decisions, members are required to express their preferences with regard to the different solutions that have been previously proposed by all the members for a specific decision problem. Our MARA approach uses ContractNet-like protocol as the allocation procedure. Figure 12.3 depicts the process for the case of a `Conflict` between two profiles. When the `Conflict` is detected by one Validator agent, it is notified to the Coordinator agent (message no. 1). Then the Coordinator distributes the `Conflict` to all the Negotiators in a Call For Proposals (mesg. 2 and 3). The negotiators respond with their alternatives, if any (mesg. 4 and 5) and the Coordinator collects all the proposals. In the collection, invalid or repeated solutions are filtered out and the set of remaining solutions is distributed again to request the preferences (mesg. 6 and 7). Each Negotiator then expresses its utility on each of the solutions at hand (mesg. 8 and 9) by giving it a value ranging from 0 (lowest) to 10 (highest). The Coordinator agent then picks the winner solution which is the one that maximizes the global utilitarian social welfare represented by the solution that accumulates highest utility among the negotiators. Finally, the winner solution is then broadcasted to all the clients (mesg. 10 and 11).

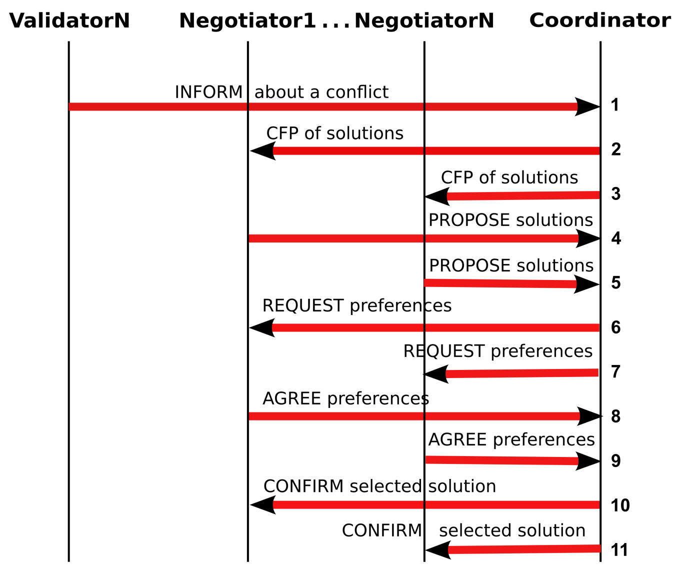


Fig. 12.3: JADE[Bellifemine et al., 2007] console showing the Conflict Solving protocol (graphics have been edited to improve the figure)

12.4 Urban Development Use Case

In order to simulate the daily work of engineers in the design phase, an urban development use case was selected. This project consists of the development of the Strømsø area in the city of Drammen, Norway. Traditionally an industrial area, after decades of growth, Strømsø became the downtown of Drammen while keeping the original industrial aspect. The authorities want to adapt it to the new residential reality. In general, the development goal is the construction of residential buildings to increase the number of inhabitants. In the initial phase, the project defines where to place buildings according to their characteristics (number of residents, floors and footprint). Further phases of the design deal with other more detailed aspects. We focused on the building placing problem to show our MAS approach.

With the aim of avoiding future traffic jams, it was agreed that there should not be more residents than the capacity of the road. Thus, the location of a building is constrained to the capacity of the road that serves the building. The current usage of the road is obtained by the sum of the inhabitants of the buildings that are associated to that road. So, in addition to their geometry, buildings and roads specify the amount of inhabitants and the road capacity respectively in their attributes.

There are two engineering profiles identified: 1) The designer that places buildings in a location of her/his choice (Building profile), and 2) The road designer that detects which road is connecting the building to the road network and checks whether the road is capable to hold all the buildings connected to it (Road profile).

For each profile there is a designer that is developing the model, and each designer has: 1) a validator agent that checks the model according to the semantics (expressed by her/his ontology and rule-set settings) and initiates the negotiation; and 2) a negotiator agent that performs the negotiation on behalf of the engineer.

12.4.1 Use Case Semantics. The Ontology System

As introduced in section 12.3.1, `Features` are the most generic object that can be defined in our ontological model (unlike pure geometry-based models where basically only the geometry is known). On top of it, we identify three specific concepts of interest for this use case: the `Road`, the `Parcel` and the `Building`. `Parcels` are `Features` that define an area in which `Buildings` can be placed. `Buildings` are `Features` representing the residential entities where people live in. In turn, `Roads` are `Features` representing the parts of the road network. Beyond specifying the type of a `Feature`, these classes define more attributes that are required to describe their characteristics such as the capacity of a `Road` or the inhabitants of a `Building`. Since these concepts are relevant for both profiles, the previous classes are defined in both Building and Road Profile-specific Ontologies.

Because the meaning of a particular type of `Feature` depends on the profile that is looking at it and, in turn, it is expressed through the relationships that it establishes with other `Features` it coexist with, each Profile-specific Ontology defines a particular set of relationships the profile is interested in. The layered design of the ontologies allowed the Road profile to define the relationship called `roadServesTo` and its inverse `isServedByRoad` which state what `Road` serves any other `Feature` and vice versa. On the other hand, the Building profile defines the relationship `holds` and its inverse `isLocatedAt` which establishes which `Feature(s)` a given `Parcel` holds and, inversely, where a particular `Feature` is located.

A set of SWRL rules has been defined for each profile so that the corresponding Validator agent can detect the errors and conflicts that appear in the model and that are related to its field of expertise. Regarding the errors, for example, each `Road` must specify its capacity in order to check if it can hold the potential traffic. If a `Road` is missing this attribute, then the model is not complete and the validator agent infers an `Error`. Equation 12.1 shows the rule used to infer a `RoadCapacityError`, a specific type of `Error` defined in the Profile-specific Ontology for the Road profile[2]. This rule could be read as: if an element *r* happens to be a `Road`, and the result of the operation *isMissingAttribute* for this road and the attribute name "capacity" resolves to true, then *r* is also a `RoadCapacityError`. The operation *isMissingAttribute* is an example of how it is possible to extend the general logic operations of ontologies with user-defined operations. This mechanism is allowed in SWRL rules by means of the use of Built-ins. Similar rules were used by the Building profile to detect when a building does not declare the amount of inhabitants.

$$\begin{gathered} Road(?r)\wedge \\ isMissingAttribute(?r, \text{``}capacity\text{``}) \rightarrow RoadCapacityError(?r) \end{gathered} \tag{12.1}$$

[2] For the sake of readability, prefixes of the atoms have been removed from the rule. Though, it is worth mentioning that all atoms have a namespace referring to the ontology in which they are defined, so that potential ambiguities are resolved.

On the other hand, the two profiles involved in this use case may also come into conflict. That is the case when the building designer places a building in a parcel where the road connecting to that parcel cannot hold the new population brought by the building. Equation 12.2 shows the rule defined in the Building profile to detect this kind of `RoadExhaustedConflict`. This rule is actually a compound rule that: retrieves the `Parcel` *p* where the `Building` *b* is placed, gets the `Road` *r* serving that parcel and computes whether the road is overloaded with the buildings that are connected to it through the operation *isRoadExhausted*. This rule leans on the inference done by another rule about which road serves a parcel (see equation 12.3). This latter rule explores the relationships of the ontology to get the `Geometry` *gp* of the `Parcel` *p* and selects the closest `Road` *r* by means of the operation *closestRoad*.

$$\begin{aligned} &isLocatedAt(?b,?p)\wedge \\ &isServedByRoad(?p,?r)\wedge \rightarrow RoadExhaustedConflict(?r,?b) \\ &\qquad Road(?r)\wedge isRoadExhausted(?r) \end{aligned} \tag{12.2}$$

$$\begin{aligned} &Parcel(?p)\wedge \\ &closestRoad(?pg,?r)\wedge \\ &hasGeometry(?p,?pg) \rightarrow isServedByRoad(?p,?r) \end{aligned} \tag{12.3}$$

12.4.2 Conflict Solving

The Conflict going to be solved is detected by the Validator agent (see figure 12.1) who provokes the initiation of the conflict solving protocol described above and depicted in 12.3. When the Coordinator is notified, he broadcasts the Call For Proposals to the Negotiators acting as proxies of the engineers. The engineers at the clients receive a message informing that a new Conflict solving sequence has been started and the Coordinator is waiting for their proposals. Knowing the details of the conflict (i.e. the Road is exhausted) they think on how to fix it. Different solutions like, for instance, enlarging the Road for more capacity; or reducing the amount of inhabitants of one or several Buildings; or maybe relocating a Building in another Parcel served by a Road with more availability for new residents; are then applied temporarily to the model. A recording system allows capturing the changes to the model. The engineers encapsulate sequences of changes (such as “on Building number 32, set the value of the Attribute ’inhabitants’ to 30 from 40”) into a Solution and provide all the alternative Solutions they have. All the alternatives are then proposed to the Coordinator who evaluates them and discards repeated or invalid ones. The viable Solutions are then sent back to the clients so the engineers express their preferences on each of them by grading each with a value ranging from 0 to 10. The grades are then sent back to the Coordinator who takes the winner solution as described above. The winner solution is then notified to all the clients and applied to the model.

12.5 Conclusions and Future Work

In this paper we presented a system designed to support collaborative work in Civil Infrastructure projects that is able to assist in the detection and solving of semantic Errors and Conflicts. These semantic problems, which also involve geometric problems, are so common that they are normally accepted so long they can be in-field detected and corrected. However, this is not always the case and they may eventually lead to project delays and to overheads. Thus, it is important that the models are delivered free of problems as much as possible. A semantically perfect model without problems or ambiguities eases the automation of the tasks, which translates to a more efficient usage of resources.

Conflicts are a special case of problem which are especially difficult to solve. Negotiating is the natural mechanism to reach an agreement on how to solve them. Our system provides a structure for this negotiation by means of suggesting alternatives and picking the preferred one among all the parties -the Profiles- involved in the conflict. The alternatives to the conflict are currently provided by the parties.

However, a learning mechanism is desirable to register the solutions applied to a certain type of Conflict. The aim is twofold: 1) as the system learns and gains in competence, it suggests solutions by itself and 2) to study whether these suggestions can be comparable in terms of quality to those issued by human experts.

Acknowledgements This work was supported by the 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 the 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.

12.6 Amendments

As mentioned in the conclusions section, this chapter describes a system to detect and treat design errors. This is a journal article. Due to restrictions on the formatting imposed by the journal in terms of length, some concepts were not fully covered to the satisfactory extent. This section is meant to extend them without changing the contents of the article that has already been published. This section is not in any case part of the original published article.

12.6.1 The relationship between MARA and Bargain Theory

The essential difference between Bargaining Theory and MARA stems from the fact that they emerged in different fields. Bargaining Theory originates from Economics and Game Theory. In contrast, MARA is a genuine Computer Science and Artificial Intelligence approach to the same problem. They are two theories that overlap to a rather big extent, but they specialize in different worlds. Bargaining Theory has a strong focus in the analysis of the set of solutions and how to find a good solution in much more abstract terms than MARA. That is why concepts like, e.g., *Pareto efficiency* are very recurrent in the literature Bargaining Theory. When it comes to MARA, it focuses in the theory of decisions in MAS. In this sense, MARA is a functional model of the Bargaining problem. That is, it makes the negotiation problem treatable by means of a more concrete approach and thus allows to implement a program more easily. As an example, in Bargaining Theory there is no special interests in the protocols followed by the agents to conduct the negotiation but just to study how will they decide. Moreover, in Bargaining Theory the decision of what will the solution be is implicitly made by the problem itself according to the axioms. For the case of MARA, the protocols are essential since without them it is very difficult to have a good coordination of agents in a MAS which, for instance, might require to define which of the agents decides which solution is the chosen one. In short, they could be seen like the abstract and the concrete descriptions of the same problem.

12.6.2 Differences between normative ContractNet protocol and the negotiation protocol presented

It is worth stressing that the protocol used is not a normative Contract-Net Protocol (CNP) but rather a CNP-inspired or CNP-like protocol. The protocol presented here mimics the sequence of the exchanged messages of CNP but without considering the semantics of the CNP. The philosophy behind a normative CNP is that agents are antagonistic and do not cooperate while in this protocol agents, i.e. engineers, have a common goal. However, in terms of the sequence of exchanged messages, both protocols are similar [3]. There is a first call for proposal (CFP) message sent by an agent that coordinates all the other agents. The negotiator agents then send their proposals (i.e. the alternatives). The next step is that the coordinator agent requests other agents to order the alternatives by giving them a preference value. Finally, according to the selection criterion (Utilitiarian in this case, but it could be Nash, or others presented in sections 11.3 and 11.2) the coordinator agent decides which alternative will be used as the solution for the conflict.

[3] Hence the use of the term ContractNet-*like* to describe the protocol (see section 12.3.2).

Chapter 13
Adaptive Software for Situated Projects

The concept of situated nature of Civil Infrastructure projects has been introduced in previous chapters. A motivation explaining why a paradigm shift in the way software tools are designed has also been exposed. In turn, a solution implementing the relevant theories has been proposed. What has been achieved so far is already relevant because current solutions are unable to hold for long each time a new [design conflict] situation arrives. As a consequence, in a traditional environment, engineers end up negotiating a solution for every design conflict. The system proposed here is capable to assist in this process. However, from a pragmatic point of view the system as it is now can arguably be considered impractical. Launching a negotiation for every conflict detected can become very tedious and, in order to be a real alternative solution, it should facilitate things, not to add an extra burden. Nevertheless, from a conceptual point of view, its ability to turn the wicked problem [Fitzpatrick, 1998] of the conflict solving into a tamed and methodologically-approachable problem is very powerful because an abstraction of the wicked problem is finally achieved. This abstraction is illustrated in figure 13.1. The negotiation becomes a self-contained piece of program which takes (1) the conflict in itself; (2) the model being designed where the conflict occurs; (3) the engineers' knowledge (by means of the alternatives provided); (4) each engineer's preference on each of the proposed alternatives to the solution; as the input of the program, and it returns a solution to the conflict which is consensuated by every party in the negotiation as the program's output.

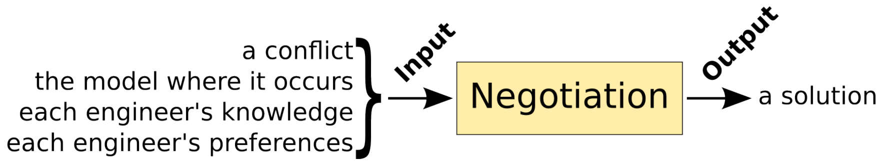

a conflict
the model where it occurs
each engineer's knowledge
each engineer's preferences
Input
Negotiation
Output
a solution

Fig. 13.1: Negotiation from a methodological point of view

What happens inside the Negotiation box is the sum of the project's distributed cognition (see Chapter 9). This distributed cognition could be captured because it was possible to create a system that allowed the interoperability between the behaviors distributed throughout each and every member of the project. The question now is: is it possible to exploit the system's ability to capture the project's distributed cognition and gain from it? The answer is: in theory, yes. And the key is to equip the agents with the capacity of learning and adaptation. If the agents can discover behavioural patterns on how conflicts are solved, then it is plausible to think of a situation in which agents achieve enough level of competence such that, in front of a conflict they can guess how the project as a whole, i.e. as the sum of all the intentions, would solve it. Consequently, by predicting the result of a conflict negotiation, the negotiation itself is no longer necessary. As a result, the system would automatically self-adapt to the project's vicissitudes and be able to issue proposals on how the conflict could be solved[1].

