

Battling Transaction Costs: Establishing an e-Exchange System for Coaseian Bargaining

Abstract

The role of Information Communication Technology (ICT) in environmental protection has become a significant topic in the last decade. Most existing ICT systems, however, don't go beyond the traditional role of distributing information to their users. Moreover, most of these software systems don't provide an integrated solution for several connected problems, which makes them hard to use.

In this paper we will explore the use of software in environmental protection in a top-down approach, starting with the basic problems of environmental economics, namely internalizing externalities. We will propose a system that will try to achieve this by providing solutions to significantly decrease the transaction costs of Coaseian bargaining, while also providing a platform for other types of tools.

Our goal in this paper is to give an in-depth analysis of the proposed software. To do that, we have to describe the system in a non-technical, feature-oriented way, and analyze its most important functions. To achieve that, we will use a software design methodology called the Rational Unified Process (RUP). In the early phase of the RUP we have to establish the requirements of the software based on the results of environmental and institutional economics and provide an analysis of stakeholders and risks. The most important part of our work is a detailed cost-benefit analysis of creating and maintaining the software, which includes the effect on the costs of Coaseian bargaining.

We have found that the system proposed in this paper would be large – though not unreasonably so. We have also found that the social benefits of creating such system are very likely to exceed costs in a 10-year period. Our analysis shows no significant risks that could not be easily averted. Based on our results we recommend that this system should be created and adopted by governments, especially on a large scale.

Table of Contents

1. Introduction	4
2. Background.....	7
2.1. The Coase Theorem	7
2.2. Transaction Costs	11
2.3. The Coase Theorem in Practice	15
2.4. The Role of ICT in Environmental Protection	18
2.5. Summary	21
3. Methodology.....	22
3.1. Software Design	22
3.2. Estimating Transaction costs.....	27
3.3. Risk analysis.....	30
4. Software Analysis.....	31
4.1. Overview of the Software	31
4.2. Understanding the Environment – The Domain Model	38
4.3. Requirement Analysis	42
5. Results	59
5.1. Cost-Benefit Analysis	59
5.2. Risk Analysis.....	72
6. Conclusion, Recommendations	74
References.....	77

List of Figures

FIGURE 1: TYPES OF TRANSACTION COSTS	12
FIGURE 2: THE PHASES AND DISCIPLINES OF RUP	24
FIGURE 3: STAKEHOLDER ANALYSIS	36
FIGURE 4: THE STRUCTURE OF THE SYSTEM	38
FIGURE 5: DECISION MAKING STRUCTURES.....	55
FIGURE 6: THE SCORE OF AN ALTERNATIVE WITH SMART	55
FIGURE 7: THE DOTMOCRACY SHEET	57
FIGURE 8: BREAKDOWN OF COSTS - HUNGARY	88
FIGURE 9: BREAKDOWN OF COSTS - EU	89
FIGURE 10: RISK ANALYSIS.....	124

List of Tables

TABLE 1: ENVIRONMENTAL CONFLICTS	7
TABLE 2: TRANSACTION COST DEFINITIONS	11
TABLE 3: TRANSACTION COSTS TYPOLOGIES	20
TABLE 4: STAKEHOLDERS AND THEIR PROBLEMS	35
TABLE 5: STAKEHOLDERS	
TABLE 6: ESTIMATES FOR TOTAL DEVELOPMENT AND MAINTENANCE COSTS	70
TABLE 7: TOTAL 10-YEAR SERVER COSTS.....	73
TABLE 8: TOTAL 10-YEAR EMISSION COSTS	75
TABLE 9: ESTIMATED DATA COLLECTION COSTS	77
TABLE 10: TOTAL 10-YEAR LAWYER COSTS	78
TABLE 11: TOTAL 10-YEAR ADMINISTRATION COSTS	79
TABLE 12: ESTIMATES FOR LEGISLATION COSTS	80
TABLE 13: ESTIMATES FOR MOTE EQUIPMENT COSTS	80
TABLE 14: TOTAL 10-YEAR AD REVENUE	82
TABLE 15: TOTAL 10-YEAR DEAL-MAKING BENEFITS	114
TABLE 16: BREAKDOWN OF TOTAL ESTIMATED COSTS AND BENEFITS	85
TABLE 17: RISKS OF THE PROJECT	97

1. Introduction

Environmental protection and sustainability have become very important topics in the modern scientific discussion. Since the spread of the Internet in the 2000s, the role of ICT, and software systems in protecting the environment have been investigated with increasing depth. A similar phenomenon is noticeable on the public policy side: Most, if not all developed countries have government-run information portals which provide access to data on pollution, or other environmental data.

Such a large use of software is an overwhelmingly positive thing, however the existing systems have several shortcomings. The most important is that even though environmental information is available, many of these information portals provide access to *raw data*. For the average person, raw data is very hard to understand, which means that he/she will have to invest a large amount of energy to be able to participate in the market. The resulting high transaction costs will exclude most of the population from using the portal.

In addition to information portals, there are quite a few other uses for software in this field. In the literature we can easily find examples of monitoring tools ([Schabauer 2012](#)), corporate management tools ([Jamous 2012](#)), and automatic evaluation of the status of the environment ([Steuer 2010](#)). The problem is, that most people would have to use at least a few of these tools, which might raise difficulties with compatibility. Also, if a solution only influences a small part of a complex process, it is unlikely that the software in question could fundamentally rework the entire process, if needed.

In this paper, we will follow a different approach. First of all, we will begin with one of the most important problems of environmental economics: externalities. We will explore on the main cause of externalities, namely the high transaction costs of internalizing an externality, and the possibility of decreasing those costs by establishing an exchange market for “difficult” transactions. We will use these results as a basis for establishing the requirements of our software system. During the analysis we will consider the participation of the population in this market as an important factor, since they are the ones who pay these external costs. We will also try to present an integrated solution, so that participants could do most of their work in one place. Moreover, we will offer a solution that presents an entirely new way for governments and individuals to manage pollution or use of natural resources. Finally, we will focus on the functions that our system would provide, and not the technology

it would use. In other words, we will try to answer the question “*What should the system do?*”, and not the question “*How would the system do it?*”.

It is important, however, that we do not try to find or claim to have found a software that magically solves all the problems of Coaseian bargaining, without proper distribution of rights and well-designed institutional background. We will propose a solution that provides an infrastructure for deal-making, but our software will be significantly less useful without properly distributed rights.

We imagined a software system that seeks to establish an online exchange system for Coaseian bargains. The system would offer a workflow, guiding the user through the steps of such a deal-making process, while also providing ways to complete other tasks related to transactions (administration, etc.). The software would be a web portal accessible by any browser, but it would also provide ways to integrate the system’s functions into other applications.

The goal of this essay will be to describe the overall structure, and features of this software in a way that focuses on the services that it will provide, and not the technologies that it would use. We will use that description to estimate the effects that the software would have and the costs of making and maintaining it. Our analysis would include an assessment of the risks, and the stakeholders of the project.

In chapter 2 we will briefly overview the theory, namely the Coase theorem, and transaction costs. We will also provide short descriptions of the various practical appliances of the Coase theorem, and give a short overview of existing environmental software, and describe the current problems with larger detail. In chapter 3 we will present our methodology both for describing the software, and for estimating its costs and benefits. For designing and describing the software we will use the Rational Unified Process (RUP), which is a modern software design methodology. For estimating the transaction costs we use the framework of McCann et al. ([McCann 2005](#)).

In chapter 4 we will give an in-depth analysis of the system. We will begin with an overview, which will include an analysis of the stakeholders. Then, we will proceed to modeling the domain the system will work in, in other words we will describe a part of the world that our software interacts with. The third and main part of this chapter will be devoted

to establishing and analyzing the requirements. In this, we will make an extra effort to collect and analyze functional requirements.

In chapter 5 we present the result of our cost-benefit analysis (CBA), and risk analysis. We will provide the CBA in case of a Hungary-, and an EU-sized implementation of the system, and we will estimate both the financial and the social benefit. In chapter 6 we will make a final conclusion, and some recommendations for future research. The Appendix contains several diagrams and tables that are the result of the software design process we used to describe the software (RUP). These are placed in the paper to ensure that a software design team could pick up the project right where we left off.

2. Background

The goal of this chapter is to give a short overview of the Coase Theorem, transaction costs both in theory and practice, and to provide examples of software applications in environmental protection. With the help of this overview we can easily define some of the core problems that we would like to solve.

In the first part of the chapter we will shortly introduce the basic problems of environmental economics and the Coase Theorem. After that, we will follow with discussing transaction costs, detailing the most important factors that influence their magnitude. In part three, we will describe some of the practical appliances of the Coase Theorem that will help to further illustrate the goal of the software. In part four we will review the existing (or currently planned) software solutions for environmental problems in a little more detail.

2.1. The Coase Theorem

In order to talk about environmental protection, we first have to define the environmental conflict. A very good definition is given in Bartus-Szalai, ([Bartus 2012, pages 10-13](#)) who list four conflicts:

Table 1: Environmental Conflicts

Source Problems	Sink Problems
By accelerating consumption of ecologically neutral, non-renewable natural resources, causing them to run out, we hurt future generations.	By emitting pollution or waste we damage the health of the ecosystem.
By expropriating resources that play a role in ecological systems, we damage the ecosystem.	This pollution can also damage human health and artificial capital.

The items in the left column are often called source problems, while the ones in the right column are sink problems. In the future we will only talk about sink-type problems explicitly, usually referring to them as pollution. The reason for pointing this out is that when a software function we will describe will be designed to handle “pollution”, we will implicitly assume that source-type problems will be handled by the described function as well. It is easy to imagine that as far as the software system is concerned, mining could be modeled as the emission of “economic antimatter”, which is a type of pollutant that turns a natural resource into money, destroying both the resource and itself in the process. This generalization is important to make so that we can discuss the system in a simpler way in the future.