[Skolicki and Arciszewski, 2003] define an *intelligent agent* (IA) as an agent that has, in addition to the standard Wooldridge properties, a substantial subset of the following features :

- Monitoring, perceiving - the agent should sense and observe the environment within which it is located [Leitão and Restivo, 2000].
- Acting, operational control - the agent should perform chosen actions which may change the environment or agent's state either autonomous or in a reactive manner [Russell and Norvig, 2009].
- Knowledge, ontology - the agent needs to have understanding of the environment to perform its task [Nwana et al., 1999].
- Learning, adaptability - the agent should enhance its behavior during its life [Tecuci, 1998].
- Continuity - for some authors, the agent is required to be able to run a process continuously [Franklin and Graesser, 1997].
- Mobility - for some authors, the agent is required to be able to change its location [Jennings and Wooldridge, 1998].

[1] As it will be described, in the system being proposed in next chapter, all agents create their own knowledge base and get trained independently. Another option could be a central knowledge base to train all agents at the same time. The study of whether a distributed or a centralized knowledge base is best is left as future work.

Technically speaking, so far the Negotiatior and Validator agents have all these properties except for their lack of Learning, adaptability and Mobiliity. Notice however, that Mobility is out of the scope of this thesis because it is not necessary for them to move from one location to another within the system. Mobility is anyhow guaranteed by the technology used [Bellifemine et al., 2007]. So, according to this classification, if the agents in the system manage to learn, they will be Intelligent Agents.

13.1 Towards Intelligent Agents with Machine Learning

In order to turn an agent into an IA it needs to learn. Machine Learning (ML) [Russell and Norvig, 2009] is what will let it do it. In ML, the Artificial Intelligence (AI) met the scientific method. In the recent past, AI started an evolution from the pure conceptual and ad hoc systems to a more empirical and statistical analysis methodology. Inspired by the scientific method, there is a trend since the last two decades that empirical methods are substituting or being used to validate the methods that originated the AI such as expert systems, first order logic, etc. [Cohen, 1995].

ML, like the scientific method, is essentially about discovering how a given setting will behave by means of making experiments about it. While the goal of the scientific method is to define or discover a model (typically a mathematical model) to explain a system, the goal of ML is not only discover it but also to exploit it in order to solve problems that are typically too complex to be addressed programmatically.

For the sake of readability, a very short introduction on how ML works is given. This thesis is not a treatise of Modern Artificial Intelligence since literature about it is very extensive. Good introductions are [Russell and Norvig, 2009] and [Witten et al., 2011]. Take this as a disclaimer if the introduction given here is oversimplified as the only purpose is to provide the reader with the necessary background to understand the concept as it can be applied.

So, ML is about discovering patterns on a set of samples called training set. Depending on which technique is used the pattern is found in one form or another. They slightly diverge depending on whether the data of the samples is, for instance, composed of discrete or continuous values, texts or other criteria. But the idea is that a function is not defined analytically but by means of the samples in the training set as it is exemplified in table 13.1. The function definition is unknown a priori.

$f(x_1, x_2, ..., x_n)$		12.1
$f(x'_1, x'_2, ..., x'_n)$		24.4
$f(x''_1, x''_2, ..., x''_n)$	$(= ??? + ??? \times ???/??? =)$	2.5
$f(x'''_1, x'''_2, ..., x'''_n)$		-57.9
...		...

Table 13.1: An example of training set used to learn a function

ML relies on an accurate training set. If the training set has no subjacent structure it might be because it is small sized, i. e. too few relevant samples, or because it has a lot of noise. Of course, a too small or too noisy training set affects to the quality of the learning.

If the function to be learned is simple, then the training set can unequivocally define it. Take as an example the case shown in table 13.2 which defines the OR[2] logical function. This function takes two parameters which are discrete and can be either 0 or 1. The function returns 1 if any of the parameters is 1, and 0 otherwise. An adequate ML algorithm (a decision tree in this case) will perform perfectly on this training set[3].

Unfortunately, there is no real science to ensure a training set is good until it is tested against a problem and a ML algorithm. Fortunately, the algorithms tend to have a similar architecture so that it is

[2] The OR, literally the English word "or", logical function is one of the logical operations that are commonly used in programming to evaluate the truth of a given situation. They operate with logical, also called Boolean, operands which can take two values: *false* or *true* often encoded as a 0 and a 1. Other standard logical functions are AND, NOT, XOR (eXclusive OR).

[3] As it will be shown in in Chapter 14, the decisions in civil infrastructure projects are within a relatively small set of possibilities (e.g., "move object X here", "move object Y there", "remove object Z", etc.). For that, the training set might not need to be big to give acceptable results. In fact, even though there is no guarantee about, it is reasonable to expect that training sets generated in this system will be sufficient. **The real challenge seems to reside in the management of the set of attributes instead because it is variable and unknown a priori**.

x_1	x_2	$f(x_1,x_2)$
0	0	0
0	1	1
1	0	1
1	1	1

Table 13.2: Complete training set for the logical OR operation

relatively easy to find software packages that allow relatively simple usage and exchange between algorithms once they are integrated[4].

Conceptually, the training set is for ML what the algorithm is for programming. It describes how the program behaves. There are some advantages that make it especially appropriate, though. The program using ML (or the agent in this case) might eventually become competent enough to solve a given problem if it is able to learn. i.e., if it can capture enough amount of relevant samples and analyze them properly. If an existing training set that has been performing well eventually turns out to be erroneous, obsolete or noisy so that it starts to produce unwanted results, it can be discarded causing the 'reset' of the agent's knowledge. The training set can also be manipulated to remove bad samples so that its quality improves[5]. Thanks to all these possibilities, the agent can automatically improve after it was created and released. In contrast, algorithmic problem solvers have only one chance to be correct that is when they are written. After that they remain as they were. It is easy to see that, in the problem of solving conflicts in Civil Infrastructure design, ML emerges as a potentially better tool than algorithms.

In the next chapters, it is described how ML can be applied to complete the system presented so far. The result is a system that can capture the knowledge of a collaborative project, the distributed cognition. The project's distributed cognition that allows complex works such as those in Civil Infrastructure design to produce results that would not be possible for single a man to produce. The system described is able to capture and organize all this knowledge in an im-

[4] For the experiments of this thesis, the Weka 3 framework [Hall et al., 2009]) for data mining was used. Weka 3 can be very easily used as a library linked to a system. It is easy to integrate and provides with a wide range of already built-in ML tools for free as well as mechanisms to extend it with others so long they follow the same API.

[5] The refinement of the training set is not covered in this work; it is considered future work.

plicitly, elegant and simple way; a knowledge, otherwise wicked and blurry, that becomes a real resource that can be -and is- exploited for obtaining better design tools. [6].

[6] Notice, though, that some of the problems that have to be solved are how to deal with the fact that the conflict detection is based in first-logic order while the learning algorithms used are based on attribute-value pairs.

Chapter 14
[Article] Distributed Cognition Learning in Collaborative Civil Engineering Projects

Jaume Domínguez Faus[1], Francisco Grimaldo[2]

[This article] Appears in: Management Intelligent Systems issue of *Advances in Intelligent and Computing*, 171, Springer-Verlag. 2012.

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

Jaume Domínguez Faus
Centre for 3D GeoInformation, Aalborg University. 9220 Aalborg, Denmark +45 9940 3699
e-mail: jaume@land.aau.dk
Francisco Grimaldo, Fernando Barber
Departament d'Informàtica, Universitat de València, Av. Vicent Andrés Estellés s/n, (Burjassot) València, Spain 46100
e-mail: francisco.grimaldo@uv.es

14.1 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 designing 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 [Eastman et al., 2008]. 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 [Kolbe et al., 2005], 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 [Eastman et al., 2008] 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 difficulties in tackling the challenges derived from distributed collaborative work but literature suggests that Multi-agent Systems (MAS) can be 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 [Ren and Anumba, 2004]. 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 [Schnellenbach and Denk, 2002]; [Udeaja and Tah, 2001] 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, 1998] 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 on construction projects where these maps are used to parameterize the agent's beliefs within the MAS negotiation [Xue et al., 2010]. 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 cognition* 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 14.2 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 [Faus and Grimaldo, 2012]. However, they are briefly introduced in sections 14.2.1 and 14.2.2 for the sake of readability. The Learning Subsystem is described in section 14.2.3. Finally, in section 14.3 we describe a use case in which the system was applied in order to illustrate its usefulness.

14.2 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 14.1). 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 [Bao et al., 2009] ontologies and SWRL [O'Connor et al., 2005] 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 agent ecosystem uses JADE [Bellifemine et al., 2007] 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.

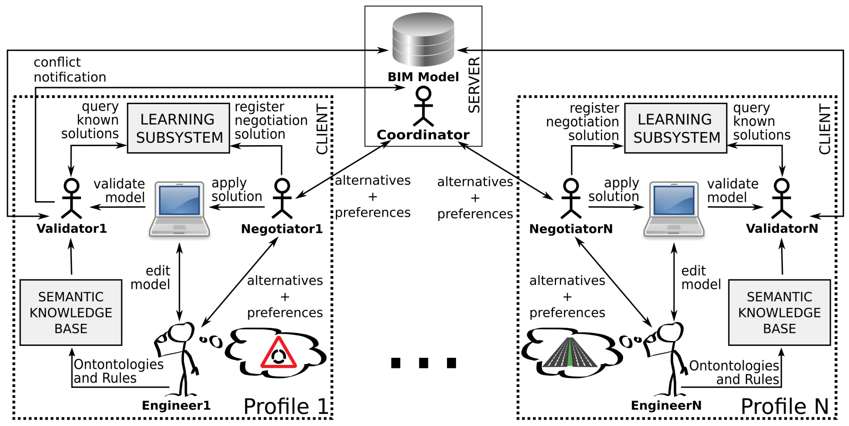


Fig. 14.1: Overview of the system

14.2.1 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 monitored by the Validator agents, which execute 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 analyze the model from each Profile's perspective in order to find `Conflicts`. More details about the semantic conflict detection can be found at [Faus and Grimaldo, 2012].

14.2.2 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 [Chevaleyre et al., 2006] 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 consist 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 [Faus and Grimaldo, 2012].

14.2.3 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 [Nash, 1950], 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 is 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 14.2.2 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.

Reich [Reich, 1997] carried out an extensive review where he analyzed experiments with ML in Civil Infrastructure concluding that

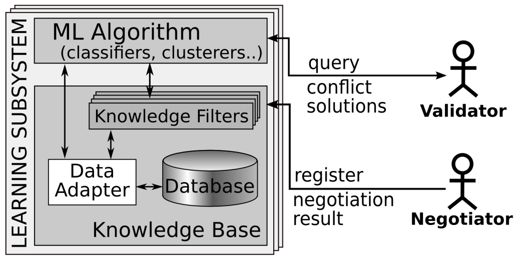


Fig. 14.2: Learning subsystem

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. [Witten et al., 2011]) 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 four 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, whenever a 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 14.2) 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.

14.3 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 [Jayanti et al., 2006], 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:

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

Upon execution, this rule would tag any feature denoted by *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 downwards (Y axis), i.e. it needs to move to the distance $d1 = (-0.03, -0.01, 0)$ (alternative $A1$). The Structure profile suggests 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 (#@0#, in this case) so they can be more generally applied. Table 14.1 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.

CTF	CWFType	CWFRF1	...	CWFRF1AttrLeg	...	Solution
Bolt	Hole	StructurePart		Right		MoveFeatureCmd(#@0#,-0.03,-0.01,0.0)
Bolt	Hole	StructurePart		Left		MoveFeatureCmd(#@0#,0.03,0.01,0.0)

Table 14.1: Simplified database for this conflict. Each row represents a conflict and its negotiated solution (CTF=ConflictedFeatureType, CWF=ConflictedWithFeature, CWFRF=CWF-RelatedFeature, “#@0#” is a symbol refering to the first feature that is in conflict that is substituted and restored by the Knowledge Filter)

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 14.2.3.

14.4 Discussion

As computers became more powerful, they allowed 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 basically 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 know-ledge-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 beginning 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.

14.5 Amendments

This is a journal article. Due to restrictions on the formatting in terms of length imposed by the journal, some concepts were not fully covered to the satisfactory extent. This section is meant to extend them without changing the contents of the article that has already been published. This section is not in any case part of the original published article.

Perhaps, what demands more explanation is the table 14.1 and how it is obtained by the LS. The strict rules in formatting forced to compress it so much that the information was degraded in excess. It is possible to find an extended version of the table in Appendix C on page 237. The following description assumes that the reader has a basic knowledge in object oriented programming and data structures. Let's now see how the LS was done.

14.5.1 The commands

Recalling that a solution for a conflict is an operation that when applied to the model it solves such conflict. How such operation can be implemented depends on the design of the application. This means that depending on how it is implemented, a learning subsystem like the one described above will be easier to achieve or not. But there is a technique that comes very handy for this system which was not described before because 1) it falls a bit out of the main topics of this thesis (i.e., Negotiation, MAS, and ML); and especially, 2) because it is just the way how it was solved by this research project but it does not mean that it is the "best" way to solve it. It consists of using the well-known *Command* object oriented design pattern[1].

The Command pattern consists of a very simple interface that allows wrapping small pieces of code into an object that the application can manage. A very useful application of commands is to add support to the classic *do* and *undo* support to programs. That is achieved by having an interface for the commands which defines an *execute* and an *undo* operations to perform the necessary actions for the command to be executed and to be undone. Let's call this interface *ICommand* following a standard naming guide line which puts an I in front of the interface name. Let's now imagine that we want to have a command that moves some feature a given distance in 3D. The only necessary things for having a fully operative command are:

[1] In fact, this technique turned to be so generic and flexible that it became the *seed* to the edition support of the Vianova Systems' new generation of products.

- Create a class named, say, *MoveFeatureCmd* implementing the ICommand interface. This forces the class to have a definition of the operations defined by the ICommand interface (i.e. *execute* and *undo*[2]).
- Add a constructor[3] to the class which takes and stores the parameters inside the command object. As said, the parameters are 1) feature being moved (or, better said, an ID so it is easily reachable without needing to carry it along), and 2) the distance vector.
- Implement the *execute* operation as follows: iterate through all the coordinates of the feature's geometry and *add* to their x, y and z components the value of the x, y and z components of the distance vector.
- Similarly, implement the *undo* operation as follows: iterate through all the coordinates of the feature's geometry but now *subtracting* to their x, y and z components the value of the x, y and z components of the distance vector.

With this simple class, an application can move a feature in the space by just creating a MoveFeatureCmd object with the right parameters and call its *execute* method. Not only that. It can also undo that move and bring the feature back to the original location by just calling the command's undo method. If the application uses a stack to manage the commands then it will achieve a rather nice do and undo support for every kind of operation it can do as long as all those operations are wrapped into an ICommand the same way we did for the move operation.

As mentioned above, supporting do and undo is not the goal of the application. They are just used to show that the command pattern can be exploited to wrap pieces of executable code and manage them in an homogeneous way. What it is wanted is that the systems learns what to do in front of a given situation. In other words, what operation to do, or, simply, which command to execute. That makes necessary that a command can be encoded and decoded in some way so it can be saved or created. This process is sometimes referred as serializing/deserializing and sometimes as marshalling/unmarshalling. In both cases it

[2] The *undo* operation is not strictly necessary in our context, but it helps keeping the flow of the discourse because it is a rather standard feature that almost all application have.

[3] In object-oriented programming, a constructor is a special operation that creates a new instance of an object.

refers to the process of 1) taking an object in memory like, for instance our commands, and transform it to a sequence of bytes so that it can be stored in a file or database, and 2) read those bytes from a storage and recreate the object they define. Similarly to the do/undo operations defined by the ICommand interface, it could define another couple of operations for serializing/deserializing so that each type of command knows how to represent (i.e., serialize) and also restore (i.e. deserialize) itself. If a command is able to do it this capability can be exploited to perform the learning. That is possible because the LS learns from a database; then, thanks to the serializing, we can store a command in the database; and finally, thanks to the deserialize, we can recover it when needed.

An object can be stored (serialized) in many different ways such as in binary code, XML, or others. Which one to choose is just a matter of meeting the needs. Because WEKA, the ML framework used in this thesis, uses text files[4], it was decided that the commands would be serialized as text strings consisting of the name of the command class (i.e. MoveFeatureCmd in this example) followed with the list of parameters enclosed in brackets as shown in table 14.2. It has to be noticed, though, that this is an arbitrary decision made for the sake of simplicity and, in any case, this does not mean that other notations are not possible or even more appropriate in production environments[5].

Command	Feature ID parameter	Distance Moved (3D vector)	Serialized Command
MoveFeatureCmd	42	(-0.03,-0.01,0.0)	MoveFeatureCmd(42, V3D(-0.03,-0.01,0.0))

Table 14.2: Serialization of a MoveFeatureCmd command. The term V3D is a short for Vector3d. Its only purpose is to ease the deserialization process of the command but it is not strictly necessary.

[4] WEKA can work with more powerful database systems without problems but the demands for this thesis were perfectly covered by the plain text files.

[5] The study of better notations is left as a possible future work.

14.5.2 The context

Being the solution for a conflict, the Commands become what the learning algorithm needs to respond in front of an input. In a table representation of the data set like the one in table 14.1, it is normally placed in the last column of the table. Often, this column is referred as the *class* attribute. Once trained, the learning algorithm will find the closest match among the samples and then return its class attribute value; i.e., the command to be executed in this case. Then, what is missing is the rest of the attributes, i.e. the rest of the columns of the table.

Recall that what describes a conflict is the conflict itself but also the context in which it occurs (see section 14.2.3). This is supported by the fact that Civil Infrastructure design is a collaborative work which implies that it is situated and, as [Fitzpatrick, 1998] explainsthe context in which problem occurs is the most important part of its definition. Considering this, it is natural to think that the context is what shapes the rest of the columns of the table.

How the context is expressed depends pretty much upon the way the model is defined. In this sense, it seems not possible to find two models that define an object in the same way. For instance, a house in CityGML 1.0 is composed of walls, roofs, and maybe some furniture. In turn, IFC is much more detailed since it defines types of pipelines, electricity and so on. So, the same object has many potential representations. The situation repeats if the internal model is a newly created one in which a house could, for instance, be defined by the inhabitants, and the address without including any geometry at all. This infinite myriad of possiblilities might seem discouraging. But actually, the situation is much better if we abstract our point of view. As it was described in chapters 6 and 10 in an incremental manner, there is a clear tendency for the models to be composed of features which are in turn composed of attributes and possibly other features as well by establishing relationships among them. Supported by this fact, a level of generality that can be detected to allow implementing the learning. With this conviction in mind, the context can be, and is, defined by means of the conflicted feature's attributes and also the attributes of the features that are related to the conflicted ones (i.e., those for which there is a relationship that connects them to the conflicted ones).

Since the conflict is defined by the conflicted features and the context and it has been said that the context is defined by the relationships woven among the features, it is reasonable to use the set of features involved and its attributes as a mechanism to capture the context. Of course, this implies that the richer the model is, the better description of the context will be obtained. The better context, the more likely the learning algorithm will detect the correct behavior pattern of a conflict.

For the use case at hand[6], the objects of the model are Structure features that connect to Bolts (see Figure 14.3).

In this model, it was decided that the `Structure` feature was composed of `StructureParts` which have an attribute called *leg* defining which leg it is (right or left). This composition is established with a relationship. A `Structure` also has `Sockets` attached to it and `Sockets` have `SocketHoles`.

Finally, the database shown in Appendix C is built by placing the features and their attributes involved in columns as follows:

For each feature involved in the conflict and its related features..

- Add a column stating the type of feature. The name of the column will be "Type" plus the index of the feature within the conflict's list of features
- For each attribute of the feature, add a column with the value taken in each sample. The name of the column will be the name of the attribute plus the index of the feature within the conflict's list of features that it belongs to.

For the use case at hand, the features involved are: on the one hand, the `Bolt` and the `SocketHole` where the bolt does not fit as the directly conflicted features; and, on the other hand, the Socket where the socket hole is, the other `SocketHoles` in the same socket, the `StructureParts` of the same structure and the `Structure` they all belong to[7]. Thus, it is possible to build a database of samples on

[6] A video showing the use case is publicly accessible on the Internet at following the Youtube address https://www.youtube.com/watch?v=xyXApszXrHo

[7] Note that the `Structure` feature is not shown in the database of the Appendix C due to lack of horizontal space. It is not a problem though since the behavior of the algorithm does not change.

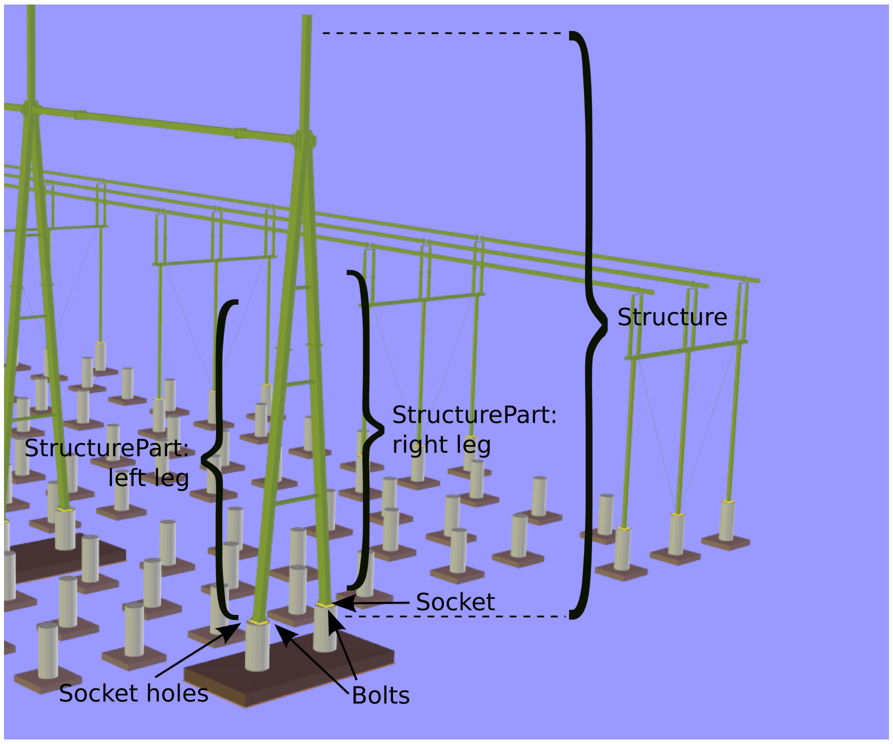


Fig. 14.3: Features involved in the conflict.

how a given conflict is solved. Then, it is only about to let the learning algorithm work by training it with the samples. The algorithm itself will find what is the relevant information that causes a good decision. In this use case, it turned out to be the attribute that defines which leg (left or right) is involved in the conflict.

14.5.3 A note on the #@0# symbolizer

The section 14.3 describes and justifies the need for using symbolizers as a way to be able to apply a solution to a similar conflict caused by other features. It seems necessary to stress that this is needed due to how the commands operate which take a feature as a parameter. The use of the sequence "#@0#" as a symbol is only due to implementation details[8]. It was just considered sufficient to reduce the risk of clashing with another portion of the command's serialized string. It should be read as a pointer to the first feature of the list of features involved in the conflict. If the solution would be to move the second feature of the list of features then the symbol used would have been "#@1#" and "#@2#" for the third and so on. As discussed in section 14.2.3 these symbols are just string substitutions performed by the KnowledgeFilters which replaces the feature ID by the symbol before introducing samples to the training set; and the symbol by the feature ID after obtaining a solution from the ML algorithm. It can be seen as a way to reduce the dimensionality of the set of possible values in the class attribute.

[8] This author does not claim that it is a good one and suggestions in this regard are welcome

Chapter 15
[Article] Infraworld, a Multi-agent Based Framework to Assist in Civil Infrastructure Collaborative Design (Demonstration)

Jaume Domínguez Faus[1], Francisco Grimaldo[2]

[This article] Appears in: Proceedings of the *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Volume 3, International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS). 2012. The article was accompanied by a demonstration video that is publicly accessible at:

http://www.youtube.com/watch?v=xyXApszXrHo

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 engineer's 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.

Jaume Domínguez Faus
Centre for 3D GeoInformation, Aalborg University. 9220 Aalborg, Denmark +45 9940 3699
e-mail: jaume@land.aau.dk
Francisco Grimaldo
Departament d'Informàtica, Universitat de València, Av. Vicent Andrés Estellés s/n, (Burjassot) València, Spain 46100
e-mail: francisco.grimaldo@uv.es

Key words: Distributed Artificial Intelligence, Multi-agent Systems, Distributed decision making, Semantic conflict detection

15.1 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 [Eastman et al., 2008] 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.

Despite previous works like [Peña-Mora and Wang, 1998] the process of alignment is still a prominent manual one, unfortunately. 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 [Eastman et al., 2008]. 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 [Bao et al., 2009] 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.

15.2 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.

15.2.1 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, sewerage conduction might only be an obstacle for an engineer that is planning 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:

$$\begin{aligned} &Conduction(?c1) \wedge Sewerage(?c2) \wedge isBelow(?c1, ?c2) \\ &\quad \rightarrow PositionNotAllowedConflict(?c1, ?c2) \end{aligned}$$

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

15.2.2 Collaboration Module

The collaboration module defines a Multiagent society (see Figure 15.1) 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 sends 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[1] 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 de-

[1] Scoring from [-5, 5] makes more intuitive for users to express utility. The system internally translates the values to the range [0, 10] since they need to be positive. This is not a problem because of the Translation Invariant property (see Chapter 11)

cision to the Negotiators. This solution is finally applied to all the models in the distributed environment.

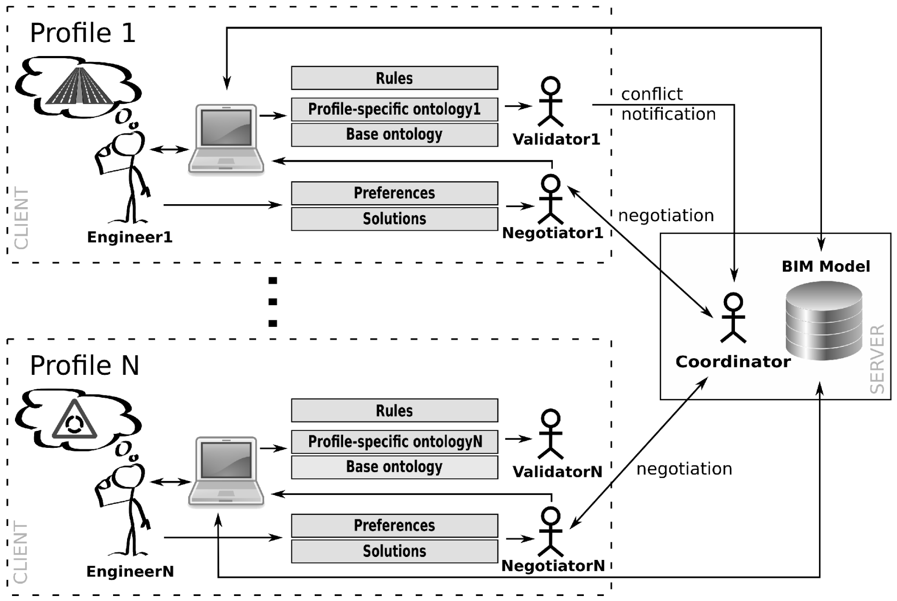


Fig. 15.1: Overview of the system

15.2.3 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.

15.3 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.

15.4 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

Chapter 16
Overview of Other MAS Systems and related works

An overview of some previous works that show similarities with this thesis is given to conclude Part II. They have been already mentioned in the State-of-the-art sections of the published articles (see chapters 12 and 14). These works are briefly discussed in this chapter and compared with Infraworld. For in-depth details, please refer to the references given. Despite what it might be expected, there are not many MAS-based systems dealing with Civil Infrastructure Design. As it has been already mentioned, there are systems dealing with other problems of the Civil Infrastructure domain like supply chain management, machinery (in the sense of robotics), or public tendering process. However, by the time this thesis was written and to the author's knowledge, the number of published works that approximate the system exposed in this Part II is very sparse. This shows how new the field of MAS systems is new to the Civil Infrastructure Design domain. Moreover, these works fail to consider one attribute of the Civil Infrastructure project that is key in this thesis: the situated nature of Civil Infrastructure projects. To the author's opinion not considering the situated nature precludes the applicability of these works beyond the use case they were applied to. Here is an overview of such previous works:

16.1 Peña-Mora and Wang [Peña-Mora and Wang, 1998]

Peña-Mora and Wang's is probably the first attempt to bring the multiagent theory to Civil Infrastructure from a Game Theory point of view. In essence, their pioneering work consists on the proposition of the Game Theory as a tool to solve design conflicts in the same way it was already applied in other fields like Economics or Political Science.

Peña-Mora and Wang present their work by means of laying out a hypothetical conflicting between an Architect, an Engineer and a

Constructor (AEC conflict) that need to decide what to do in front of an unexpected situation. The situation emerges when an old industrial installation is discovered just after the Constructor starts digging to construct a section of a major express way.

In a system called CONVINCER, which is Game Theory-based, the authors refer to the agents as in Nash's definition of agent. Thus, formal MAS (as Wooldridge's definition) standard techniques like speech acts are formally covered. These authors focused on the negotiation which is based on utility functions. Thus, their methodology suffers from the problem of utility functions (see 11.4). To be based on utility functions has been proven to be inoperative as utility functions are easy to understand but difficult to define. The main contribution is the consideration that agents[1] in a negotiation can enjoy different negotiation power[2].

16.2 MASCOT [Ren et al., 2003]

The main focus of MASCOT, "a high-level multi-agent system architecture for construction claims negotiation", is the negotiation and the distribution of risk the participants of a negotiation take. MASCOT's main goal is to improve the negotiation mechanism itself without paying attention to the result. MASCOT authors have the belief that the key is to give good support to the negotiation.

Compared to Infraworld, MASCOT lacks the capacity of evaluating alternatives. The support for the negotiation is based on pure Game Theory. For this reason, MASCOT also suffers the problem of the utility functions described in section 11.4. They assume that engineers have a formula that considers their tastes, their previous knowledge, and many other variables for which units of measure barely exist or not at all. In the cases where defining a utility function is possible it is when it refers to consumable resources like e.g. money or time. But as described in this thesis, the concept of utility is wider than that.

[1] "agents" as in Nash's definition

[2] Strictly speaking, Infraworld does not consider asymmetric bargaining power currently. But, as shown in 11.5 (page 144) it is absolutely possible to add support for it. It basically consists of including the α parameter for each agent involved in the negotiation.

16.3 ADLIB [Ugwu et al., 2005]

The name ADLIB stands for Agent[3] based support for the collaborative Design of Light Industrial Buildings. ADLIB, though having some similarities to Infraworld, has some differences. The first difference comes with its purpose: the design of light buildings. It is the only work of the presented that considers a design conflict within the Design phase only. I.e., the conflict is treated before it involves the constructor. The second one is that the human engineers are not involved in the negotiation. The negotiation is conducted by the agents automatically.