Now, that we have defined the environmental conflict, let us give a short explanation of its causes. Environmental economics offers two of the four common market failures accountable for these problems: Externalities, and the Tragedy of the Commons. Externality is, when in a market transaction the buyer and the seller are not the only ones who bear costs or enjoy benefits as a result of the transaction. In this case the social cost/benefit of that transaction includes the benefits and costs of all the people impacted.

The problem of common goods arises when people cannot be excluded from using, or consuming the good, resulting in people unwilling to pay for using that particular good, which will make the market inefficient. However this inefficiency will only remain if an investment required to end slacking would be more expensive than the loss of the individuals. In this situation people would have to tackle the problem *together*, which results in a classic example of the prisoner's dilemma.

A most insightful explanation for why these failures happen came from Nobel Prize-winning economist Ronald Coase who was trying to find an explanation for a seemingly different question:

“..in the view of the fact that co-ordination is usually done by the price mechanism, why such an organization [a firm] is necessary? Outside the firm, price movements direct production which is co-ordinated by a series of exchange transactions on the market. Within the firm, these market transactions are eliminated and in place of the complicated market structure with exchange transactions is substituted the entrepreneur-co-ordinator who direct production. It is clear that these are alternative methods of production.” ([Coase 1937, page 388](#))

Coase was wondering that if the market is always the most efficient, why an alternative is used so widely. The answer he came up with were the “costs of using the price mechanism”, later named transaction costs. Later, in 1960 he discovered that in a zero-transaction cost world harmful transactions simply would not happen, because the private bargaining of parties would always solve the problem efficiently. This argument became known as the Coase Theorem. In the literature there are two forms of the Coase Theorem, often called the weak and the strong form ([Bartus 2012, pages 54, 55](#)):

- Weak form:
 - If the rights of the parties are known and well defined ex-ante

- And the transaction costs are zero
- Then the free bargain of the parties will achieve a Pareto-optimal result
- Strong form:
 - If the two conditions of the weak form still hold
 - And there is no effect on wealth
 - Then the free bargain of the parties will be Pareto-optimal, and invariant to the initial distribution of rights

The Coase Theorem has several important implications. First, the argument Coase was trying to make was that there is little use of examining a zero-transaction cost world. Second, that just because the market doesn't work state intervention is not necessarily needed. This is best summarized in [\(Beckmann 2002, page 2\)](#):

“According to this theorem, private bargaining among relevant actors will solve the problems of harmful effects independent of the initial distribution of property rights as long as transaction costs are zero and property rights are well defined. However, as many environmental and resource economists argue, since transaction costs are not zero, state intervention is necessary. Ironically, Coase introduced the zero transaction cost argument exactly to criticize the logic of government intervention to solve harmful effects advocated by Pigou.”

Indeed, the Coase Theorem shows us, that decreasing transaction costs can be a good way to make markets efficient, and that the intervention of the government might not be needed. Another important result was that Coase defined markets and hierarchies as alternatives for the same transaction. It is an extremely important question to ask then: How do these alternatives differ? Beckmann gives a nice summary of the three main governance structures:

“The market analogy in the case of harmful effects is private bargains for compensation. The ex-ante decision making of the actors is independent and harmful transactions, if they occur, are only co-ordinated ex-post. Thus, only if a harmful transaction takes place the parties' will bargain the compensation. If they cannot reach a decision they may rely on the court to settle the dispute. This governance structure is based on compensation as the main conflict resolution mechanism and has only weak instruments for prevention. The private

bargain mainly relies on the mechanism of incentives and not on administrative controls. However, as Coase (1960) already noticed, private bargain will benefit very much from the liability law.

The firm as an alternative governance structure relies on common ownership or centralized residual control rights. Both activities are decided in the same decision making unit. The potential harmful effect will be decided taking the effects of interrelated activities into account. The firm therefore has more instruments for prevention. If a harmful effect occurs, however, compensation is not necessary. The firm does rely mainly on administrative controls to handle problems of harmful effects and not on incentives. Furthermore it enables simultaneous adaptation. Dealing internally with harmful effects does not require support by the liability law.

The hybrid mode of governance, finally, is characterized by ex-ante coordination but instead of common ownership, a long-term contract or some hybrid organization is established that mediates the conflict. This hybrid form of governance enables ex-ante coordination to prevent harmful effects but at the same time use instruments of compensation. It combines incentive instruments with administrative control and simultaneous with autonomous adaptation. Hybrid modes of governance are supported partly by the liability law.” [\(Beckmann 2002, page 13\)](#)

In his 1960 paper Coase discussed alternatives to the Pigovian methods of dealing with harmful transactions. From those recommendations come the two basic tools of Coaseian environmental policy: The first is rights allocation: By making rights well-defined, and trying to make transaction costs small, we can make the Coase-Theorem approximately true, so that parties could find the optimum in private bargains. The second tool is responsibility rules and compensation. In this case transaction costs are too high, or can't be reduced, and it will be the task of courts to find the optimum. [\(Bartus 2012, page 79\)](#)

Now we can see the clear contrast between Pigovian and Coaseian environmental policy: While Pigou relied on the government to find and enforce the optimum, Coase said that the government's task is only to establish a legal infrastructure to allow private bargaining. We would also add, that in our view the government's task is not to provide *legal* infrastructure alone, but *all kinds of infrastructure* that would help the deal-making process. In this case the software that we will discuss can be seen as an extension but also an organic part of Coaseian environmental policy. It is also worth noting that the fact that such system does not exist could very easily be a result of the tragedy of the commons: A software is a very expensive

tool, and it is very unlikely, that there is a single transaction where the benefits of one party would be higher than the cost of development.

2.2.Transaction Costs

Since our goal is to create a software system that can significantly reduce transaction costs in Coaseian bargains, we will need to understand what the factors that drive transaction costs are. But first, it is important to clarify what we mean by transaction costs. According to Allen ([Allen 1999](#)), in classical and institutional economics transaction costs have a different meaning. Despite that, both definitions of transaction costs are linked to property. So we will begin by introducing Allen's definition of economic property rights ([Allen 1999, page 6](#)):

Property Rights: the ability to freely exercise a choice over a good or service.

Following from that, the neoclassical, and the property rights approach have quite different definition of transaction costs ([Allen 1999, pages 6 and 9](#)):

Table 2: Transaction Cost Definitions

Neoclassical	Property Rights
Costs resulting from the transfer of property rights	Costs of establishing and maintaining property rights

In the future, we will use the property rights definition, since it will be convenient for our goals, since we can include all the costs that come from establishing and maintaining our system, including the change in the costs of deal-making. Now, that this is clarified, we will move on to our next target, which is to find a good typology of these costs. To do that let's go back to Allen:

“Coase provides examples of what he meant by the costs of the price mechanism: discovering what the prices are, negotiating and closing a contract; and he hints at problems of enforcement, but he stops short of any definition.” ([Allen 1999, page 3](#))

Coase basically mentioned three parts of transaction costs:

- Search and information
- Bargaining costs
- Enforcement Costs

These parts, however only include costs that result from transferring a property right, so this is a “neoclassical” approach. “However, as noted by Krutilla (1999, p. 250), *“transactions costs terminology has also been construed more broadly to refer to any costs associated with establishing, administrating, monitoring or enforcing a government policy or regulation.”*” (Barbier 2011, page 60) Figure 1, extracted from McCann et al. (McCann 2005, page 530) illustrates this point perfectly. One can easily mention costs associated with (B) or (C), in our case these could be costs of designing a software in the case of (B), and difficulties caused by vested interests in the case of (C) (Barbier 2011). The typology of transaction costs we will use are the laid out in McCann et al. (McCann 2005, page 533):

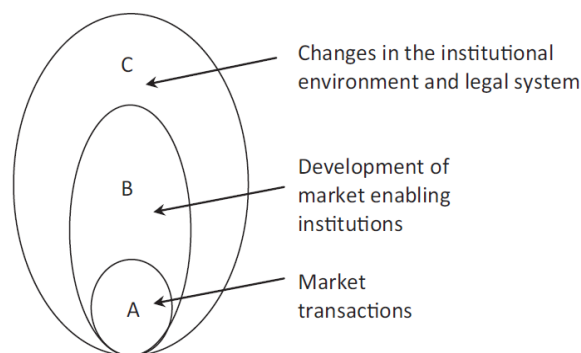


Figure 1: Types of transaction costs
Source: McCann, 2005

1. Research, information gathering, and analysis associated with defining the problem;
2. Enactment of enabling legislation, including lobbying and public participation costs, or, alternatively, the costs of changing laws through the courts or modifying existing regulations;
3. Design and implementation of the policy, which may include costs of regulatory delay
4. Support and administration of the ongoing program
5. Contracting costs, which may include additional information costs, bargaining costs, and decision costs, which are relevant when a market has been set up for a pollutant, or natural resource;
6. Monitoring/detection, which may include both the monitoring of the environmental outcome, or the level of compliance with the regulation, tax/subsidy scheme, or private contract, as well as the development of monitoring technologies;
7. Prosecution/inducement/conflict resolution costs incurred if lack of compliance is found.

Now, that we have introduced the typology we will use, it is time to focus on the factors influencing transaction costs. The most important work in this area comes from Oliver E.

Williamson. In his work, he identified four important determinants. ([Williamson 1975, 1985, 1991](#))

- Behavioral attributes of actors
- Attributes of the transaction
- Type of governance structure chosen (market, hybrid and hierarchy)
- Institutional environment (property and contract law)

Williamson identified two important behavioral attributes: bounded rationality and opportunism. In our work we will consider the latter as the more important problem of the two. To illustrate the importance of opportunism, let us quote Professor Brad DeLong:

“Now market systems can and do go awfully, horribly wrong. But when they go wrong it is because markets are very good at getting people to respond to incentives, and they go wrong when we do not like the pattern of incentives that the market provides.