Focusing on the building design, ADLIB has a restricted, fixed and predefined Knowledge Base. In this sense, ADLIB does not have the same level of generality than Infraworld. To give an example, ADLIB has a limited set of choices whenever it is being decided how a building has to be. For ADLIB, a building can only be "big", "medium", or "small" sized. It can be argued that it is conceptually valid, but to the author's opinion, such a simplification does not recognize the situated nature of the projects causing that it covers very specific negotiations.

ADLIB does not have a spatial model. While Infraworld is geared to 3D environments, ADLIB limits its decision making support to attributes.

Another difference is that ADLIB limits the profiles of the system to 1) Architectural, 2) Structural, 3) Costs and 4) Constructability / Safety. In contrast, Infraworld supports an arbitrary number of profiles.

In the technical aspect, ADLIB does not use normative FIPA speech acts standard in the multiagent systems world, but the involved agents communicate by means of standard WWW services like email, FTP, etc. Agents in Infraworld are implemented in JADE, so the communication is based on FIPA messages.

The negotiation in ADLIB is triggered on-demand when the user clicks on the "Design" user interface. Then Agents in ADLIB negotiate a design by choosing from a set prefabricated pieces. Unlike Infraworld, once the negotiation is finished, the result is not used to feedback the system so that agents learn from it. From ADLIB's point of

[3] "Agent" as in Wooldridge's definition

view this makes sense since the agents know the preferences of each role (they are (pre)defined at the same level of the problem). That is only possible if the amount of choices is assumed to be closed and small. In a situated nature project, this assumption is not valid. Hence, agents in Infraworld do not know preferences out-of-the-box unless they get trained.

Conclusions and Future Work

This thesis is a research work that deals with interoperability in the Design phase of Civil Infrastructure process. The interoperability problem has been approached in a way that is not normally done in this kind of researches. The nature of the Industrial-PhD programme of the Norges Forskningrådet (the Research Council of Norway) allowed that the research could enjoy of a level of immersion higher than "standard" PhD researches. This allowed to apply ethnographic methods to gain an in-depth insight of how the daily work of civil engineers and their projects is. The aim of the research was to identify interoperability problems that ultimately cause loss of resources in a highly resource-consuming field. In this research, two types of interoperability problems where identified:

1. Problems related to the exchange of data in which data format play a crucial role; the problems of **Interoperability at a Data Exchange Level** role, and
2. Problems related to the exchange of knowledge in a field characterized by the heterogeneity of expertises working together but in a distributed manner; the problems of **Interoperability at a Distributed Behavior Level**.

1 On the Interoperability at Data Exchange Level

It has been shown that there are many efforts to design software tools and data formats that make the process of data importing and exporting easy and unequivocal. Despite these efforts, engineers still experiencing data exchange issues because it is very difficult to implement software packages that perform the exchange process with no loss of information at any of its intermediate steps.

To overcome this issue, a new technology called Geographic Managed Object introduced by Dr. Jan Kolar has been proposed as a means to avoid loss of information because, thanks to the use of a Virtual Machine, there is no need to use an intermediate data format to transfer the information. GMOs are regular (Java) objects. As a consequence, the can carry not only data, but also behavior. Moreover, the transmission of those objects exploits the concept of serialization. The serialization is a standard process in Virtual Machines that provides guaranteed 100% of information transmission for free. Thus, the problems of data exchange are literally avoided so long there is a VM available in every software tool that will make use of the GMOs and a relatively simple and small module for loading them is implemented on those tools. In comparison with traditional data format parsers, the module required to load the GMOs is much easier to implement. And the same model is able to handle any kind of data format embedded in the objects. Because the objects are Java code (even though the same concept can be implemented with other types of VM such as, for instance, .Net), they allow the highest levels of flexibility when it comes to support data formats.

Currently, GMOs have been used for demonstrating their applicability as data transmitters. They have been also used to demonstrate the transmission of pre-programmed behaviors. Among the behaviors implemented there are sensing the environment (e.g. Taxis implemented as a GMO that it is able to sense its position real-time in the real world, building energy consumption), or simple design conflict resolution between colliding objects (e.g. light poles that are unacceptably close or overlapping a manhole).

So far, however, GMOs have paid attention to the spatial dimension mainly. The temporal dimension is still a research in progress. Wan Wen, from the Aalborg Universitet is carrying this research in order to complete the spatio-temporal domain of GMOs. GMOs are also called to be a viable technology to distribute Virtual Reality (VR) models. However, aspects like appearance and animation have a limited support yet compared to what it is expected on those models. Normally, other 3D technologies available have native support for that almost from the beginning. In this sense, what makes GMOs powerful, i.e. the VM, shows its downside regarding animation and appearance aspects. The higher level of abstraction that VM bring comes at the price that more memory is needed to represent the same information. While this did not seem to be a problem for the experiments carried out in this thesis, it implies a big challenge making realistic VR because of the large amounts of information required to represent graphical animations, appearances, textures, and so on. Therefore, it is proposed as a future work. In the field of Civil Infrastructure, GMOs can also be used to simulate large objects composed of pieces like pipe networks, power lines, or structures by implementing each piece as a GMO. Another topic in which more research is necessary is about how to make GMOs interact with other tightly connected GMOs so that they all behave as a complex structure. In case any of these batons are taken by a new researcher, this author encourages contacting Dr. Jan Kolar at his web page (http://www.grifinor.net), or Dr. Erik Kjems at the Department of Development and Planning of the Aalborg University in order to align efforts.

2 On the Interoperability at Distributed Behavior Level

This work also shows that interoperability issues exist among the people that work in the design of a Civil Infrastructure project. Due to their multi-disciplinary and complex nature, these projects demand that the design work is carried out by teams of experts. Because these teams work in parallel and distributed, necessarily design conflicts appear. It does not seem that this is going to change in the future simply because that is an inherent feature of distributed systems. Currently, engineers solve these conflicts by meeting and negotiate possible alternative solutions.

On the other hand, Civil Infrastructure projects are necessarily situated. They depend not only on the people collaborate to carry on the project, which of course has a considerable influence on how the project evolves, but also on the simple fact that each project is executed in different places of the world with their corresponding particular conditions and vicissitudes. It has not been a satisfactory answer to this topic from the Computer Aided Engineering (CAE) software making industry. That is because the traditional approach of creating software is not well-suited to tackle it. Theories that emerged to cover situated and cooperative works which influenced this work are predominantly academic and experimental worlds that still have long way to evolve. However, these theories, such as CSCW, point out that the problems are more easily tackled if the social aspect of the problem takes a central role when designing the system that solves them.

When the word "social" shows up, for a Computer Scientist's mind it is almost inevitably to hear a bell ringing which points to Multiagent Systems (MAS) as a technology that might fit. This impression becomes stronger if the problems to be solved (the design conflicts) are solved using negotiation. Actually, other properties of MAS (i.e. autonomous, social, proactive, reactive, and adaptive) turn them out in almost the only option possible.

Thus, the MAS system presented in this work is able to improve interoperability in Civil Infrastructure projects at the Distributed Behavior level. It allows to interpret a model (reasoning) focusing on aspects that are important for some discipline's perspective. The analysis of the model from multiple points of view reveals the design conflicts it contains exactly the same way human engineers do. A semantic reasoning can reveal conflicts beyond the pure geometric overlapping ones. On the other hand, thanks to the Bargaining Theory, MARA Theory and similar, the MAS system presented also allows the conflicts to be negotiated and solved while the project's distributed cognition is captured in an easy and elegant way. Due to the situated nature of these projects, that cannot be done with traditional approaches based in algorithms and parametric data. That is because representing all the potential ways to interpret a model would demand so rich data models that they can hardly exist. The ability to capture the project's distributed cognition allows training the agents in the MAS so that the system can learn how to make high-level decisions by discovering the project's behavior patterns. As a consequence, the system can suggest possible solutions to conflicts with similarities to others that have been already solved in the past. This research also showed that the learning capability of the system is not free of challenges. The training set to train the agents does not need to be big. The agents in the system have showed capacity to learn with a relatively small set of samples. But what seems to be more challenging is how to structure this data set. Again, due to the situated nature of Civil Infrastructure projects, it is difficult to keep the set of parameters in the data set (i.e. the structure of the dataset) coherent so that the Machine Learning algorithms can take advantage of it.

It is in order, though, to say that this work has just begun. As it is natural in research many questions remain unanswered due to lack of resources. Here is a list of some of them.

1. During the ethnographic study, participants pointed out that the vast majority of negotiations among engineers occur typically under a *cooperative-collaborative* environment. The system proposed was created accordingly. It does not mean that there are no other possible situations. Often a Civil Infrastructure project is carried out by different companies. Depending on the companies' real interests in the long term, *competitive* or even *strategic influence* behaviors can occur. For instance, trying to cause extra costs to a competitor which could weaken its position in future conflicts or bids for projects. These situations were not covered by this thesis because they fall outside its purpose. Nevertheless, it might be of interest for future studies dealing with long term Civil Infrastructure strategic decisions.
2. One of the things that distinguish this work from others is that while others rely on utility functions to make the decision on how to solve a conflict, this work considers this option too abstract to be feasible in a productive environment. Hence, alternative/preference pairs to define the solution set *S* (see Chapter 11) are used instead (see Chapter 12). It is true that alternatives make the system more feasible. But it is also true that a set of alternatives is a set of points in the Euclidean space while utility functions are continuous curves. Thus, due to the finite set of choices possible in solution sets built up from finite series alternative/preferences, the efficient solution picked will always be one of those points in *S* (see figure 11.1 at page 134), i.e. one of the alternatives provided by the engineers in the negotiation. While it is guaranteed that the most efficient solution among those provided is picked, it is also true that it is not possible to consider other alternatives than those proposed. So long the proposed alternatives are efficient enough, this approach is good. But what if the set of alternatives proposed does not contain the best possible alternative? Unfortunately, the current approach cannot improve them.
   It seems, however, that under certain circumstances, the current negotiation can be upgraded to a pure Nash's Bargaining Problem.

That is, when the set of alternatives belongs to the same vector space, it is actually a set of points that can be converted into a canonical utility function by means of extrapolation, convex-hull, etc.. In such situations, it could be possible that the most efficient solution can be found regardless it is one of the alternatives provided by the actors. In this research, saying that all the alternatives in the set solution set S belong to the same vector space can be read off as all the alternatives are of the same type (e.g., “move the object A to X”, “move the object A to Y”, ..).
In summary, this means that in theory and under these circumstances, the system will perform better than the humans when proposing alternatives to a given conflict. The study of this hypothesis is also a suggested future work.

3. If the hypothesis above resolves to be true, other possible future work could be to study whether is it possible to find a homomorphism that can take all the alternatives into the same vector space (i.e., homogenize the alternatives) so that the previous case can be generalizable to every negotiation regardless the type of the alternatives provided.
4. Currently, the knowledge base created by the agents in the learning process is created by each agent and stored locally. This might imply unnecessary data duplication throughout the system. An interesting alternative would be to study the performance of the system if the knowledge is collected and exploited centrally by, for instance, the Coordinator agent instead (see chapters 14.1 and 15).
5. As already mentioned above, structuring the training data set used to train the agents has shown to be challenging. There is a big amount of research work in the data capture and analysis that could not be completed due to lack of resources. This is something that is not difficult technologically but it is technically complex. The difficult part is to collect the engineer’s data. This work could consist of investigating what information is very relevant and what is not so relevant in order to ensure that the most relevant information is always captured by the system. If necessary the system would discard unnecessary information to keep the data structure of the training set consistent.
6. Currently, the training data consists of operations performed to a system. These operations are a custom and very simple lan-

guage that executes commands. While the current solution was sufficient enough to functionally validate the system, it probably lacks of the level of abstraction used by humans when considering tasks to be done. It would be interesting to investigate how more semantically-rich concepts like, for instance, Ontologies already used in the validation part of the model can be applied to the knowledge base in order to simplify its use and make it more generalizable.

Glossary

Agent (as in Bargaining Theory). Each of the individuals involved in a bargaining. In Bargaining Theory, it is assumed that agents are highly rational, that each agent can accurately compare his desires for various things, that they are equal in bargaining skill, and that each ahs full knowledge of the tastes and preferences of the other [Nash, 1950].

Agent (as in Multiagent Systems and Computer Science). A program that acts autonomously, reactively, and/or proactively and has social capabilities. Additionally, agents can enjoy adaptive and learning capabilities so that their behaviour can change through time as a response of the evolution of the environment in which they exist.

Asymmetric Bargaining Power. In a bargaining, it models situations where there is a difference on the influence a given agent's preference has in the result of a bargaining compared to other agents' influence. In other words, the opinion of one of the agent is more (or less) important than the opinion of on of the other agents in a bargaining problem.

Bargaining Problem. Historically, bargaining has been a mechanism to solve conflicts between two or more parties. The Bargaining Problem is the mathematical model that formalizes bargaining in a methodological way and allows, among other things, to define different types of efficient solutions to a given problem that are guaranteed to be the best, according to the corresponding definition of "*best*", ones (formally known as Efficient Solution).

BOE. Acronym for Boletín Oficial del Estado. Literally "Official State Bulletin", is the gazette where the Spanish government publishes all its contractual decisions, resolutions, laws, public tenders, public job positions offered, etc. Any decision made by the authorities remains unofficial until it is published in the B.O.E.. Once published, it becomes official and legally binding for all parties involved.

CAE. Acronym for Computer Aided Engineering. CAE are software packages that extend the concept of CAD (Computer Aided Design) programs with specific rules of the engineering discipline they are created for. For instance, in a CAE an engineer can not only design the geometry of the component he is designing but also ensure that such geometry meets specific requirements.

CSCW. Acronym for Computer Supported Collaborative Work. It is a field of study that emerges to address the design of software for collaborative (and situated) work.

Distributed Cognition. In a collaborative environment, the term Distributed Cognition refers to the sum of every individual's knowledge, intellectual and/or behavioral contribution which results in the global knowledge of such system as a whole.

DCM. Popularly, the three phases the Civil Infrastructure consists of. DCM stands for Design-Construction-Maintenance. In this thesis, the Data Gathering phase is also considered as phase in its own as it the necessary first task to do and it has its own challenges. In the popular term, it is understood that Data Gathering is included in Design phase.

DGN. Bentley Systems$^{\text{TM}}$ data format used in many large projects including bridges, buildings, roads.

DXF. AutoDesk$^{\text{TM}}$ text-based data format for data exchange.

DWG. AutoDesk$^{\text{TM}}$ data format that is probably the most widely used in the design world due to the success of AutoCAD®.

Efficient solution. One of the choices of a bargaining problem's set of possible solutions (see Chapter 11 on page 131);

ESRI$^{\text{TM}}$ shape file. A data format commonly used in GIS systems.

Existing situation. Technical term referring to the current state of a piece of land where an infrastructure is going to be built.

FTP. Acronym for File Transfer Protocol. Is a protocol that allows sending and receiving computer files that is commonly used as a way to share documents (models) in any computer network.

GIS. Acronym for Geographic Information System. It refer to a software packages that are geared for computer map analysis and visualization.

Holistic approach. In recent times, there is a tendency for approaching problems considering all aspects it concerns beyond the pure technical ones (e.g. ethical, ecological, social, etc.). The term *holistic approach* refers to this practice.

Inheritance. In Computer Science, inheritance means that a data type (class) that is a subtype (subclass) of another type (superclass), inherits the properties of the superclass and provides a specialization of the superclass. E.g. a Bike class and a Car class can be subclasses of a Vehicle class that provide specialized versions of Vehicles. One has two wheels and the other has four and an engine, but they both are Vehicles.