One major potential flaw is if we do not like the distribution of wealth. Another is that, even when the distribution of wealth is right, the market system can provide the wrong incentives. The brilliant Ronald Coase of the University of Chicago--still productively at work as an economist even though his age is now in the three figures--said that any arrangement of property rights will do about as well as any other. He adduced as his canonical case the locomotive that occasionally throws off sparks that burn the nearby farmer's crops. If the railroad has a duty of care not to burn the crops, Coase said, the railroad will attach spark-catchers if it is cheap and makes sense to do so and pay damages if it is expensive and doesn't make sense to do so. If the railroad has no duty of care, Coase said, then the farmer will offer to pay the railroad to install spark-catchers, and spark-catchers will be installed if the potential damage to the crops is greater than the cost of the spark-catcher. Thus the same decisions will be made whatever the property rights are, as long as there are property rights. If there aren't--then the crops burn, and lawyers grow fat.

This seems to me to be wrong. If there is no duty of care on the part of the railroad, it doesn't just asked to be bribed to install a spark-catcher. It builds the most spark-generating engine it possibly can to cause as much of a chance of fire as possible. And then it demands that the farmer become its serf before it will install the spark-catcher.

Getting institutional property rights wrong in details can be immensely damaging.”
([Brad DeLong, 2013](#))

Of course, Professor DeLong does not provide a falsification of the strong form of the Coase Theorem, since he talks about a non-zero transaction cost world (since the opportunism of the railroad company makes it so), but it also tells us, that in a non-zero transaction cost world the distribution of property rights can also influence transaction costs, and that rights should be distributed so that they would provide “minimum opportunity to be opportunistic”. Later we will see, that the efficiency of our system will also depend on the distribution of rights.

Going back to Williamson, he also identified the main attributes of a transaction that influence its costs. These are uncertainty, frequency, and asset specificity. The first two are easy to understand: higher uncertainty increases the cost of transactions, while more frequent trades tend to cost less. Asset specificity, however needs further discussion, and was also labeled as the most interesting and important of the three by Williamson himself. ([Williamson 1981](#)) The problem with asset specificity is, that if an asset that is traded is very unique, then it is not only harder to find, but also propels its owner into a monopolistic situation, which leads to inefficiency. Williamson identified three types of asset specificity: Site specificity, when the deal is bound to a specific location, physical asset specificity, when for example a special component is needed for production, and human asset specificity, when a special knowledge is needed. In our case the most important one will be site specificity, since most local environmental deals have this problem on some level.

Another problem, quite related to asset specificity is the holdup problem. This problem is wonderfully described in ([King 2007, page 982](#)):

“As Coase himself noted, when companies must make an investment to reduce their environmental impact, ex post transaction costs become much larger (Coase, 1960). Returning to the case of sparks from railroad engines, suppose that reducing the threat of fires requires a capital investment in a filter to catch sparks before they leave the engine’s smokestack. In order to make this investment, the firm would need to be confident that stakeholders would provide them with an ongoing stream of payments for the use of trains that incorporated this technology. However, once the company had made this investment and installed the filter, stakeholders would have little incentive to provide any such payment.”

As the paper also notes, this problem can be solved by either having the environmental stakeholders invest into the asset, but this would only reverse the problem unfortunately.

Another solution would be that the company, and stakeholders invest into a bundle that combines environmental, and financial advantages. ([King, 2007](#))

As Williamson discovered, the institutional environment is a significant determinant as well. Namely, the inflexibility of bureaucracy can significantly raise transaction costs, as well as the uncertainty of the legal environment. Another factor that often raises transaction costs is the problem of too many parties in a negotiation, which makes negotiation extremely difficult. It is important to note here, that a favored semi-Coaseian policy tool, namely the tradable rights system (which will be discussed later) can work well only in situations where the frequency of the trade is not too low, and asset specificity is not too high. In all the other situations bargaining or litigation is required.

2.3. The Coase Theorem in Practice

There aren't many practical appliances of the Coase theorem, since many environmental economists and policymakers consider it an interesting, but useless theory. Still, there are a few cases we will have to mention. First, the courts are an important place where many legal battles over pollution or other environmental damage take place. In theory the courts could use one of the three principal theories of action: Nuisance, Trespass, of Negligence. According to ([Juergensmeyer, 1967](#)) air pollution was labeled as a nuisance at least as early as 1611. The appliance of nuisance to environmental problems has very important economic basis: First of all, they use a very important insight provided by Coase: That in the case of externalities the line between the polluter and the polluted is ambiguous, because an externality requires *both* parties. ([Bartus 2012, page 42](#))

Based on that courts apply the doctrine called *come to nuisance*. Come to nuisance means that if one of the parties moved to their present location knowing about the negative externality (they came to nuisance) then the courts will refuse their demand for compensation, since the price of the site they bought already included the price of the negative externality. ([Bartus 2012, page 41](#))

In other cases the courts will perform a balancing of equities to decide who should pay for resolving the externality: *“What the would-be private pollution controller finds, whether he proceeds in nuisance, trespass, or damages for negligence, is that his claim of right to pollution-free air will be subjected, along with the polluter's claim of right, to a balancing of the equities in which the court will attempt to measure benefits and costs to each, and,*

*hopefully, to society as well. The plaintiff's success depends upon how much importance those who make, apply, and interpret the **law** attach to control of pollution as well as the technological capability and cost of controlling it.”* ([Juergensmeyer 1967, page 1154](#))

2.3.1 Emission Trading

Emission trading is probably the most important appliance of the Coase Theorem as of now. It must be said though that it is not a clearly Coaseian approach, but a combination of command and control with rights allocation. ([Bartus 2012, page 86](#)) The point of this system is that the regulator determines the maximum overall level of pollution in a “bubble”. Then the government issues enough permits to allow for the maximum pollution, and distributes them among polluters. After that, these rights can be traded by anyone – this is why this system is at least partly Coaseian.

Emission trading systems can be categorized by many properties, of which the most important is the method of the initial distribution of rights. There are three major methods described in ([Beliczay, 2003](#)):

- Auction, in which companies have to pay for the rights.
- Grandfathering, in which companies get free permits based on their emission in the reference year.
- Updating, in which companies get free permits based on their pollution at the beginning of every stage.

There are significant differences between these methods. In short, updating does not provide enough incentives to companies to reduce their pollution, and grandfathering gives them opportunities to lobby for an increase in their share of permits. Auction has also lower social costs, and does not disadvantage new entries to the market. ([Beliczay, 2003](#))

Emission trading systems are established in reality as well, one prominent example being the EU’s own Emission Trading System, which distributed rights mostly for free. The EU ETS was established in stages: The first stage was a learning stage, in which a requirements were weaker than they would eventually be. In stage two, the actual requirements were used. ([Beliczay, 2003](#)) Stage two ended in late 2012, which means that the EU has evaluated, and revised the functioning of the system based on that experience, so that it could work better in phase three. When phase three began, there was a surplus of allowances which made the EU

to postpone auctions they were planning. They also devised a few ways to bring supply and demand in line: [\(Structural reform ...\)](#)

- Increasing the EU's greenhouse gas emissions reduction target for 2020 from 20% to 30% below 1990 levels;
- Retiring a certain number of phase three allowances permanently;
- Revising the 1.74% annual reduction in the number of allowances to make it steeper;
- Bringing more sectors into the EU ETS;
- Limiting access to international credits;
- Introducing discretionary price management mechanisms such as a price management reserve.

There is an additional real-life trading system, which is the Voluntary Carbon Credit Market. [\(bBlue ...\)](#) Voluntary Carbon Credits are created by actual reduction in emission that is caused by investment into greener technologies. These credits are traded similarly to permits, and are used to finance these green investments, while they also represent advertising value for companies. This is an example of Corporate Social Responsibility (CSR), which is a process in which companies include environmental considerations into their decision making process. Companies engaging in CSR are usually ahead of the current environmental standards and regulations according to Szlávik [\(Szlávik 2005\)](#).

2.3.2 Local Bargains

There are a number of local bargains recorded, that either failed because of high transaction costs or were only struck after expensive and time-consuming litigation. We will shortly examine two of them here, because understanding the problem might help us avoid them in the future.

The first case is the Sziget festival, which is a week-long music event hosting numerous live concerts in the heart of Budapest. The loud concerts lasting well into the night have obviously angered the citizens of Budapest, especially the district of Újpest, which is near to the festival's location. The mayor of Újpest first launched litigation against the festival called Sziget in 2002, but he lost in first and second instance as well. In 2006 he launched another lawsuit, with both the first instance, and the appeals court ruling in favor of Sziget again. The major then turned to the Supreme Court. In 2011 the new mayor of Újpest managed to strike

a deal with the organizers of Sziget, in which the festival pledged not to surpass an 55 decibel threshold (or pay daily damages), and to install sound insulating tents, and to rearrange the concert locations to minimize noise pollution. ([Zipp ...](#)) The case could be made that with the right infrastructure and tools that deal could have been struck in 2002.

Another example is the case of the Garé Incinerator, which is described in detail in ([Bartus 2006, pages 10,14](#)). In the late 1970s a landfill for hazardous waste was established in Garé, a town in south Hungary. A few years later the inhabitants discovered that the hazardous waste (mostly PCB) caused extreme environmental damages. To avoid the transportation of hazardous waste to an incinerator, the owners of the landfill considered establishing an incinerator on Garé. Therefore, the people of Garé made a deal with the company that would build the incinerator, to ensure that the incinerator would not harm their health. However, they left other surrounding towns out of the deal, who also had possible damages from the new incinerator, which caused a massive campaign against the incinerator. The incinerator was never built, and the waste was transported and burnt in another place.

Both cases show, that with ambiguous property rights, and no infrastructure for bargains, making such deals is difficult. The second case also shows, that if not all parties are included, the bargain will be likely to fail. On the other hand, even in these circumstances successful bargains are made. Most examples include a company, and an environmental stakeholder group making a bargain, that results in a win-win situation for the parties: Usually the stakeholder group shares in an investment that is not only greener but also a more cost-effective alternative for the company. These cases are:

- Environmental Defense and McDonalds devised new packaging, and food processing methods. ([King 2007](#))
- Greenpeace and Foron created and popularized hydrocarbon refrigerators. ([Stafford & Hartman, 2001](#))
- The Alliance of Environmental Innovation helped UPS to create a new type of delivery box. ([King 2007](#))

2.4.The Role of ICT in Environmental Protection

In this chapter we will give a short review of existing and planned computer systems that are designed to aid environmental protection. We will divide these into two categories:

Information Systems and Tools. Both of these systems are especially important from the perspective of transaction costs. If people are affected by an externality, but they are not aware of it, do not know or understand the cause of that externality, or are not aware of their alternatives, and have no way to easily learn about *all three*, then transaction costs will be extremely high (high enough to make sure that there will be no transactions at all). Tools can help further reduce the costs of the transaction, once the problem of insufficient information has been solved.

Let's look at information systems first. We will start by a system that is not finished yet, called INSPIRE. [\(INSPIRE\)](#) INSPIRE is an EU directive, seeking to establish a common infrastructure for uniform spatial information access across the EU. Spatial information is something that environmental applications (including ours) will excessively rely on. The INSPIRE directive addresses 34 spatial data themes that are divided into three annexes: In Annex I there is mostly geographical information about "places". (Administrative units, addresses, etc.). Annex II includes geological information, land cover, and elevation, while Annex III mostly includes data on natural and artificial capital. (Buildings, species distribution, resources, etc.) As we mentioned before, INSPIRE is currently in development, according to the EU, it will be ready by 2019.

Most information systems around the world are pollution databases. They mostly differ in scope, but also in the quality of the interface. There are several examples of national level databases, like [\(TEIR\)](#) (Területi Információs Rendszer) and [\(OKIR\)](#) (Országos Környezetvédelmi Információs Rendszer) in Hungary. TEIR is a spatial information database, which contains interactive tools for data search, but it is difficult to access. OKIR is the pollution database of the Ministry of Rural Development, and is free to use without registration. It includes really intuitive, easy-to-use tools to find pollution near one's location. Another example is [\(PortalU\)](#) which is the environmental database of the German government. PortalU is examined in-depth in [\(Konstantinidis 2009\)](#).

There are quite a few examples of higher-level databases, for instance there is [\(GMES\)](#) (Global Monitoring for Environmental Security), which is a monitoring database, operated by the European Space Agency. The database provides data on atmosphere, water, agriculture, disasters, etc., but that data is really hard to access. I had similar problems with the [\(CEIP\)](#) database, which is an EU level emission database. It offers to display emissions on a

geographical map, similarly to OKIR, but the data resolution is low in many cases, and it is very hard to get information out of it.

Before we draw a conclusion on existing databases, it is important to note, that environmentally themed product databases would also aid reduction of pollution, however, most existing databases ([Database...](#)), ([Environmental...](#)) offer little of that. Our conclusion is, that while there is an extremely large amount of good information available for people on the internet, the method of presentation could be improved significantly. The problems are lack of user-friendliness, the difficulty of getting information, and on many portals information is scattered, you cannot access “all there is to know about a place” with just one query. This makes search tedious. However, there are several attempts at creating integrated information portals: In fact, PortalU is actually the integration of Federal information portals ([Ebel, 2009](#)), and there is a quite rich literature on the idea of SISE (Single Information Space for Europe). ([Pillmann, 2009](#)), ([Kruse 2009](#))

The most crucial problem with existing systems is, that most of these information portals provide access only to what we would call *raw data*. The problem with raw data is that it provides information, *but not understanding of the information*. This means, that average people must invest a large amount of energy to understand the information in order to participate in this market. If the energy required is very high, then very few people will decide to participate, and the market will not function well.

Let's talk shortly about tools as well. The tools aiding some part of environmental protection are numerous. One such example is pollution propagation/transmission modeling. ([Daly 2007](#)) and ([Riecken 1995](#)) present a collection of air and water pollution models that are used in real life, while ([James 2001](#)) models the quality of coastal waters. We can see, that these models usually require little other than meteorological or simple geographical data. It is important to note, that though the referenced articles might seem very old, the Environmental Protection Agency in the USA is using several of these models (or updated versions of the models). ([EPA Air](#)),([EPA Water](#)) Another example for pollution modeling is the company Vibrocomp ([Vibro](#)), who develop software that models air, noise, and radioactive pollution.

There are several other tools available as well, for instance in ([Steuer 2010](#)) they use satellite imagery to evaluate the state of the forests automatically, in ([Jamous 2012](#)), they develop a tool that aids environmental management for companies, and there are tools for

monitoring as well ([Schabauer 2012](#)). The ICE ([theICE](#)) e-market data system is also worth mentioning as an important tool that provides trading technologies to its users on a single webpage. There are several types of products available on the ICE, including natural resources, like oil, or gas, and emissions.

The problem is with existing tools, that there is not one of them that would provide an integrated solution for striking and managing environmental bargains from start to finish, which could greatly decrease transaction costs. This means that an agent would have to download, install and use several tools for just one purpose.

2.5. Summary

In this chapter we have seen the collection of the problems that contribute to the environmental problems. We discussed market failures, the Coase theorem, transaction costs, and the important factors that make these costs high. We have seen practical appliances of the Coase theorem, of which probably the most important to date is rights trading. We think that rights trading works well, when the problems of asset specificity and low frequency can easily be avoided by creating a stock exchange-like marketplace, i.e. there are enough permits in trade and enough people looking for them. We have also discussed local bargains, and concluded, that the problem of too many parties, and less than full involvement can easily derail a bargaining process.

In our proposed software, we will try to tackle these difficulties: We will try to decrease the problems caused by high asset specificity and low frequency even in the case of local bargains, while provide solutions to the other problems mentioned in this chapter. While explaining the software we will assume that we have access to the data provided by INSPIRE, and the other databases we discussed.

3. Methodology

In this chapter we will present our methodology for analyzing the environmental effects software, and for estimating transaction costs of establishing and maintaining the system we described. To do that, we will use the transaction cost typology established by McCann et al. ([McCann 2005](#)): we will collect the costs that fall in one category, and try to give a good estimate for them. The categories and the costs are presented in table 3.

Table 3: Transaction Cost Typologies

Type	Cost Parts
Research and Information	Data Collection
Legislation	Passing Legislation
Design & Implementation	Development & Maintenance Server Upkeep Carbon Emission Caused by the System
Support & Administration	Legal Support Administration
Contracting	Benefits from lower transaction costs Ad Revenue
Monitoring Detection	Costs of extra monitoring equipment
Prosecution/Enforcement	No change in costs

All of the costs and benefits we estimate will be on a ten-year period, and the discount rate we use is 5.5%. Our analysis will also include an analysis of stakeholders and risks. There is one more thing of crucial importance here. In order to estimate the costs of this software, and the effect it would have on transaction costs we need to know what the software we are talking about actually does. This means, that before explaining in detail how we are going to estimate all of those costs, we need to explain how we will describe the system in a non-technical way we promised in the introduction. To do that we will have to explore software design.

3.1. Software Design

Software design is a field at the crossroads of management and engineering, which focuses on maximizing the chance of success of a software project. It is also a rapidly developing field of engineering, since mass demand for software products is a relatively new phenomenon. Moreover, as a very new science software engineering is just beginning to tackle the special problems that differentiate it from other disciplines of engineering.

In this chapter we will shortly describe the core problem of software engineering, since the understanding of that will be needed to justify our choice of methodology. After that in the reaming two sub-chapters we will describe the methodology of our choice.

3.1.1. The Problems of Software Design

As mentioned before, software design encounters some serious problems that make standard engineering and design approaches inefficient, and incredibly risky. What are these problems? The core of these problems lies in the requirements we expect the product to meet.

First of all, a classic scenario is that a software company has a customer who doesn't really know what kind of software he or she wants. The customer probably has a clear problem that he wants the software to solve, but that is hardly an exact description of the software. One needs to go no further than the author of this paper: At the inception of the idea that we could design a software that tries to lower transaction costs of bargaining, I had no idea what exactly the software would do. The methodology that we are about to describe was needed to transform that vague idea into a specific description. If the software company would spend a large amount of money to develop a software based on such an empty description, then it would likely end up with a product that the customer won't want. ([Evolutionary, 2008](#))

Second, even if the requirements are established well in the beginning of the development, they might change along the way. The customer might change his mind. The legal, or the business environment might change. The product might have intense competition, which means that new features would have to be added along the way. If we apply a methodology that is not prepared to tackle these events, we are going to spend high amount of money to build an inferior product. ([Evolutionary, 2008](#)) The third problem is partly a consequence of the first two. The problem is, that in their initial phase software projects are associated with enormous uncertainty. The actual time and money required to complete a software project is usually between 25% and 400% of the initial estimate.

One of the simplest design models is the Waterfall model. The waterfall model is really easy to describe: It involves five basic development steps, performed consecutively:

1. Establishing Requirements
2. Design
3. Implementation

4. Testing
5. Transition

It is important to note that this methodology, though very simple, fails to solve any of the three problems mentioned above. Because requirements are established fully before design begins, and design is fully completed before implementation begins, it is very painful to change requirements in a later phase. There is also no feedback to the developers, since the customer will only see the software in the transition phase, but if requirements were not established correctly, then the developer wasted his time to create a software that nobody actually wanted. To underline this point a research conducted in 1994 (when waterfall was a dominant methodology) showed that software projects had a success rate of 16% (and over 50% of software projects were “completely unsuccessful”), and in 2002 still about 45% of the features of an average software were never used. ([Standish Group 2010](#))

The solution to these problems were agile methodologies. Agile methodologies in opposition to “rigid” ones, perform the five development steps mentioned in the waterfall model *parallel*. These methodologies describe an iterative and incremental software development process. This means, that the software is developed in iterations, and at the end of each iteration we have a software that satisfies a bit more of the initial requirements. If requirements change along the way, then the next iteration will work with the changed requirements. It is also important, that the end of each iteration serves as a checkpoint. The stakeholders of the software are active participants in the development process, and they review the result of every iteration. This helps the developers to actually create a software that the customers want. Also, this flexibility allows software developers to focus on difficult, or uncertain parts of the software first, without even touching the easy ones. This way we can reduce risk very early in the process.