IFC. Acronym for Industry Foundation Classes. It is both a definition of a set of fundamental classes from which 3D engineering models can be built, and also a standard file format to exchange those 3D engineering models.

Java: It an object-oriented compiled programming language. Programs compiled in Java are a bytecode that requires a Java Virtual Machine in order to be executed.

Java3D. It is a software package that allows Java language to run hardware accelerated graphics.

Java Virtual Machine (JVM). It is a runtime that executes on top of an operating system and allows running Java bytecode and abstracts the underlying system so that it is possible to run exactly the same bytecode in different architectures so long a JVM is available.

LiDAR: It is a data format for data captured with a laser scan technique. The data consists of a cloud of 3D points containing the position of the point in the space and, optionally, color values, reflection and possible other kinds of optical or electromagnetic measurements.

Multiagent Systems: Computer environments that allow Agents to exist.

MVC Model View Controller. It is a software architecture that separates a piece of software in three components: 1) the Model that holds the data operated by the program (e.g. coordinates of the nodes of a geometry); 2) the View, that represents this data in a way that it is easy for a user to understand (e.g. a drawing that graphically shows geometries); and 3) Controller that holds all the operations than can be performed to the data model. This separation makes easy to have multiple uses of the same component. For instance, a model can be viewed differently (e.g. graphically, tabulated data) or operated differently (e.g. introducing text commands, or by mouse inputs) by having different Views or Controllers on the same Model respectively.

The Postmodern Inflection. In the mid XX century and aligned with the growing feminism in the western society, Ethnography received a lot of criticism regarding sexist analysis. Ethnography is supposed to provide an agnostic and objective analysis of the subjects under study. However, a lot of previous ethnographic works were accused to focus on a patriarchal, i.e. sexist and therefore subjective, point of view. The Post Modern Inflection is the term that acknowledges that a fully agnostic and objective ethnographic study cannot exist simply because the ethnographer comes from a background that will influence his or her point of view, interests, knowledge and goal. In this thesis, the Postmodern Inflection is identified by the fact that the researcher comes from a Computer Science background. Hence, his interest and capacity to capture information are focused in the computer systems used (or that could be used) in the Civil Engineering process.

Overfitting. When trying to discover a pattern, overfitting refers to the situations in which an extreme, singular, and most probably invalid sample has not been discarded and therefore it is influencing the conclusions of the study to the point that these conclusions can be completely wrong.

Programmer's Dictatorship, the. (see section 9.3)

Schema. In Computer Science, a schema often refers to the set of data definitions that, together with their relationships, compose a complete and consistent data model.

Situated Nature. A property defining that a task, job, or project heavily depends on the contextual environment: e.g. its location, members, weather conditions, suppliers, etc..

System. A complex whole; a set of things working together as a mechanism or interconnecting network.

SWRL. Simple language to define rules that allow reasoning against OWL ontologies.

Utility function. A mathematical function that defines the preferences of an agent within a bargaining.

Utopia. In Bargaining Theory, it refers to the maximum pay-off an agent in a bargaining could get regardless considering other agent's possible pay-off.

Virtual Reality. A computer graphical representation of reality which aims to be as reallistic as possible.

Welfare. The health, happiness, fortunes, etc. of a person or group.

XSD. An acronym for XML Schema Definition language. Language to define data types allowed in an XML document that conforms with such XSD.

XML. An acronym for eXtensible Markup Language. Is a structured text file meta-format that formally defines how to encode data formats in text so that it is easily parsed by client applications.

Troubleshooting Problems

1 When I try to connect to the central AI I always get a connection error saying that there is no JADE engine running.

Surprisingly, even though the distributed computation nature of Multi-agent Systems, JADE (at least at the version 3.7 used in this thesis) still have problems resolving hosts and in some cases even specifying the IP address of the host is not enough. This is a known bug in JADE and it happens very often. A workaround of this is to specify the address in the hosts file of your system (`/etc/hosts` in Linux and `C:\Windows\System32\drivers\etc\hosts` in at least Windows 7 and probably all versions too) by adding the following line:

```
IP_ADDRESS              HOSTNAME
```

Example, if the host name is "Test" and the address is 192.168.1.13, then add the following line:

```
192.168.1.19            Test
```

2 When I try to test it the negotiation sometimes hangs.

I noticed along the years that Java desktop applications sometimes show modal dialogs behind the current window. This is certainly annoying and collides directly with the concept of modal dialog which it is meant for requesting all the input from the user by locking all other windows.

The negotiation process uses modal dialogs for voting. If after every player has provided alternatives you do not see the voting window, please try Alt+Tab -like combination to make the voting window go above the others. This is most likely to happen if an open source JVM is being used (like `icedtea`, `gcj` or `openjdk`).

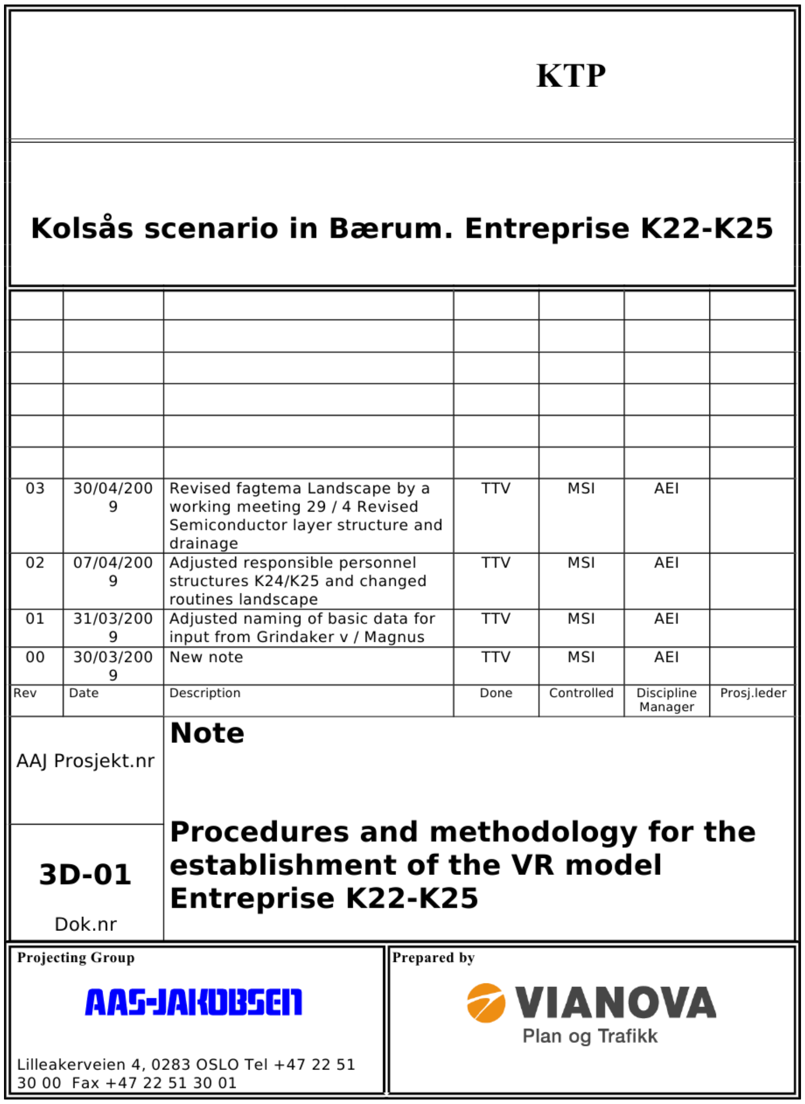
3 Yes, ok. But the negotiation still hangs within Linux.

All the experimenting has been performed in Linux. However, as Java-based, it should be possible to run it anywhere without problems. A problem in the negotiation synchronization sees to appear when using the open source implementation of Java (like `icedtea`, `gcj` or `openjdk`) that comes by default in most of the Linux distributions. Admittedly, The Infraworld experimental framework is not a finished product and unfortunately it is far from being bug-free. But it does not seem to be the case here, or at least the problem is not present with Sun's JRE. Thus, in case you are experiencing something similar, please make sure that ALL the Java Virtual Machines you are using to run the framework (both on all clients and the AI) are Sun/Oracle's which can be freely obtained from the Java official web page which is like `http://java.oracle.com` [accessed on 31/05/12].

4 Known issues

After finishing the negotiation, the view does not get refreshed, so the changes don't seem to be applied (but they are). A workaround is to close and open the view again.

Appendix A. Sample Handbook on Producing Data in Interdisciplinary Design Work (Kolsås Scenario).



KTP

Kolsås scenario in Bærum. Entreprise K22-K25

03	30/04/2009	Revised fagtema Landscape by a working meeting 29 / 4 Revised Semiconductor layer structure and drainage	TTV	MSI	AEI	
02	07/04/2009	Adjusted responsible personnel structures K24/K25 and changed routines landscape	TTV	MSI	AEI	
01	31/03/2009	Adjusted naming of basic data for input from Grindaker v / Magnus	TTV	MSI	AEI	
00	30/03/2009	New note	TTV	MSI	AEI	
Rev	Date	Description	Done	Controlled	Discipline Manager	Prosj.leder

AAJ Prosjekt.nr

Note

Procedures and methodology for the establishment of the VR model Entreprise K22-K25

3D-01

Dok.nr

Projecting Group

AAS-JAKOBSEN

Lilleakerveien 4, 0283 OSLO Tel +47 22 51 30 00 Fax +47 22 51 30 01

Prepared by

VIANOVA
Plan og Trafikk

Content

1 General

In this note describes the procedures and methodology for the establishment and maintenance of the VR model for Kolsås path. The model will be established as a so-called "engineering model" where all relevant disciplines will be incorporated in a common model above and below ground. The data will be incorporated should be in line with the necessary stikningsdata for the facility and based on experience with similar projects carried out and general work around this has been going on in the industry.

The aim of this document is to make good and reasonable procedures and working arrangements so that the model is updated quickly and efficiently and with minimal costs. The works will go in line with the ongoing engineering work and thus be an important tool in quality assurance process.

Furthermore, the document provides a common understanding and commitment of all parties in the project. By accepting this document, the Parties undertake to relate to and work in accordance with this document and any attachments.

2 Objective

The objective of this work is to establish a complete 3-dimensional engineering model. Work on the model will follow the suggested approach in this note and the goal is to create common best practices and interaction between firms so that one provides for an updated model at all times taken an active part in the use of all parties.

The geographical avgrensingen is in accordance with the refinements of the engineering work for all disciplines and is split according to the various contracts.

On the basis of experience with similar projects will <u>some</u> of the benefits of such a common "engineering" model be:

- Utilizes interdisciplinary
- Easy to control projections and existing data from different disciplines (air check) for all parties in the project
- Ensures higher quality data that is sent to the contractor
- Visual representation of what data there is stikningsdata on and what we may lack
- FDV-documentation for all disciplines. That is, one through a common model provides for common documentation on projected stikningsdata and can later take the "as built" data, if desired.

Level of detail and level of ambition for the various disciplines described in more detail under the proposed project design methodology and discussed further with all parties throughout the process.

The model is primarily intended to show a permanent situation for the project, but it will, in consultation with the KTP considered a plus to establish delmodeller for selected areas in the appropriate phases.

3. 3 Time and refresh rate

Work on the model will continue and follow the project, the engineering time. For the model will give us advantage in prosjekteringsfasen must all subject areas provide the basis according to the note as soon as possible. Moreover, the model will be updated and adjusted until the issue of work basis. Later, it will together with KTP considered whether the model should be part of the follow-up during construction and about the content of the model to show the "as built" data.

The model will initially be updated continuously in time with the engineering works. Moreover, the model will be presented and reviewed at fixed intervals, where all changes are presented and displayed. This is clarified further and continuously with the project managers for the various contracts.

In addition, it continuously if it will be a need for working meetings with the responsible of the 3D delivery to discuss any issues and the content of the model.

4 Structure

ViaNova will be responsible for model work and the work will be part of the planning work with the AAJ as contract manager. On the engineering meetings to be held to the status of the model work is a point on the agenda. Need for any clarifications to be addressed in separate work meetings where the minutes of these meetings are attached or referred to in the engineering meetings.

The following people will be central to the establishment and maintenance of the model:

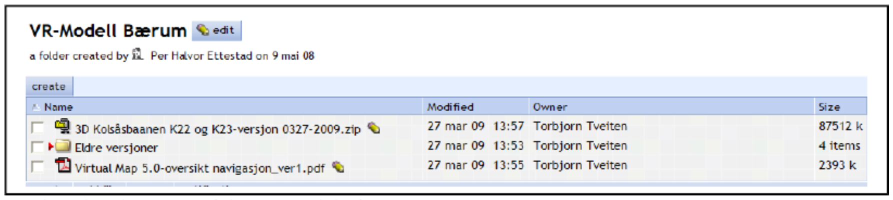
Role	Company	Person
Project manager 3D	ViaNova Plan and Traffic	Torbjørn Tveiten
Model Manager	ViaNova Plan and Traffic	Morten Simonsen
Performing geometry road and path K22-K25	ViaNova Plan and Traffic	Rune Rian
Performing markup and railings K22-K25	ViaNova Plan and Traffic	Rune Rian
Performing structures K22 and K23	Aas-Jakobsen	Jone Stangborli
Performing structures K24 and K25	Aas-Jakobsen	Jørn Ola Sørensen
Performing geotechnical K22-K25	Geovita	Ingunn Veimo
Performing VA and drainage K22	ViaNova Plan and Traffic	Jon Erling Einarsen
Performing VA and drainage K23-K25	Vianova Systems	Rune Johnsrud
Performing electrical K22-K25	Electro Nova	Anders Pedersen
KL-performing plants K22-K25	ECT	Benoni Nera
Performing Locations K22	Grindaker	Magnus Greni
Performing Locations K23-K25	LINK mark	Gisle Totland
Performing Architect K22-K25	Arne Henriksen Architects	Brede Henriksen

5. 5 Data exchange

The model is placed on common eRoom continuously as soon as it is updated. The models are published as a complete file package in Zip format for download and is marked with dates. Older versions of the model is added to the directory.

As a user must check all the eRoom to see that the latest version of the model and delete / overwrite the previous older versions on their local PC / server. It established a group that will automatically notify the eRoom when new model is added above. The group established by VNPT.

Latest edition and updated model is added to eRoom subdirectory "VR model Bærum" as shown below:



VR-Modell Bærum edit

a folder created by Per Halvor Ettestad on 9 mai 08

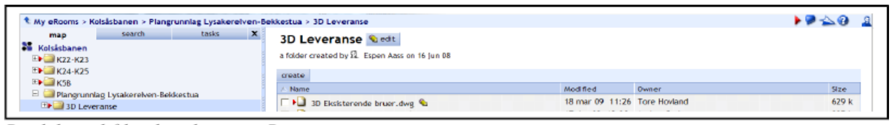
create

Name	Modified	Owner	Size
3D Kolsåsbaanen K22 og K23-versjon 0327-2009.zip	27 mar 09 13:57	Torbjorn Tveiten	87512 k
Eldre versjoner	27 mar 09 13:53	Torbjorn Tveiten	4 items
Virtual Map 5.0-oversikt navigasjon_ver1.pdf	27 mar 09 13:55	Torbjorn Tveiten	2393 k

Rank updated version of the VR-model of eRoom

In addition, it is here posted a user review for navigation in the model and that it will be posted an Excel document that is a log of all changes in the model with a short description. This log is updated and revised by the project manager 3D

All new and revised plan data to enter in the model are added on a separate area of eRoom as shown below:



My eRooms > Kolsåsbanen > Plangrunnlag Lysakerelven-Bekkestua > 3D Leveranse

map search tasks

Kolsåsbanen
- K22-K23
- K24-K25
- K5B
- Plangrunnlag Lysakerelven-Bekkestua
  - 3D Leveranse

3D Leveranse edit

a folder created by Espen Aass on 16 jun 08

create

Name	Modified	Owner	Size
3D Eksisterende bruer.dwg	18 mar 09 11:26	Tore Hovland	629 k

Rank based file plan data on eRoom

File names for the various topics will be according to the description of the engineering methodology in Chapter 7 There is automatically notified via email from eRoom the project owner and the model responsible for the 3D model.