Agile has a few other characteristics: It prefers collaborative development, and also prefers to mark milestones with working software, and not excessive documentation. This also presents a weakness: If a software is to satisfy a standard, (for which requirements are well-established, and documentation is essential) then “rigid” methodologies still outperform agile ones. In all other fields, however, agile methodologies have been an overwhelming success: Above 50% off all agile software has been a complete success, with failure rate being just 6%. ([Benyó, 2013, slide 51](#))

3.1.2. The Unified Process

The Unified Process (UP) is a popular agile software design framework. There is a well-known refinement of the UP, called the Rational Unified Process (RUP). The RUP is not really a design process, but a framework that has to be fit for a given task. However, the RUP is also a full framework, for it includes recommendations not only for the design process, but for project management and documentation as well. During the design process we will have to create *artifacts* which are basically end products describing our software. The appendix contains a detailed description of these artifacts. In this chapter we will summarize the main characteristics of RUP. The first important feature is, that RUP relies on object oriented analysis and design:

- Object oriented analysis: Collection of things related to the problem, their properties and relations between them.
- Object oriented design: Defining the elements and objects of the software to be implemented. Specifying the attributes and collaboration of these objects in order to create a system that satisfies the requirements.

The RUP is also architecture centered. ([Rational 1998](#)) This means that we try to select an architecture that is robust (easy to change/responds well to change). It is also, as an agile methodology iterative, and incremental. The iterative part is easy to imagine as “many little waterfalls”, which helps to reduce risk early in the process. The incremental part means that at the end of each iteration the software becomes more completed. But we only enhance the result of the previous iteration, the software doesn’t become more “complicated”. Another important thing is that at the end of each iteration we must have a working software. A software that might not do everything it needs to do yet, but it must be functional nevertheless.

The most important property of the RUP is that it is use case-driven. ([Rational 1998](#)) A use case is a function that the system provides towards the outside world (to a user). (In other words a use case is a functional requirement.) So, for example a library’s computer system would have use cases like “Rent a Book” and “Register New Member”, etc. Use case driven design means, that during the design process, *we have to justify every decision with a use case*. To give an example: In the library system we would have to store data on people (our members). Every person has a birthday, so we could store their birthdays in our system. Should we? The RUP tells us to look at the use cases we have. If one of the use cases requires

us to know the members' birthdays, then we should store them. If not, then we must not store the birthdays. This might seem like a triviality, but it is way too easy to make a mistake, and end up making software functions that nobody asked us to.

The RUP has a two dimensional structure. [\(Rational 1998\)](#) Along one dimension it is divided to separate engineering disciplines. The other dimension is time: The design process consists of four phases, each phase consist of one or more iterations. The disciplines and phases of the RUP are shown in figure 2.

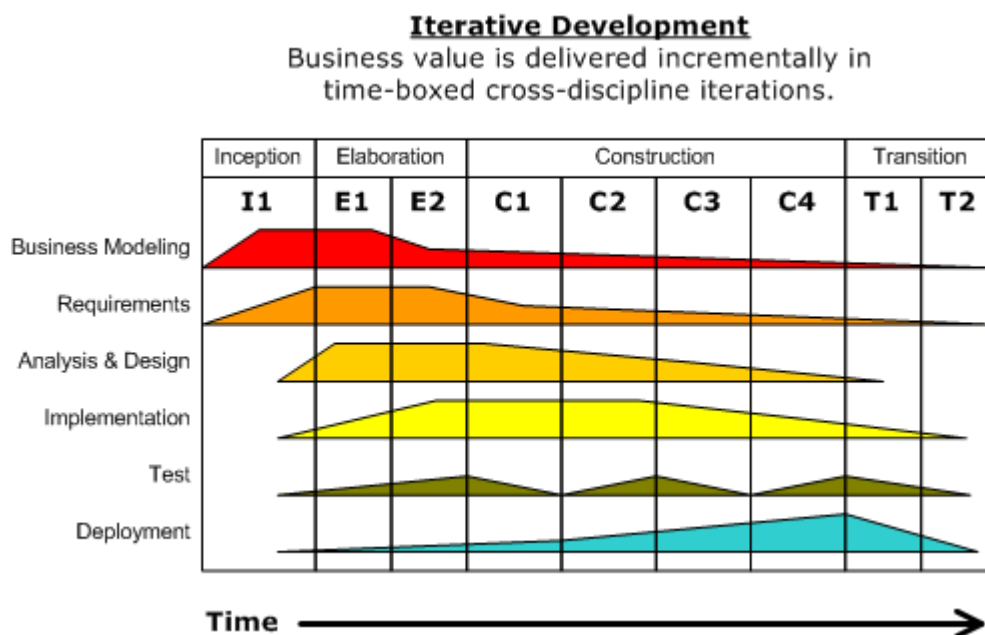


Figure 2: The Phases and Disciplines of RUP

Source: (Rational 1998)

The two disciplines we need to talk about are Business Modeling, and Requirement Analysis. First let me introduce an alternative name for Business Modeling: Domain Modeling. The reason for this is to avoid confusion that the name “Business Model” usually stems. Domain Modeling is not the discipline of making a software profitable. It is about describing, and understanding the world or environment that the software will interact with. To create a domain model we will have to collect the “things” in the environment of our software, the properties of those things, as well as the connections between them. Requirement Analysis is about collecting and describing the requirements we want our product to satisfy.

By now, we can see that in order to describe what our software should do, or to answer the question of how large it would be, we will need to perform, or at least start the disciplines of Domain Modeling and Requirement Analysis. To do that we will more or less perform the

Inception phase of the RUP. The recommended documents of a complete inception phase are as follows: [\(Rational 1998\)](#)

- A vision document: a general vision of the core project's requirements, key features, and main constraints.
- An initial use-case model (10% -20%) complete).
- An initial project glossary (may optionally be partially expressed as a domain model).
- An initial business case, which includes business context, success criteria (revenue projection, market recognition, and so on), and financial forecast.
- An initial risk assessment.
- A project plan, showing phases and iterations.
- A business model, if necessary.
- One or several prototypes.

As you will see, we will provide most of these documents (they are in the Appendix), except for the project plan, since that would require knowledge of the development team. In future chapters we will also explain what these documents are, and what purpose they serve.

3.2.Estimating Transaction costs

By now we know that by describing the software in a non-technical way, we basically meant producing the artifacts of the inception phase. Now that we have cleared that hurdle, it is time to return to the question of estimating the transaction costs. As mentioned in the introduction to this chapter, we will use the typology provided by McCann et al. [\(McCann 2005\)](#)

In the research and information bracket the only cost we mentioned was the cost of data collection. To be able to give an estimate, we have to know what kind of data is needed. Luckily, we will describe not only the software, but also the domain, that the software will work in, so from that description we will know what kind of data that our system will use. As a second step, we will have to find out what data is missing, since data that is available in other databases, needs no collection. The third step is to estimate the effort in man-hours or man-months, and turn that into a cost by multiplying it with average salaries.

The second type was enactment costs. It is not yet clear what kind of legislation the system would need, and whether it is likely to work well without any serious legislation. But still, we will include the cost of passing an “average” law, since we strongly recommend a

general distribution of rights, which needs legislation. To do that we will use the methodology of ([Wilson 2012](#)) who estimated the cost of passing a law in the New-Zealand parliament. The study mentions, that passing one extra law has likely very little extra costs in the short run, but we will still estimate the social cost of passing one law.

Most of the costs of running the software are in the implementation category. To estimate the development effort in man-hours, we will use the methodology presented in ([Carroll 2005](#)), which simply assigns difficulty points to actors and use cases, and includes modifying factors based on technological and team-experience variables. Based on a study on software maintenance ([Jones 2006](#)) the maintenance costs in a year can be estimated based on the development cost. The other two costs are server upkeep, and carbon emission caused by the system. It is important that both of these costs depends on how much data transfer our software will perform over the internet. That depends largely on three factors:

- The number of users who use the system
- How they use the system (the number of transactions they request, frequency)
- What kind of transactions they use (Simple data request, Simple page loading, Complex computation, Multimedia, etc.)

We will use website analytics ([Alexa](#)) to estimate the number of monthly users, using magyarorszag.hu (or more precisely, Ügyfélkapu) as a reference. We will use our descriptions we made with the RUP to estimate the remaining two factors. Based on that we can give an estimate for the number of servers needed, by looking at websites with similar content or functions. A survey ([What the web's...](#)) has data on a few relatively different websites. If we have the number of servers we can use a web tool (referenced in ([Determining... 2005](#))) to estimate the 10-year total cost of ownership (TCO) of a server rack (which is roughly 40 servers).

Now, that we have estimated the amount of data transfer, we can use a study, that estimated the carbon emission of internet advertising. ([Taylor 2008](#)) This study includes an estimate for the GHG emission per data transfer. The study is relatively old (2006), but based on the trend of the computational efficiency of server machines we can update the estimate.

The next category of costs is administration and support. Based on the use cases we will be able to estimate what kind of administration or support tasks the software will require, and

from that we will be able to give a rough estimate for the number and skill of the staff required.

Another very important category is contracting, because contracting will include the financial and social benefits from the software. The first benefit is ad revenue. To estimate this benefit, we will use a study of small websites, and a web calculator based on it. ([Real Revenue ...](#)) To estimate revenue, this method needs the following:

- The content type
- The audience type
- The willingness of the audience to spend
- The revenue source
- The number of monthly users

The other benefit in this category are the costs of deal-making. Or more precisely, the difference between bargaining costs before and after the system was established, multiplied by the number of transactions. The problem is all three elements are very difficult to estimate, since there are hardly any studies or articles about “average” transactions. There is a way around this problem however: Since we will have a description of the system’s features, we will be able to break down the process of deal-making into several activities and estimate the time required to complete each. This is the most important benefit of having the system designed. We can of course give an estimate for how long these activities would take without the system, but such estimates will be highly uncertain. If we have the effort of a transaction in man-hours, we can turn that into a cost by using average salaries. Also, we could estimate the number of monthly transactions as a percentage of our monthly users in order to get the total costs or benefits of deal-making with our system.

The last category is monitoring. In monitoring we will include the cost of extra equipment needed for monitoring. We will simply estimate the number needed based on the system’s functions, and multiply it with the market price. For this estimate we will ignore the costs of maintaining the equipment.

Of course, there is one more category in McCann et al., namely Enforcement, but we did not include it here, because this system will not change anything in that area. (And the enforcement costs of individual bargains will be naturally included in the bargaining costs.)

3.3.Risk analysis

This very short chapter will be devoted to describe the process of early risk analysis. During this process, we collect all the risks that might affect our process or the product. When this is done, we put these risk into categories based on their probability and their impact. Then the risks with high chance of occurring and large impact need to be handled in the following iterations. This usually means that an aversion plan is created or that the risky features are implemented first, or a proof of concept version of the software is created early in the process.

The impact categories we mentioned before are the following:

- Insignificant
- Tolerable
- Serious
- Catastrophic

Also, we often collect risks by types:

- Technological risks
- People risks
- Organizational risks
- Requirements risks
- Estimation Risks.

4. Software Analysis

In this chapter we will start the design of the software we expect to reduce transaction costs of Coaseian bargaining. To do that we will use two of the disciplines of the Rational Unified Process: Domain modeling and Requirement analysis. With the help of these we aim to help the reader (and ourselves) understand how the software would work, and also to reduce the incredibly high uncertainty of estimation. It is worth noting here, that even if we manage to reduce this uncertainty by half, we are still in danger of underestimating the time and cost of the software by half. That is not something we can ignore in the future analysis.

In the first part of this chapter we will give a general overview of the software product, which will include the most important elements of the artifact Vision ([Appendix 5.](#)). In the second part we will construct a Domain model to help us understand the environment of the system. In the third, and largest part we will collect, and analyze the requirements. We will select a few of the most important features, which we will describe in depth in order to provide the input to our cost or effect estimating methods.

4.1. Overview of the Software

In this chapter we will give a coarse overview of the software system. We will begin by stating the business opportunity and the problems the system aims to solve, and other general questions. We will then collect, and analyze stakeholders, while at the last part of the chapter we will shortly describe the main features of the software, as well as its structure.

4.1.1. General Description

Business opportunity

There are many existing information portals, but only a few of them provides easily understandable or accessible information about pollution or the environment. There is no integrated tool that helps to make Coaseian bargains more effective. Creating such a tool and information portal could have significant social benefits.

Problem statement

Let us make a list of the problems we mentioned earlier:

- It is hard to find all information about the quality of one's environment in one place.

- It requires a large effort to understand the particular effects that pollution has on someone.
- It is hard to find the source of the pollution in one's property.
- It is difficult to find out what one can do to reduce the pollution in his or her property.
- In the case of new economic activities the process of impact assessment, and/or authorization can be long and expensive.
- There is no established infrastructure for negotiating and striking environmental contracts.
- If many are affected by a pollution, then negotiation will be very expensive.

Product owner

The question here is, who would create and run this system. The two possibilities are a private company and the government. Since this project is more likely to have large social benefits than financial benefits, and for some features legislation might be required, we strongly recommend the latter. Whether this system should be established by local, national, regional governments or by the EU itself is a question that will be left unanswered for now.

Users

This system is supposed to have a large variety of users. Environmental stakeholder groups, regulators and professionals will almost certainly use it. If this system is to be a standard platform for making environmental bargains, then companies and other polluters should be required to represent themselves in this system. Of course the most important user group would be the general population.

Promotion possibilities

Promotion of the system is important to engage a large part of the population. If the government is to create this system, then it could raise awareness of this system through TV ads, or by writing a letter. Another promotion possibility is allowing third party developers to integrate some of the system's services into their own applications. Similarly developing smartphone and tablet client applications for the system could significantly raise engagement. Also, if general population is to be engaged, then the system's design needs to be "cool", and needs to be integrated into services that people frequently use. (E.g. Social Network systems)

One additional important point can be made here, which is that the distribution of property rights will largely impact the engagement of the general public. If in most cases rights protect the general public, and not companies and factories, then for most people this system might be a source of revenue, which will be a strong motivation to use the system, even if the individual is not particularly interested in environmental protection.

4.1.2. Stakeholders

In this chapter we will collect the stakeholders of this software, and provide a stakeholder analysis. We decided that we will only consider influence and impact that the stakeholders have on the *software project itself*, and we will not consider their influence and impact on the possible legislation that surrounds it. Table 4 shows the stakeholders and their problems.

Let's continue with assigning influence and impact of the stakeholders. The government is the easiest: since it is the product owner it naturally has a **very large** influence. It also has concerns that will **largely** impact the project, especially because it is in the government's best interest to let end users influence the project. Regulators are a part of the government, or in a close relationship, that means they have a **large** influence. However, they will only use a very small subset of the features, which gives them a **small** impact. Green Technology Companies are interested in the system somewhat, since it will give an opportunity to them to advertise their products, but they will only reach the system through lobbying. This gives them a **small** influence. They would not use the software and are only interested in a really tiny part of it, so their impact is **very small**. Companies have even less interest to lobby to influence the software itself, which gives them a **very small** influence. They would be moderately important users, but they might be required to use the system anyway, which results in a **small** impact. Environmental Organizations can reach the development of the software through lobbying, but they will be very motivated in doing so. This gives them a **moderate** influence. They are also important users, who have concerns that will impact quite a few of the features. This means a **moderate** impact. Finally, individuals will not lobby, but they are the most important group of users, which means they should be involved at every stage of the development. This gives them a **large** influence, and for reasons mentioned above, their impact will be **very large**. Table 5 and Figure 3 show the Stakeholder Analysis.

Later, during establishing the functional requirements, we will largely focus on solving the problems of individuals, since their participation in the system depends largely on that.

Table 4: Stakeholders and their Problems

Stakeholder	Concerns of the Stakeholder
Government(s)	• The solution needs to be efficient and worth it (positive social benefits) • The solution needs to be popular • The solution shouldn't have negative macroeconomic effects • The solution needs to provide alternative ways to manage pollutants
Regulators	• The system should provide an easy way to administer pollution • The system should aid estimating the effects and dispersion of pollution
Companies	• The cost of making deals should be minimal • The solution should offer ways to minimize risks (opportunism, asset specificity, holdup) • Possibility of using reduction of emission for PR
Green Technology Companies	• The system should provide features that aid discovery of their solutions • The price of advertising their products in the system should be reasonable
Environmental Organizations	• The solution needs to effectively reduce pollution • The solution needs to be transparent, easy to monitor
Individuals	• The solution needs to be easy to use, easy to access • Deal making should be easy, and aided by templates • Information should be understandable • Users want to decide how much information they want • The solution should offer ways to minimize risks (opportunism, asset specificity, holdup) • The solution needs to provide ways to reduce difficulties of bargaining (many parties, opportunism) • The solution needs to be transparent

Table 5: Stakeholders

Name	Influence	Impact on Project
Government	Very Large	Large
Green Tech Companies	Small	Very Small
Regulators	Large	Small
Companies	Very Small	Small
Environmental Organizations	Moderate	Moderate
Individual Users	Large	Very Large

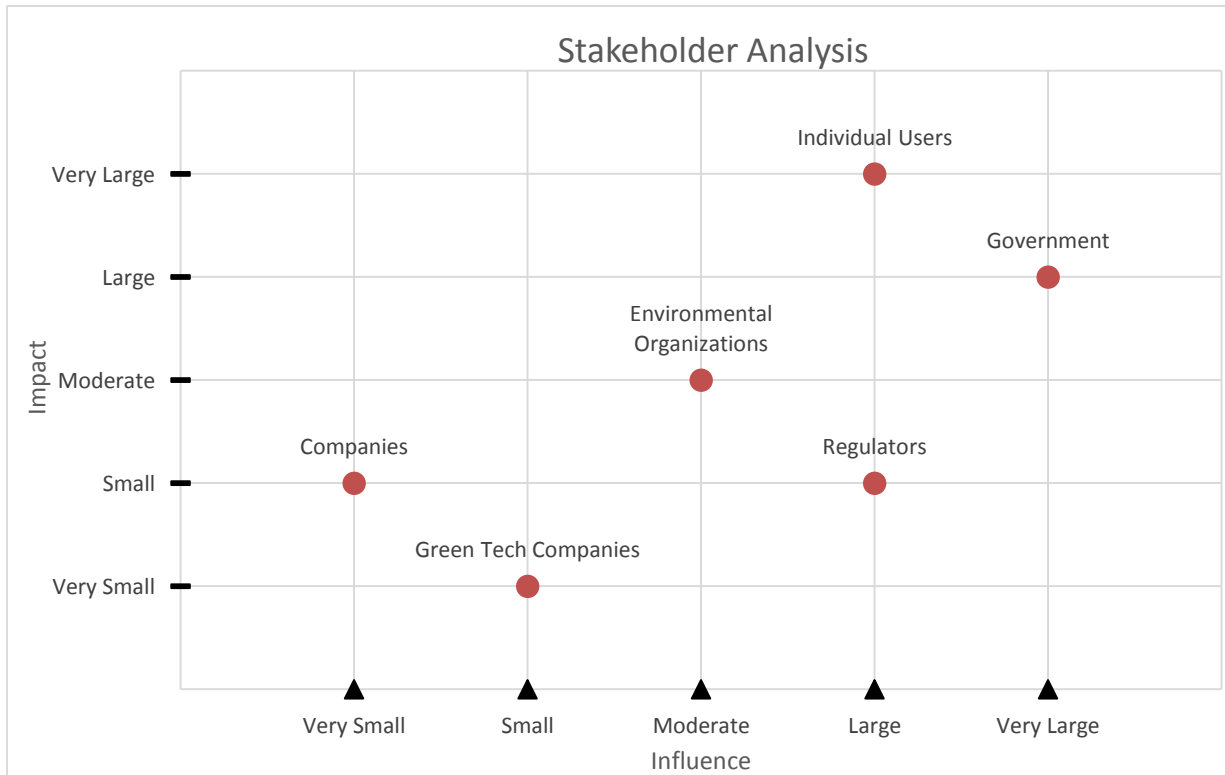


Figure 3: Stakeholder Analysis

4.1.3. Main features, and Structure

In this section we aim to outline the most essential features of the software, and to give a coarse description of the structure. But before that, let's briefly describe what activities are performed during a bargain. First, one party has to register a new activity that impacts the environment. The other party has to find out about the effect, the activity has on it, and link it to the first party. Then the two parties have to negotiate, and both need to make a decision. After that a contract has to be made and concluded. Finally the two parties have to monitor each other, and enforce the deal in case of trouble. Our software has to help both parties in most of these activities. To describe the system in a clean, comprehensible way, we will divide the system into modules. These modules are logically consistent, each aims to impact