In addition, it posted an Excel document that everyone should fill that describes the delivery and what changes have been made.

6 Software

3D-model is established in the program Novapoint Virtual Map. This is a VR-model so that everyone can move freely look around there a wish. It established a complete model that includes all relevant elements to be built above or below ground level. It added up to active use of the opportunity to turn on and off elements in the shoulder view so the user can see a desire to focus on at any given time.

The models will be established as engineering models and it will not be made any manual processing. That one looks in the model will thus be a mirror image of the projected data at any time in the plan work.

The model is made available to all the updates and all are provided a free viewer. This requires no installation of software, but sets no requirements for the performance of your PC. The model sought generally to be made as "easy" as possible, so that most can run and move seamlessly into the model.

It is also laid out a simple description using the eRoom as described above for the functionality of the view holder. If in addition, the need for training in the use of the model may be directly responsible for the project.

7 Project design methodology

7.1 Existing environment

The model for the existing environment is established on the basis of the corresponding maps and digital terrain model used in engineering work.

The model is supplemented with text for selected place names / street names.
Name of the subscription: *3D_K22-K25_stedsnavn.dwg*

7.2 Projected road and track data / VIPS

All VIPS projects are saved into the TMOD as wire models. Each of the Quadri-bases should include a group of setup where all the line models to be displayed in the 3D model is active.

Based on this established an ini file in the Virtual Map which dedicates different textures to different surfaces in VIPS. If you wish to change the textures, changing the ini file and the models are updated automatically.

It is established as well as a separate VIPS for superstructure up to the ballast which will be incorporated in the model.

7.3 Marking

The model incorporated all the markings of all the projected paths. It established a separate dwg drawing that only contains the markup. It should be made with Nova Point's markup module and established in 2D. It is split in different layers from the type of markup (see details in Annex). In addition, it established a separate layer that contains all the special markup lines that are defined as areas.

Name of the subscription: *3D_K22-K25_oppmerking.dwg*

Proposed layer structure: OPPM_ "type"

7.4 Railing

It established a separate dwg drawing with 2D lines that show the entire rail length. Lines for the longitudinal railings be extracted from the VIPS and separated on different layers for different types of railings.

Name of the subscription: *3D_K22-K25_rekkverk.dwg*

Proposed layer structure: REKKV_ "type"

Stretch The works shown in the images based on a flat, but it can be considered if it can be done as the volume, so that the railing will have a more appropriate thickness.

Railing on the structures are not included in this drawing, but are part of other areas where this is described.

There is not stikningsdata for post position on the feature works but it is proposed that the model is a volume (rekkverksstolperom) from the ground to the bottom bar to detect collisions with other fagtemaer. The same projected data as described above is used.

7.5 Structural

All projected structures modeled and incorporated directly into the 3D model as an external reference (XREF). It has been established common detailed internal procedures for the establishment of the various structures with a focus on data flow and the volume of data and related layer structure. These procedures are distributed and used. This is described, therefore, not further in this document.

All the structures provided separately in relation to its construction number (K?) And part of the contract to which they belong.

Name of the subscription: *3D_K "contract number" _K "design number". Dwg*
Example: *3D_K22_K31.dwg*

Proposed layer structure: "As agreed in the internal procedures"

Adjacent terrain to the structures established as part of the landscape plan, see section 7.5.

7.6 Geotechnical Engineering

This field includes all relevant geotechnical data to be incorporated in the VR model. This can be:

- Mountain Surf and borpunkter
- Sheet piles
- Piles
- Building pits

Mountain Surf and borpunkter:
Mountain surfaces are supplied as triangulating the geometry performed borpunkter. Borpiunkter delivered as well as punk data in 3D.
Name of drawings: *3D_K "entreprise name" _GEO-Fjellflate.dwg*
3D_K "entreprise name" _GEO-Borpunkter.dwg

Sheet piles:
Recent spuntlinjer established that structures and follow internal procedures in accordance with Chapter 7.5. It established close contacts with Aas-Jakobsen so that delivery of this discipline is in line with the procedures established for all structures.
Name of drawings: *3D_K "entreprise name" _GEO-Spunt.dwg*

Piles:
Recent piles established that structures and follow internal procedures in accordance with Chapter 7.5. It established close contacts with Aas-Jakobsen so that delivery of this discipline is in line with the procedures established for all structures.
Name of drawings: *3D_K "entreprise name" _GEO-Peler.dwg*

Build pits:
All relevant building pits are established as separate VIPS projects and saved into the TMOD of thread models, corresponding to the projected road / path. See further details in Chapter 7.2.

7.7 VA and drainage

For all tanks established the objects as solids for outer geometry, similar to what was done for K5B on Bjørnsletta. It is separated on different teams for different type of tanks.

For all even fall lines established the 3D lines the inside bottom wire. For all pressure lines established the 3D lines outside the top wire. It is separated in different layers of different types of wire. The wires are established as solids and "sweeps" with the correct dimension.

Name of drawings: *3D_K "entreprise name" _VA.dwg*
3D_K "entreprise name" _DREN.dwg

It elected not to distinguish between the VA and the drainage system on Team name, so that drainage system is the same material terms as VA. Differs also in the projected and existing in the Quantities.

Proposed layer structure: 3d_XVAcon_prosj _ *"Type" (con = concrete)*
3d_XVAcon_eks _ *"Type"*
3d_XVAplast1_prosj _ *"Type" (plast1 = PVC)*

3d_XVAplast1_eks _ *"Type"*
3d_XVAplast2_prosj _ *"Type" (plast2 = DV)*
3d_XVAplast2_eks _ *"Type"*
3d_XVAplast3_prosj _ *"Type" (plast3 = PE)*
3d_XVAplast3_eks _ *"Type"*
3d_XVAGRP_prosj _ *"Type" (GRP = GRP)*
3d_XVAGRP_eks _ *"Type"*
3d_XVSteel_prosj _ *"Type" (Steel = steel / cast iron)*
3d_XVSteel_eks _ *"Type"*

7.8 Electro

Similarly, the VA and fro drainage established the objects as solids for the outer geometry of the different elements in this field. It separated the teams for different type of materials, so that this can be pre-configured to the VR model.

Name of the subscription: *3D_K "Entreprise name" _EL-"field". dwg (Partition of the field when needed as Cable Channel, trekkekum + +)*

Proposed layer structure: *EL_ "field" _ "type"*

7.9 KL-plant

All KL-masts and associated foundations and power rails established that structures and is referenced in the procedures in Chapter 7.5. It established close contacts with Aas-Jakobsen so that delivery of this discipline is in line with the procedures established for all structures.

Name of the subscription: *3D_K "Entreprise name" _KL-"field". dwg (Partition of the field when needed as KL-master, the power rail, etc.)*

Proposed layer structure: "As agreed in the internal procedures"

7.10 Locations

Primarily established a triangulating surface from established kote plan. This is stored as a block in Autocad and be on the team XXBlock. To ensure a good flow of data between disciplines is established, there is a 3D delineation line link between kote ring, projected road / track and structures. Delineation line is part of the plan-kote delivery and separated on separate layers.

As a basis for the establishment of appraisal line is taken as the basis of completed projects VIPS (surface model and 3D-lines) from the performing road and track geometry. Last update of this foundation is the eRoom in the form of 3D lines in the academic responsibility geometry. For the structures obtained 3D lines from the various modeled structures that are continuously updated on the eRoom. In the transition to the existing terrain established 3D-line, regardless of completed 3D model, and is a premise for the final triangulation. In addition, be submitted to the necessary sections of the complete updated VR model in dwg format for eRoom If desired, after ordering from academic responsibility landscape.

Important individual trees as it should be stikningsdata at which point information is supplied in 2D and separated on different type. Remaining plants that can not be stikningsdata the priority is not in the model, but can be as closed areas such as 2D-lines so that a visual can make it if desired. If these areas will have a volume above ground level called the height of the layer names. Visual expressions of trees / shrubs areas described and or attached photo.

Data on kantstein taken not as part of the delivery of the 3D model in this field, but are produced and delivered from academic responsibility geometry. For material areas that have not encroach on the terrain and to be treated is supplied as a closed 2D line description of the type of material in the layer names.

Simple points where it may be established structures (plant boxes) are separated on separate layer and delivered as 3-D line for the top box. It may be separated on different types and comes with its type drawings so that this pre-defined as objects that are automatically incorporated in the model volume.

Name of the subscription: *3D_K "Construction name "_LANDSKAP.dwg*

Proposed layer structure: *Lark-L000-contours-3D |*
Lark-L181-Border Measures-3D
Lark-L771-Grass Area-2D "material description"
Lark-L772-Busk Kant-"2D/3D" - "material description"
Lark-L772-Hedge-"2D/3D" - "material description"
Lark-L772-Three new-2D "type description"
Lark-L867-H-coating type-2D "material description"

7.11 Architect

All items pertaining to the subject area architect who drive furnishing, technical buildings, railings mm. established that structures and is referenced in the procedures in Chapter 7.5. It established close contacts with Aas-Jakobsen so that delivery of this discipline is in line with the procedures established for all structures.

Name of the subscription: *3D_K "Entreprise name" _ARK-"field". dwg*
(Partition of the field when needed as a technical construction, railings etc.)

Proposed layer structure: "As agreed in the internal procedures"

Appendix B. Interviews.

0.1 Interview # 1: Vicente Chorques

Vicente Chorques is a Civil Engineer at CMD Ingenieros who has been working on real projects since 2006 mostly located in the city of Valencia. Among his projects we can find urban bridge building, street development, and some other urban interventions. Nowadays, he is in charge of managing the development of the city of Batumi in the ex-soviet republic of Georgia.

Batumi has been known to be a vacational place in the soviet hemisphere until the fall of the USSR. It is located at the Black Sea near to the Turkish border. As part of the union it had a planned economy in which lots of industries were established to serve the productive system in the communist era. As the system fell down, the undergoing little investment in infrastructure was canceled and the country entered an economic lethargic phase that lasted more than one decade. Internationally ignored until the South Ossetia war. It was then that the western hemisphere, led by the USA, understood that helping developing the country would be very important to ensure western interests were not threatened by the Russian Federation influence. The consequence is that large amounts of capital have been given to the Georgian government to spend in the construction of infrastructures that will bring back the country into a state of development in which it may counteract the Russian influence.

Interviewer: Could you explain how a civil infrastructure project is normally carried out based on your own experience?

Vicente: Yes, let's do it from the beginning. All the projects we have done are contracted by governments. The way they do it in Spain is as follows: if the administration wants, for instance, to build a road they have to publish it in the B.O.E. (acronym of the Spanish for Official State Bulletin) which is the official channel used by the government to communicate with the people. So, any company interested in participating in such contracts usually has people keeping track on this publication and looking for projects that may benefit the company.

When we, as a civil engineering company, notice this announcement we immediately contact the corresponding department that issues the new project's definition and ask for the details we need to submit our proposal to the public tender.

I: You said details. What do you mean by details?

V: All the details. You know, most of the times, basically the project's definition is not much more than "we would like to build a road from here to there with capacity for some several thousands of vehicles per hour in each sense and without crossing this, this, and this other protected areas" and, of course, we need to know much more than that. For instance, we need to know what kind of terrain we are going to find there. Not only geologically but also how it is treated officially.

I: How it is treated officially?

V: Yes, if an area that falls in the way where the future road should be has been defined as a public green land, it is illegal for this area to hold human constructions. So, the road must somehow go around it.

I: I see, you mean maps of land uses.

V: Exactly. It is possible (it is actually more than just possible) that our company does not have such information or the information we have is too old to be reliable. And not only that, we must have information about the terrain and its composition, so we contact the authorities to get a DTM (digital terrain model) as close and updated as possible. We must have information about the facilities built in the area occupied by the future road. That is: plumbing, drainage, telecommunication lines, power lines, gas pipes, railroads, and so on. Having preliminary information the engineers start working on the project planning. They produce a report that includes a 3D model and the costs of building it and the economical offer. Once all the offers have been submitted, then the tender is resolved and the government picks up the one that is more of its interests.

I: Tell me about the estimation of costs. How do you do to know what the project's cost will be?

V: That is what is called Material Execution Budget and defines the costs of each resource unit and how many units of each resource are required to execute the project. By resource we understand everything: from worker hours to cubic meters of concrete. At the end, the project's cost is the sum of each resource unit multiplied by the amount of units it requires to complete the work.

I: And then you add the profits you want to get for your company from the project.

V: Actually, not. I don't know how it is in other countries (despite I'm now working in Georgia! [laughs]), but in Spain the benefits that you are allowed to add are defined by law and they can be, depending on the type of the construction, up to 16% or 18%. Even more, the costs you estimate for the project are then signed in a contract but it does not mean that you will receive the exact amount of money specified in the contract from the owner. What you get is the costs of the actual resources spent. For example, if the project estimates that 500 meters of pipelines are needed and estimates a cost of 10€/meter, then the signed contract will define that the costs of the pipelining is 500x10 = 5.000€, but if at the end only 450 meters of pipelines are necessary, the constructor will receive 450x10 = 4.500€.

I: Interesting.

V: Yes, it does not work as the normal people would expect. So the designers must be very careful to evaluate the costs of the materials, and the builder has to validate the estimations he receives on the planning from the engineers. An error here may be very costly.

I: I can imagine. Have you ever experienced any situation in which you delivered a wrong project to the builder contractor?

V: Yes, several times. It actually happens quite often.

I: Oh, yes? What kind of mistakes lead to this?

V: Any kind of error you can imagine. From an underestimation of the material costs to drilling in a place where an unexpected water distribution line was and it is causing inconveniences to the users of that line.

I: Who is responsible for the overrun caused by those mistakes?

V: Well, if there is a mistake in the project that wasn't detected in design time and the construction works have been started, you (the builder) are entitled to demand a change in the project or, even more, to invalidate it. Briefly, this means to start over the process from the very beginning to solve this problem. I mean, to publish in the B.O.E.

that this project needs to be fixed. This means that the engineering-company may lose the contract. On the other hand, if the error in the project happened because of incorrect information supplied then the engineer has the right to rectify it without losing the contract. Then there is a commission that studies the case and decides how the overhead is satisfied. In case the information supplier was a private company and if the information was not what it was supposed to be by contract, then it is likely that this company will have to carry on the overhead. Otherwise, the government is the subsidiary responsible.

I: You have highlighted the importance of the information provided to ensure the quality of the project and for the responsibilities on design flaws. But what exactly does this information look like. I mean does this information look like?

V: Plans and reports in any kind of medium. From paper reports and plans to PDF files containing the description and AutoCAD files. It depends on the area and the type of the work. If your work will be located in a small village, then don't expect to get digital data. You will be lucky if there is any available information! They don't have the infrastructure to store all the generated information along the history. They mostly store the documents and papers in their archives and as the amount of material increases it is harder to maintain the order and finally the information gets lost under tons of papers. In contrast, when you work for bigger authorities, they usually have infrastructure to store the information and they usually do it in digital formats that allow them to easily find it. It is expected that in the future all the entities will go to digital storages, but for the moment a lot of information is still in paper.