one part of a Coaseian bargain. The modules are the following: *Information, Administration, Community, Rights Trading, Contract, Product Information and Measurement & Monitoring.*

We mentioned in the promotion section, that we should provide ways to third party developers to integrate some of our services into their applications. A great way to do this is creating a Software Development Kit (SDK). SDK is a freely distributed program code that provides an interface to a service. By interface we mean both a graphical user interface and a programming interface. With the help of an SDK third party developers can effortlessly integrate our services into their applications. To imagine, how it is used, consider this example: You are the developer of a smartphone application that helps people find hotels at their traveling destination. You would like to display pollution information about the hotels as well. If our system does not have an SDK, the best you can do is put a link to our site next to the hotel. But if we do provide an SDK, then with about two lines of code, you can display the pollution, not to mention that the user could also manipulate that interface element to make it display additional information. Since SDKs need to be done separately for different operating systems, we recommend making four: for iOS, Android, MacOSX and for Windows. With those considerations, the final structure of the system is shown in Figure 4.

The main features of the software would be the following:

- Users can access understandable environmental information
- The administrators can perform quick computer-aided propagation estimation and impact evaluation
- Users can link pollution to the source with one click
- Users can form groups and enter negotiations as a group
- Users can access group decision aiding services
- Parties can use computer-aided negotiating and contracting
- Users can access e-Exchange system for emission permits from within the system
- Users can search environmental information about products, or green technologies
- The system can automatically monitor areas using real-time data coming from mobile measurement units (will be called **notes** from now on)

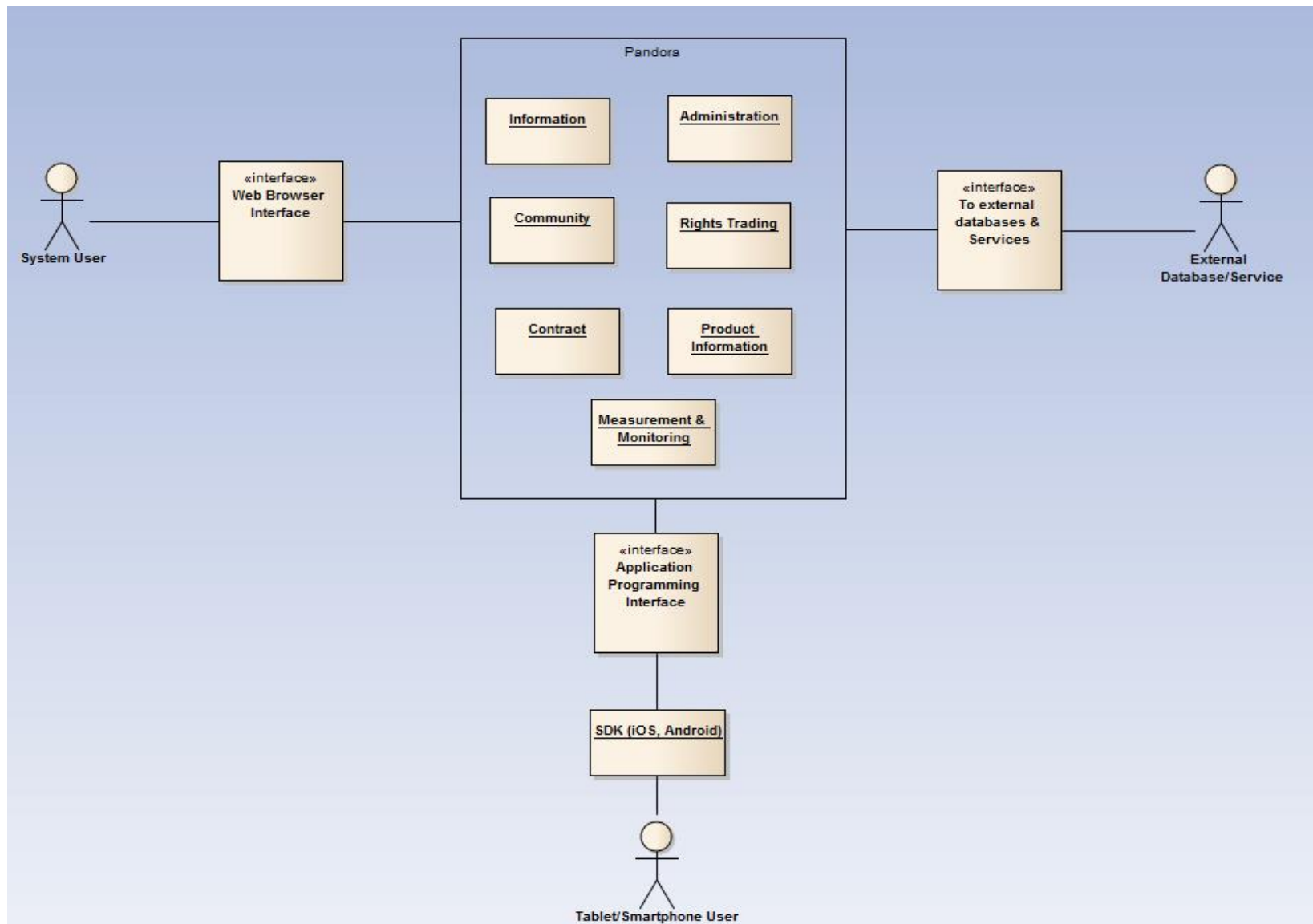


Figure 4: The Structure of the System

4.2. Understanding the Environment – The Domain Model

In this section we will collect the things in the software's environment and the connections between them, in order to construct the Domain Model. It is useful to remember that the purpose of the Domain model is not to arrive at a fancy version of “the theory”, but to arrive at important insights about how the theory looks like from the systems perspective. We will start drawing the domain model by considering how the software will see the pollution chain model. We will then expand our model to make up for the simplifications of the pollution chain model, and to include the activities of rights trading, and bargaining. During the construction of the model we will often refer to “all pollution”, by which we will always mean “all pollution or use of nature that the system covers”, since our system will naturally not cover every possible pollutant.

Before we start building the model, we will need to pull up two important definitions from the Glossary ([Appendix 7.](#)). These are Effect Area and User.

Effect Area: A geographical area where the concentration of a pollutant can be considered constant in space. (Often simply called Area.)

User: Individuals or organizations that use one or more of the system's features.

4.2.1. Importing the Pollution Chain Model

Now, we will start constructing a Domain Model by importing the pollution chain. The diagrams we draw can be seen in [Appendix 2.](#) The first element of the pollution chain is **Demand**. Demand for a product or service is the reason for the existence of the polluting economic activity. It is hard to see however, how our system would use the concept of demand for anything, therefore we will not represent demand in the domain model. Let us proceed to the second element, **Product**. We mentioned among the main features, that we'd like users to be able to find information about products, so our system will have to know about them. Let us refer to Product as Product Type from now on, and place it on the model. The next element is **Technology**, which we also might store in the system. Technology is of course associated with Product Type. Multiplicity depends largely on what we mean by technology. Our take is the following: one Product Type is produced with 1...* (at least one, but at most infinite) technologies, while one technology is used to produce 0...* types of product. Technology is linked to the **Pollutants** it emits. One technology emits at least one

pollutant, since there's no such thing as a completely clean one, so the multiplicity is $1 \dots *$ in this case. One pollutant is emitted by $0 \dots *$ technologies.

The next element is **Emission**. It is important to note, that we understood technology as a possible way of production, and not as an instance of that technology installed in a factory somewhere. Because of this, emission cannot be linked directly to technology. Moreover, we will define two different types of emission: The first is emission unit, the second is emission.

Emission unit: The unit of a given pollutant that is defined by law (the unit used by tradable permits), or the magnitude of a pollutant/use of nature emitted by one user in one area.

Emission: A user's aggregate pollution or other use of nature.

This means that an emission belongs to a User, in a 1 to 1 relationship, and it is a collection of all the emission units that the user is responsible for. Emission units are made of exactly one pollutant (but one pollutant might make infinite emission units), and they are bound to the area where they are emitted. If the given pollutant is not handled by the rights trading system, then an emission unit is the total emission of one pollutant by one user in one area. Otherwise it is exactly the amount of emission allowed by one permit. It is also worth noting, that because of Coase's insight, it is ambiguous which user the emission unit belongs to. This depends largely on law, but for now, let's suppose that emission units always belong to the user whose property the emission took place on.

The next element in the pollution chain is transmission. Transmission is not in the domain model as a class, because it is an association. Transmission is the connection of an Emission Unit and an **Imission Unit**. What is more, to one imission unit there is just one emission unit by definition:

Imission unit: The concentration caused by one emission unit in one area.