I: I'm curious. Why do they give you AutoCAD files?

V: It is kind of a standard. All the programs we use can open AutoCAD files perform their computations and then export the results as AutoCAD files again so they can be used in other programs.

I: So, do you use more than one software package?

V: Of course. There is no software that solves all the problems we have to face. For example, to design the road and calculate the earth volume to be moved we use ISPOL ISTRAM and CLIP they also calculate the slopes. But for structure estimation you have to use others like SAP2000.

I: And you need to know how they all work.

V: It is recommended, yes. But the companies usually have experts in each topic that mostly use the applications that are specific to their topic and they concentrate in that topic along projects.

I: Because the project is quite big and needs to be simplified, Divide et vinces.

V: Yes, each group concentrates in specific topics.

I: How do you manage the understanding among teams?

V: Well Usually, there is always a project manager who is in charge of revising the designs and arranging meetings with the groups where the teams seat and discuss what has been going on. The rest of teams listen and give their opinions, offer help, warn about conflicts in two or more designs, and so on.

I: It seems like most of the work is done by hand, don't you think? I mean: collecting information, importing data into different programs, sit and discuss the work on each team and so on. Don't you miss any automatic tool of doing all of this?

V: As I said, each team has to deal with their own battles. Each topic has its own mathematical and physical model and each model has been implemented by different applications. That is why there are several teams and it does not seem that the situation is going to change in a near future. Obviously, each new version of the softwares we use tries to give more functionality and we may reach a point in which we will only need one product. Maybe then the things can be more automatic. And maybe then is when we all lose our jobs! [Laughs]. But I honestly doubt it since the models out there are a lot and so complicated. So, the only way is to have a server where the files are stored and all the people can access them and see what happens.

I: What happens when you work together with other companies? All the companies share a server?

V: Definitely not! Nobody wants to give others the access to the internal network.

I: So then it is completely manual.

V: I would say yes. Only when the company is big and it has offices in several cities they use a software to organize all the data they have.

In other places I have been, the server is just a network drive in which you save your files.

I: Thank you for your time.

V: You are welcome.

0.2 Interview # 2: Raúl Néstor Rodríguez Fajardo

Raúl Néstor Rodríguez Fajardo is a Civil Engineer specialized in Structures. He worked for 10 years until the end of 2011 at In Hoc Signo Vinces, S. L. which is the office the famous architect Santiago Calatrava has in Valencia (Spain). Currently, he is part of the Degree of Freedom company in Oslo and collaborates with the Bridge Department of Multiconsult.

Interviewer: Hello Raúl. Let me ask you, what kind of projects do you work on?

Raúl: I'm currently working on bridge projects. I've also worked on train stations, covering stadiums in the past; projects with a lot of coordination between disciplines. In my company there were architects, structural engineers, industrial engineers, and IT engineers, too.

I: IT engineers?

R: Yes, well. They do not do the project per se, but provide with the computer infrastructure for it. Since we were a company with several offices in Zurich, New York and Valencia, we needed an infrastructure to allow us to communicate with others. It was a job for the IT engineers to build the system we used for our job, that is: videoconferencing, file sending, etc.

I: File sending, how did it work?

R: It looked like a folder in Windows Explorer where you put files. It was not an FTP[1], for example, but the subfolders were organized in a way that it was easy for the engineer to share plans. So, for instance, you could be working on a structure and when the design was done, you would store the file in the corresponding folder and then send an email to whomever it was intended for, telling him that "I left the design for such in folder such"... or the other way around if I was the intended recipient.

I: What kind of files did you use?

R: DWG, GSDB, PDF, Doc, and XLS.

I: Text Documents? For reporting?

R: Yes, when I made a design, I would also write a document describing it. You know, specifications and things like that.

[1] File Transfer Protocol

I: It looks like you didn't have a big variety of software tools, isn't it? I see here AutoCAD, SAP2000, and Microsoft Office.

R: Yes, well, those were the basics and there could be some other programs for calculus like a Swiss one called FAGUS. But with the first ones, probably 95% of the work was covered.

I: Could you describe an example of communication between different parties in a project?

R: Yes, if the constructor had our plans but during the construction he or she realized that what we delivered could not be done because the plan designs collided with the existing situation, the constructor proposes alternatives that we evaluate and accept.

I: How were these alternative proposals sent?

R: That would depend on how complex the issue was. If it was just a simple thing, a written explanation of the change in an e-mail could be enough. If the situation was difficult to explain, the constructor could send us an email with a drawing, even a hand-made one. Or if it was really difficult, we could even get the plans we delivered rectified.

I: What kind of issues could you encounter?

R: For instance, when we were building the Ciutat de les Arts i les Ciències (CAC) of València and the constructor was digging the earth for the foundations we found a surprise... When they started excavating we found an old dump full of sport shoes and other city waste which was not described in the geotechnical study. So, the ground was bad quality to support the structure as we designed it. So we had to dig deeper.

I: These are problems that happen in the Construction phase. But, did you experience "surprises" as you call them in Design phase.

R: Well, yes, in big projects, when you deliver your designs to the customer it is usual that they have their own engineer teams that validate what you delivered. If they don't like it then you have to fix it. It was not really the case for the CAC. The control in that project was very light, but it is quite usual that they double-check your work.

I: Curious, the CAC had an overhead of... how much? 300%?

R: I don't know. Im in design, it is not my duty to negotiate projects or prices. Anyway, if the project changes on-the-go as it happened in the CAC, inevitably the cost goes up.

I: Ok, so it was more about factors that were external to your company. Did you had “internal” errors or even delivered projects with errors?

R: It is true that there is no perfect project, but not in basic things. I think it could happen in secondary things like lack of specs of a secondary beam or something like that.

I: A problem in the beam does not sound like a secondary thing a problem in the beam.

R: These kinds of things can be solved in the construction phase.

I: But do you validate the plans?

R: Yes, there is a person who validates the plans.

I: Who is this person?

R: Each project has a hierarchical structure and the person in charge of this project validates all the designs.

I: One person validates the whole project, or it is done by more than one?

R: If the project is small, there is no need. But if it is big it is validated per [spatial] areas.

I: It seems different than in Norway. Here the extension is not the way to split the project but by disciplines.

R: Oh, I understand you now. Yes, we do that here as well. What I was trying to explain you had to do with the Structure, my field, but there is actually a higher level distribution of the work in disciplines as you say, of course. We have Landscaping, Architects, Structure, Installation... yes, it is normal. Inside this discipline there are groups of work.

I: How do you coordinate the teams?

R: Look, for instance, if the Installation team needs that some pipe needs to cross the structure, they must tell the structural engineer (me). I, then, would either be told or receive an email saying so. I can consider making a hole in the structure and reinforce it around or, if it is not possible, respond saying that it is not possible.

I: Even if you have been notified, the notification is quite informal, isn’t it?

R: Well, if you mean that there is a procedure or a protocol for it, I think there is none. At least I'm not aware of it.

I: But how can you be sure that you are delivering correctly? I mean, you exchange files, you validate and you coordinate it manually. Isn't that like quite error-prone?

R: You know, this is human relationships. And it normally works. I mean, if the Installation guy wants to put cables from one side to another, we talk it and then we put it in the plans.

I: At some point, do you see everything, all designs put together or have separated plans all the way?

R: There are more and more programs that try to integrate everything. But.... most of the times it is not necessary, I think. At least, from my point of view, I don't need to know all about all the other disciplines. A structural engineer would be interested on the architecture and installation drawings, but not for example of the, say, traffic signs. The only thing a structural engineer needs to know is the weight he has to hold. Everything else is not really important for him.

I: How, when facing an issue, do you proceed to solve it? How would you describe it?

R: Well, if you receive a superior's command about how to fix something, you just apply it. Sometimes it is a bit like a market where you offer and other accept and vice-versa.

I: A bargaining..

R: Exactly, a bargaining.

I: Do you experience problems... well, I suppose you don't since you use basically 2 programs, but... did you ever experienced that when sharing files you have problems importing and/or exporting?

R: You know, even using only two software packages, yes, it happens. Even when working with different versions of AutoCAD the file might experience problems. If two engineers are using two different versions, then problems are likely to occur.

I: Big issues?

R: Well, as long as we detect them, it is just a matter of asking the author of the problematic file to recreate it with the correct version. No big issue, I suppose, because things that are big are easily detected.

I: Thank you very much Raúl.

R: Thank you.

Appendix C. Sample database used for the learning.

```
@relation BoltDoesNotFitConflict
@attribute Type0 {Bolt3}
@attribute Type1 {SocketHole}
@attribute Type2 {Socket1}
@attribute Type3 {StructurePart}
@attribute Leg3 {Right}
@attribute Type4 {SocketHole}
@attribute Type5 {SocketHole}
@attribute Type6 {SocketHole}
@attribute Type7 {SocketHole}
@attribute Type8 {SocketHole}
@attribute Type9 {SocketHole}
@attribute Type10 {SocketHole}
@attribute class {MoveFeatureAndAttachedCmd(#@0##c#V3D(0#p#03#c#0#p#01#c#0#p#0));MoveFeatureAndAttachedCmd(#@0##c#V3D(-0#p#03#c#-0#p#01#c#0#p#0));}

@data
Bolt3,SocketHole,Socket1,StructurePart,Right,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(0#p#03#c#0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Left,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(-0#p#03#c#-0#p#01#c#0#p#0))
Bolt3,SocketHole,Socket1,StructurePart,Right,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(0#p#03#c#0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Right,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(0#p#03#c#0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Left,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(-0#p#03#c#-0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Left,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(-0#p#03#c#-0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Right,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(0#p#03#c#0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Right,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(0#p#03#c#0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Left,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(-0#p#03#c#-0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Right,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(0#p#03#c#0#p#01#c#0#p#0));
Bolt3,SocketHole,Socket1,StructurePart,Left,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,SocketHole,MoveFeatureAndAttachedCmd(#@0##c#V3D(-0#p#03#c#-0#p#01#c#0#p#0));
```

Raw WEKA database

Type(	Type1	Type2	Type3	Leg3	Type4	Type5	Type6	Type7	Type8	Type9	Type10	class
Bolt3	SocketHole	Socket1	StructurePart	Right	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(0.03,0.01,0.0));
Bolt3	SocketHole	Socket1	StructurePart	Left	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(-0.03,-0.01,0.0)
Bolt3	SocketHole	Socket1	StructurePart	Right	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(0.03,0.01,0.0));
Bolt3	SocketHole	Socket1	StructurePart	Right	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(0.03,0.01,0.0));
Bolt3	SocketHole	Socket1	StructurePart	Left	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(-0.03,-0.01,0.0)
Bolt3	SocketHole	Socket1	StructurePart	Left	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(-0.03,-0.01,0.0)
Bolt3	SocketHole	Socket1	StructurePart	Right	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(0.03,0.01,0.0));
Bolt3	SocketHole	Socket1	StructurePart	Right	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(0.03,0.01,0.0));
Bolt3	SocketHole	Socket1	StructurePart	Left	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(-0.03,-0.01,0.0)
Bolt3	SocketHole	Socket1	StructurePart	Right	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(0.03,0.01,0.0));
Bolt3	SocketHole	Socket1	StructurePart	Left	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	SocketHole	MoveFeatureAndAttachedCmd(#@0#,V3D(-0.03,-0.01,0.0)

Tabulated and human-readable version of the WEKA database

Bibliography

References

[Adachi et al., 2003] Adachi, Y., Forester, J., Hyvarinen, J., Karstila, K., Liebich, T., and Wix, J. (2003). Industry Foundation Classes IFC2x Edition 2, International Alliance for Interoperability.

[Anumba et al., 2002] Anumba, C., Ugwu, O., Newnham, L., and Thorpe, A. (2002). Collaborative design of structures using intelligent agents. *Automation in Construction*, 11:89–103.

[Arrow, 1950] Arrow, K. (1950). A difficulty in the concept of social welfare. *The Journal of Political Economy*, 58(4):328–346.

[Bao et al., 2009] Bao, J., Calvanese, D., et al. (2009). *OWL 2 Web Ontology Language Document Overview*. W3C.

[Basanow et al., 2008] Basanow, J., Neis, P., Neubauer, S., Schilling, A., and Zipf, A. (2008). Towards 3D spatial data infrastructures (3D-SDI) based on open standards—experiences, results and future issues. *Advances in 3D Geoinformation Systems*, pages 65–86.

[Bellifemine et al., 2007] Bellifemine, F., Caire, G., and Greenwood, D. (2007). *Developing Multi-agent Systems with JADE*. John Wiley & Sons Ltd.

[Bentley, 2012] Bentley (2012). http://www.bentley.com/en-US/Products/MicroStation/.

[Bergenty and Ricci, 2002] Bergenty, F. and Ricci, A. (2002). Three approaches to the coordination of multi-agent systems. pages 367–372.

[Bernard, 1994] Bernard, H. (1994). Research methods in anthropology. qualitative and quantitative approaches.

[Berry, 1964] Berry, B. (1964). Approaches to regional analysis: a synthesis. *Annals of the Association of American Geographers*, 54(1):2–11.

[Bishr, 1998] Bishr, Y. (1998). Overcoming the semantic and other barriers to GIS interoperability. *International Journal of Geographical Information Science*, 12(4):299–314.

[Blomberg et al., 2002] Blomberg, J., Burrell, M., and Guest, G. (2002). An ethnographic approach to design. In *The human-computer interaction handbook*, pages 964–986. L. Erlbaum Associates Inc.

[Bubenko Jr, 2007] Bubenko Jr, J. (2007). From information algebra to enterprise modelling and ontologies-a historical perspective on modelling for information systems. In *Conceptual Modelling in Information Systems Engineering*, pages 1–18.

[Burke and Coyner, 2003] Burke, E. and Coyner, B. (2003). *Java extreme programming cookbook*. O'Reilly Media, Incorporated.

[Butler, 2006] Butler, D. (2006). Virtual globes: The web-wide world. *Nature*, pages 776–778.

[C.A.C, 2012] C.A.C (2012). Ciutat de les Arts i les Ciències de València. http://www.cac.es.

[Chevaleyre et al., 2006] Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., et al. (2006). Issues in Multiagent Resource Allocation. *Informatica*, 30:3–31.

[Chun, 1988] Chun, Y. (1988). The equal-loss principle for bargaining problems. *Economics Letters*, 26(2):103–106.

[Clients, 1997] Clients, F. C. (1997). 60% of clients dissatisfied. *Building*, 8.

[Cohen, 1995] Cohen, P. (1995). *Empirical methods for artificial intelligence*, volume 139. MIT press Cambridge, MA.

[Cook and Churcher, 2003] Cook, C. and Churcher, N. (2003). An extensible framework for collaborative software engineering. In *Software Engineering Conference, 2003. Tenth Asia-Pacific*, pages 290–299. IEEE.

[Crabtree et al., 2005] Crabtree, A., Rodden, T., and Benford, S. (2005). Moving with the times: IT research and the boundaries of CSCW. *Computer Supported Cooperative Work (CSCW)*, 14(3):217–251.

[Dahm, 1998] Dahm, M. (1998). *Byte Code Engineering with the Java Class API*. Freie Univ., Fachbereich Mathematik und Informatik.

[DelPico, 2004] DelPico, W. (2004). *Estimating Building Costs: For the Residental & Light Commerical Contractor*, volume 7. RSMeans.

[Dix et al., 2008] Dix, M., Riley, P., and Autodesk, A. (2008). Discovering AutoCAD 2009 (Autodesk Design Institute Press).

[Dubra, 2001] Dubra, J. (2001). An asymmetric Kalai–Smorodinsky solution. *Economics Letters*, 73(2):131–136.

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

[El-Mekawy et al., 2012] El-Mekawy, M., Östman, A., and Hijazi, I. (2012). An Evaluation of IFC-CityGML Unidirectional Conversion. *International Journal*, 3.

[Erickson and Kellogg, 2000] Erickson, T. and Kellogg, W. (2000). Social translucence: an approach to designing systems that support social processes. *ACM transactions on computer-human interaction (TOCHI)*, 7(1):59–83.

[Faus and Grimaldo, 2012] Faus, J. D. and Grimaldo, F. (2012). Multiagent System for Detecting and Solving Design-time Conflicts in Civil Infrastructure. *Advances in Intelligent and Soft Computing*, (157).

[Ferbert, 1999] Ferbert, J. (1999). *Multi-Agent Systems: An introduction to Distributed Artificial Intelligence.* Addison-Wesley Publishing Company.

[Fisher and Unwin, 2005] Fisher, P. and Unwin, D. (2005). Re-presenting geographical information systems. *Re-presenting GIS*, page 1.

[Fitzpatrick, 1998] Fitzpatrick, G. A. (1998). *The Locales Framework: Understanding and Designing for Cooperative Work*. PhD thesis, University of Queensland.

[FMI/CMAA, 2005] FMI/CMAA (2005). FMI/CMAA Annual Survey of Owners (FMI/CMAA websites). Technical report.

[FMI/CMAA, 2006] FMI/CMAA (2006). FMI/CMAA Annual Survey of Owners (FMI/CMAA websites). Technical report.

[Fox and Gruninger, 1998] Fox, M. and Gruninger, M. (1998). Enterprise modeling. *AI magazine*, 19(3):109.

[Franklin and Graesser, 1997] Franklin, S. and Graesser, A. (1997). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Intelligent Agents III Agent Theories, Architectures, and Languages*, pages 21–35.

[Goclenius, 1964] Goclenius, R. (1964). *Lexicon philosophicum, quo tanquam clave philosophiae fores aperiuntur (etc.).* Vidua Matthiae Beckeri.

[Goodchild, 2001] Goodchild, M. (2001). A geographer looks at spatial information theory. *Spatial Information Theory*, pages 1–13.

[Goodchild et al., 2007] Goodchild, M., Yuan, M., and Cova, T. (2007). Towards a general theory of geographic representation in GIS. *International Journal of Geographical Information Science*, 21(3):239–260.

[Green, 2001] Green, E. (2001). Can qualitative research produce reliable quantitative findings? *Field Methods*, 13(1):3–19.

[Greenberg, 2001] Greenberg, S. (2001). Context as a dynamic construct. *Human–Computer Interaction*, 16(2-4):257–268.

[Greif, 1988] Greif, I. (1988). Overview. *Computer-Supported Cooperative Work: A Book of Readings. San Mateo, Calif.: Morgan Kaufmann Publishers*, pages 5–12.

[Grimaldo, 2008] Grimaldo, F. (2008). *Integració d'habilitats socials en l'animació comportamental d'actors sintètics*. PhD thesis, Universitat de València.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18.

[Halverson, 2002] Halverson, C. (2002). Activity theory and distributed cognition: Or what does CSCW need to DO with theories? *Computer Supported Cooperative Work (CSCW)*, 11(1):243–267.

[Harsanyi and Selten, 1972] Harsanyi, J. and Selten, R. (1972). A generalized Nash solution for two-person bargaining games with incomplete information. *Management Science*, 18(5-Part-2):80–106.

[Harvey et al., 1999] Harvey, F., Kuhn, W., Pundt, H., Bishr, Y., and Riedemann, C. (1999). Semantic interoperability: A central issue for sharing geographic information. *The Annals of Regional Science*, 33(2):213–232.

[Holt and Resi, 2011] Holt, H. and Resi, A. (2011). Bruk av BIM- og 3D modeller i felt. http://www.geoforum.no/kurs-og-konferanser/publiserte-foredrag/2011/stikningskonferansen-1/bim-og-3d/at_download/file.

[Hunter, 2003] Hunter, J. (2003). Enhancing the semantic interoperability of multimedia through a core ontology. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(1):49–58.

[Hutchins and Lintern, 1995] Hutchins, E. and Lintern, G. (1995). *Cognition in the Wild*, volume 262082314. MIT press Cambridge, MA.

[ISO/TC 211, 2009] ISO/TC 211, A. G. o. O. (2009). http://www.isotc211.org/Outreach/ISO_TC_211_Standards_Guide.pdf.

[Jayanti et al., 2006] Jayanti, S., Kalyanaraman, Y., Iyer, N., and Ramani, K. (2006). Developing an engineering shape benchmark for CAD models. *Computer-Aided Design*, 38(9):939–953.

[Jennings and Wooldridge, 1998] Jennings, N. and Wooldridge, M. (1998). *Agent Technology: Foundations, Applications, and Markets*. Springer Verlag.

[Kalai, 1977] Kalai, E. (1977). Proportional solutions to bargaining situations: interpersonal utility comparisons. *Econometrica: Journal of the Econometric Society*, pages 1623–1630.

[Kalai and Smorodinsky, 1975] Kalai, E. and Smorodinsky, M. (1975). Other solutions to Nash's bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 513–518.

[Kashyap and Sheth, 1997] Kashyap, V. and Sheth, A. (1997). Semantic heterogeneity in global information systems. *Cooperative Information Systems: Current Trends & Directions", Eds: M. Papazoglou and G. Schlageter*, page 1997.

[Kjems et al., 2009] Kjems, E., Bodum, L., and Kolar, J. (2009). Managed objects for infrastructure data. *3D Geo-Information Sciences*, pages 97–107.

[Kolar, 2007] Kolar, J. (2007). Managed code interoperability in digital earth technology. In *Proceedings of the 5th International Symposium on Digital Earth, University of Berkeley, CA, USA*.

[Kolbe et al., 2005] Kolbe, T. H., Gröger, G., and Plümer, L. (2005). CityGML: Interoperable Access to 3D City Models. *Geo-information for Disaster Management*, pages 883–899.

[Langrana, 1992] Langrana, G. (1992). *Time in geographic information systems*. CRC.

[Laurenzo, 2005] Laurenzo, R. (2005). LEANING ON LEAN SOLUTIONS-The concept that turned Toyota into a powerhouse has now begun transforming aerospace companies, even during industry downturns. *Aerospace America*, 43(6):32–36.

[Le Clerc, 1736] Le Clerc, J. (1736). *Eloge historique de Feu Mr. Jean le Clerc*. J. Wetstein & G. Smith.

[Lehan, 1986] Lehan, T. (1986). The influence of spatial information on data queries. In *Proceedings of the ASPRS-ACSM annual convention*, pages 250–257.

[Leitão and Restivo, 2000] Leitão, P. and Restivo, F. (2000). A framework for distributed manufacturing applications.

[MacEachren et al., 1999] MacEachren, A., Wachowicz, M., Edsall, R., Haug, D., and Masters, R. (1999). Constructing knowledge from multivariate spatiotemporal data: integrating geographical visualization with knowledge discovery in database methods. *International Journal of Geographical Information Science*, 13(4):311–334.

[Massey, 1999] Massey, D. (1999). Space-Time,'Science'and the Relationship between Physical Geography and Human Geography. *Transactions of the Institute of British Geographers*, pages 261–276.

[McGuinness et al., 2004] McGuinness, D., Van Harmelen, F., et al. (2004). OWL web ontology language overview. *W3C recommendation*, 10(2004-03):10.

[Mennis et al., 2000] Mennis, J., Peuquet, D., and Qian, L. (2000). A conceptual framework for incorporating cognitive principles into geographical database representation. *International Journal of Geographical Information Science*, 14(6):501–520.

[Moulin, 1983] Moulin, H. (1983). Le choix social utilitariste. *Ecole Politechnique DP*.

[Mubarak and Means, 2012] Mubarak, S. and Means, R. (2012). *How to Estimate with RSMeans Data*. RSMeans.

[Nash, 1950] Nash, J. F. (1950). The bargaining problem. *Econometrica*, pages 155–162.

[Nwana et al., 1999] Nwana, H., Ndumu, D., et al. (1999). A perspective on software agents research. *The Knowledge Engineering Review*, 14(2):125–142.

[O'Connor et al., 2005] O'Connor, M., Knublauch, H., S.W. Tu, B. G., Dean, M., Grosso, W., and Musen, M. (2005). Supporting Rule System Interoperability on the Semantic Web with SWRL. *4th International Semantic Web Conference (ISWC), Galway, Ireland, Springer Verlag*, pages 974–986.

[OpenDesignAlliance, 2012] OpenDesignAlliance (2012). https://opendesign.com.

[Palmer and Felsing, 2001] Palmer, S. and Felsing, M. (2001). *A practical guide to feature-driven development*. Pearson Education.

[Peña-Mora and Wang, 1998] Peña-Mora, F. and Wang, C.-Y. (1998). Computer-supported Collaborative Negotiation Methodology. *Journal of Computing in Civil Engineering*, pages 64–81.

[Peuquet and Duan, 1995] Peuquet, D. and Duan, N. (1995). An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data. *International journal of geographical information systems*, 9(1):7–24.

[Pipek and Kahler, 2006] Pipek, V. and Kahler, H. (2006). Supporting collaborative tailoring. *End User Development*, pages 315–345.

[Protégé, 2006] Protégé (2006). Protégé. http://protege.stanford.edu. Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. October 2011.

[Rector and Horrocks, 1997] Rector, A. and Horrocks, I. (1997). Experience building a large, reusable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of the Workshop on Ontological Engineering, AAAI Spring Symposium (AAAI'97). AAAI Press*, page 26.

[Reich, 1997] Reich, Y. (1997). Machine Learning Techniques for Civil Engineering Problems. *Microcomputers in Civil Engineering*.

[Ren and Anumba, 2004] Ren, Z. and Anumba, C. (2004). Multi-agent systems in construction–state of the art and prospects. *Automation in Construction*, 13:421–434.

[Ren et al., 2003] Ren, Z., Anumba, C., and Ugwu, O. (2003). Multiagent system for construction claims negotiation. *Journal of computing in civil engineering*, 17(3):180–188.

[Riehle, 1997] Riehle, D. (1997). Composite design patterns. In *ACM SIGPLAN Notices*, volume 32, pages 218–228. ACM.

[Rising and Janoff, 2000] Rising, L. and Janoff, N. (2000). The Scrum software development process for small teams. *Software, IEEE*, 17(4):26–32.

[Rittel and Webber, 1973] Rittel, H. and Webber, M. (1973). Dilemmas in a general theory of planning. *Policy sciences*, 4(2):155–169.

[Roy and Ramanujan, 2001] Roy, J. and Ramanujan, A. (2001). XML schema language: Taking XML to the next level. *IT professional*, 3(2):37–40.

[Russell and Norvig, 2009] Russell, S. and Norvig, P. (2009). Artificial Intelligence: A Modern Approach Author: Stuart Russell, Peter Norvig, Publisher: Prentice Hall Pa.

[Schmidt, 2009] Schmidt, K. (2009). Divided by a common acronym: On the fragmentation of CSCW. *ECSCW 2009*, pages 223–242.

[Schnellenbach and Denk, 2002] Schnellenbach, M. and Denk, H. (2002). 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.

[Scott et al., 2003] Scott, S., Grant, K., and Mandryk, R. (2003). System guidelines for co-located, collaborative work on a tabletop display. In *Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, pages 159–178.

[Segarra and Vilar, 2011a] Segarra, J. G. and Vilar, M. a. (2011a). Weighted Losses.

[Segarra and Vilar, 2011b] Segarra, J. G. and Vilar, M. b. (2011b). Weighted Proportional Losses Solution. Technical report, Department of Economic Theory and Economic History of the University of Granada.

[Silverman, 2011] Silverman, D. (2011). *Interpreting qualitative data*. Sage Publications Limited.

[Skolicki and Arciszewski, 2003] Skolicki, Z. and Arciszewski, T. (2003). Intelligent agents in design. In *Design Engineering Technical Conferences, Chicago, Illinois, USA*. Citeseer.

[Smith and Nair, 2005] Smith, J. and Nair, R. (2005). The architecture of virtual machines. *Computer*, 38(5):32–38.

[SolidWorks, 2012] SolidWorks (2012). http://www.solidworks.com/ .

[Sperberg-McQueen and Thomson, 2000] Sperberg-McQueen, C. and Thomson, H. (2000). XML Schema.

[Spyns et al., 2002] Spyns, P., Meersman, R., and Jarrar, M. (2002). Data modelling versus ontology engineering. *ACM SIGMOD Record*, 31(4):12–17.

[StatensVegvesen, 2010] StatensVegvesen (2010). Én lang tunnel i Oslo; Operatunnelen. http://www.vegvesen.no/Om+Statens+vegvesen/Media/ Siste+nyheter/Vis?key=176482.

[Sycara, 1998] Sycara, K. (1998). Multiagent systems. *AI magazine*, 10(2):79–93.

[Tecuci, 1998] Tecuci, G. (1998). *Building intelligent agents: an apprenticeship multistrategy learning theory, methodology, tool and case studies*. Morgan Kaufmann.

[Thomson, 1994] Thomson, W. (1994). Cooperative models of bargaining. *Handbook of game theory with economic applications*, 2:1237–1284.

[Touran, 2003] Touran, A. (2003). Calculation of contingency in construction projects. *Engineering Management, IEEE Transactions on*, 50(2):135–140.

[Udeaja and Tah, 2001] Udeaja, C. and Tah, J. (2001). Agent-based material supply chain integration in construction. *Perspectives on Innovation in Architecture, Engineering and Construction, CICE, Loughborough University*, pages 377–388.

[Ugwu et al., 2005] Ugwu, O., Anumba, C., and Thorpe, A. (2005). Agent-support for collaborative design. *Agents and multi-agent systems in construction. Oxford: Taylor and Francis*, pages 102–61.

[Usery, 1996] Usery, E. (1996). A feature-based geographic information system model. *Photogrammetric Engineering and Remote Sensing*, 62(7):833–838.

[van den Brink et al., 2012] van den Brink, L., Stoter, J., and Zlatanova, S. (2012). Establishing a national standard for 3D topographic data compliant to CityGML.

[Van Nederveen and Tolman, 1992] Van Nederveen, G. and Tolman, F. (1992). Modelling multiple views on buildings. *Automation in Construction*, 1(3):215–224.

[Weiss, 2000] Weiss, G. (2000). *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press.

[Witten et al., 2011] Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining Practical Machine Learning Tools and Techniques*. Elsevier.

[Wooldridge, 2002] Wooldridge, M. (2002). *An Introduction to Multiagent Systems*. John Wiley & Sons Ltd.

[Worboys, 1994] Worboys, M. (1994). A unified model for spatial and temporal information. *The Computer Journal*, 37(1):26–34.

[Xue et al., 2010] Xue, X., Ji, Y., Li, L., and Shen, Q. (2010). Cognition driven framework for improving collaborative working in construction projects: Negotiation perspective. *Journal of Business Economics and Management*.

[Yu, 1997] Yu, E. (1997). Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235. IEEE.

[Yuan et al., 2004] Yuan, M., Mark, D., Egenhofer, M., and Peuquet, D. (2004). Extensions to geographic representations. *A research agenda for geographic information science*, pages 129–156.

[Zhang et al., 2012] Zhang, K., Li, P., and Ma, Q. (2012). Complementary Study Between VRML and 3 DMAX Modeling. *Jisuanji Xitong Yingyong- Computer Systems and Applications*, 21(2):85–88.

List of Figures

List of Tables