From this definition it is obvious that imission units are made up by just one pollutant as well. This also means, that more imission units might belong to one emission unit, since an emission unit is likely to cause non-zero concentration in more areas. Of course imission units in one area make up a total imission.

Imission: The aggregate concentration of all pollutants in an area.

Some versions of the pollution chain model include dose, but like many others, we will skip it. That brings us to effect. First, we need to assign possible effects to pollutants.

Possible effect: An effect that one or more pollutants may cause.

And similarly to the imission unit and imission, we will define unit effect, and aggregate effect.

Unit Effect: The effect of one imission unit.

Aggregate Effect: The aggregate of unit effects in one area.

And we do the same with the last element of the model, namely damage:

Unit Damage: The money-value of one unit Effect.

Aggregate Damage: The aggregate of Unit Damages in one area.

By that time we might ask, why exactly did we define a unit and an aggregate of these concepts? The answer is quite simple. A user might be curious about *all the damage*, and *all the effects in the area the user lives in*. If the user is a bit more of an expert, then we could add all the pollution in that area. Another user (who is more of a polluter) might be curious about all the pollution he emits. These are precisely our aggregate concepts. We want these two users to find each other easily. But the aggregate damage a user suffers is made up from unit damages. Unit damages belong to unit effects, unit effects to imission units, imission units to emission units. Emission units are a part of an emission, and emission belongs to the user responsible for it. This means that the connection between the parties is in the domain model *by design*. If these classes become actual classes in a software database, then our system will be able identify *all relevant parties* in a bargain in the matter of milliseconds.

4.2.2. Expanding the Domain Model

Now, that we have a good basic domain model based on the pollution chain, we need to add other objects to it. The first is **Effect Area**. Remembering the definition provided at the beginning of this chapter, effect area is a geographical area in which the concentration of a pollutant can be approximated as a constant in space. This naturally means, that different kinds of pollutants require areas of different size. The best way to create this is a hierarchical system. And of course, the smallest sized area will always be a single property. To create areas we might use data from INSPIRE when it will be ready. We could also make the

functioning of the system much easier if areas were aligned with administrative areas, when possible. Every area has one imission, one aggregate effect, and one aggregate damage. In the hierarchical system the aggregate effect of a parent area is the sum of the aggregates of the children areas, and so on.

In the initial pollution chain-based model we made two associations that are particularly interesting. Namely, the transmission (linking emission units to imission units), and the association between imission units, and unit effects. None of these are trivial. Transmission is influenced by the dispersion attributes of an area.

Dispersion attributes: The geological, meteorological, geographical, etc. attributes of an area that influence the dispersion of pollutants.

Based on the pollution modeling methods, these attributes are usually geographical and meteorological properties. In the domain model we will represent dispersion attributes as a class that belongs to one area, and that to one area belongs one collection of dispersion attributes, but in the actual implementation of the system dispersion attributes should be stored along a spatial grid, and not inside areas of wildly varying scale.

The other association is influenced by two factors: First, unit effect is determined by the possible effects of the pollutant that the imission unit is made of. Second, the unit effect is also influenced by the natural capitals in that area.

Natural Capital: Natural capital is an extension of the economic concept of capital: Natural capital is the stock and flow of natural resources of ecosystems that provide goods and services of present and future value. ([Török 2010](#))

The introduction of natural capital into effect is easy to understand. If in an area we have a large concentration of a pollutant that causes asthma, but in that area all we have is a lake, then the actual effect of that pollution will be small, since lakes or fish don't get asthma. With that we have a final domain model that can describe the expanded pollution chain.

As the last part of our Domain modeling, we will also draw domain models for rights trading, and traditional Coaseian bargaining. These models are extensions of the pollution chain model, but in order to keep them clean and simple, we will only include the boundaries to the previous model, and focus only the new elements. We will begin by talking about users and communities a little bit: Users and communities will be parties in bargains. Communities

are of course made by users, for a variety of reasons. Communities might be formed to exchange information, or they might simply be clubs, or environmental stakeholder groups. The third, and most important reason is representation. Communities are an attempt to solve the problem of too many parties and to some extent also the problem of slacking. Of course that is much easier said than done, but we will devote quite a few pages to discuss this in chapter 4.3.

Users or communities might hold rights that were distributed at some earlier time. These rights could be divided into two categories:

Protection right: A right that is not directly tradable, but linked to another right.

Tradable right: Emission or Use permit that is directly tradable.

By Tradable rights we mean rights usable in an emission trading system, while Protection rights serve a function when emission trading is not possible, but a local bargain could be a good solution. In this second case parties would strike a contract in which the owner of the right, who would allow some violation of that right in exchange for compensation, or other conditions. Both kinds of rights are only valid on a certain area.

We mentioned contract, but still have to define it. A contract will be mostly based on protection rights, but two parties might strike one without any kind of rights. Also a contract will mostly specify the allowed imission or emission units of a user.

Contract: Legal contract between two or more users that defines the rules and conditions of their use of nature.

With the help of these new classes, we can finally draw the domain model for the pollution chain, rights trading, and bargaining. These models are shown in [Appendix 2](#).

4.3.Requirement Analysis

This section is perhaps the most important in describing the software, because establishing requirements well is essential in the success of the project. We will collect requirements based on the FURPS+ typology, putting functional requirements into the use case model, while putting non-functional ones into the SRS. After that, we will analyze the most important requirements module by module.

4.3.1. The Use Case Model

In this chapter we will present the use case model. To ensure understandability, we will present the models separately for each module. We will also shortly explain, how the modules will work, and will present the use case diagrams for each. The short use case descriptions are found in [Appendix 8](#), while the diagrams are in [Appendix 3](#) for every module.

Information Module

The goal of Information module is to provide easy to understand information about pollution to users. It would include features like, searching for emission or imission data, viewing information about pollutants, and viewing the sources of the pollution at a certain location. The most important feature would be viewing area information. In this use case the user would navigate over a map (very much like in Google Maps), and access interactive content on the map about a certain area. Since the Information module is the basic module of the software, it would also include some basic functionality (registration, login, error reporting, etc.). The module would also require heavy access to other databases.

Administration Module

The goal of the administration module is to provide a solution for system administrators to handle newly reported pollution or transactions. Administrators would calculate the transmission, effect, and the damage of the pollution, also editing the rights associated with users. Users would also come here to connect their accounts to Ügyfélkapu (or another e-Government system) so that they could identify themselves to engage in official business. Also, users could report new planned emission in this module.

Community Module

The goal of the community module is to tackle one serious problem: the case of too many parties. If in any deal there are too many interested parties, managing negotiations will soon become impossible. Users in this module could form and join communities, while the module would provide basic infrastructure for communication (forums and messaging). Users could also form communities with other reasons too: environmental stakeholder groups would represent themselves here, often asking for donations for a certain goal (green investments,

permit buying, etc.). The most important feature however, would be a group decision support feature, which would aid groups to easily arrive at important decisions.

Rights Trading Module

The Rights Trading is one of the simplest modules in the system. It would provide an interface between our system and an emission exchange system that would allow users to buy, sell, and monitor rights. The inclusion of this module will allow users, companies and governments to incorporate tradable rights based solutions in this system as well.

Contract Module

Contract is another extremely important module. It would allow users to negotiate and create contracts with the aid of a computer system. This system would allow users to build terms and conditions easily, and aid the contract building with automatic review, and by offering pre-defined templates for most common deal-types. Users could also easily turn to a lawyer, should it be needed. The members of the Legal Support team would review problematic contracts, and also add new templates, should they be needed.

Product Information Module

The Product Information module is not closely connected to Coaseian bargaining. It was included in the system because it is a wonderful source of revenue. The module would offer environmental information about a variety of products, and allow users to get information about a product simply by showing it to their smartphone cameras. Companies could build interactive product pages, while some of the “cool” interactive elements would only be available to premium users. Basically companies would pay for advertising. The other important feature would be similar to an online store for green technologies. This would allow companies to find alternative solutions for pollution easily, while users could also learn about the new technologies that their neighbors will install. Whoever runs the system might be able to ask for a small share after every piece green technology “sold” through the system.

Measurement & Monitoring Module

In this final module administrators would be able to tell the system to monitor an area automatically. The system would collect data from all the measurement units in that area, and check whether the level of pollution is consistent with the reported emission. Users could also ask for monitoring, should they grow suspicious. One important addition is the introduction

of motes, which are small mobile computers with radio communication and a measurement device attached on them. These would be very handy in conducting a large amounts of measurements in many different locations, with just one device. Also, transportation of these motes can be solved by simply putting them onto a vehicle, or into somebody's pocket.

4.3.2. Non Functional Requirements

In this section we will briefly describe the most important non-functional requirements. All the requirements can be found in the Supplementary Requirement Specification (SRS) in [Appendix 6](#).

Usability

- User interfaces for colorblind or visually disabled people. (Easily resizable interface elements, voice over, etc.)
- Simple addition of language sets is required, since the software could be used in numerous different countries.

Reliability

- Security, privacy, protection against hacker attacks

Performance

- Often used data (maps, images, etc.) should be stored locally on the user's computer, and not sent repeatedly over the internet.
- Some data could be stored locally as well, and updated only when it changes.
 - Seldom changing data like area, natural capital, etc.

Supportability

- Occasionally the system would be expanded to handle more pollutants, or changes could be made to the geographical areas. The system must be flexible to such database changes.
- Handling changing legal circumstances should be solved (pluggable rules).
- Users should be able to provide feedback
- Support service helping users with problems should be maintained.

Interfaces

- Public interface for requests from the internet (authentication required)
- Private interface for notes
- Interfaces for external databases
- Interface for Ügyfélkapu
- Interface for online payment systems (PayPal, etc.)
- SDK for iOS, Android, Windows, and MacOSX

4.3.3. Analysis of the Requirements

In this chapter we will select the most important requirements of the system that we have collected in the two previous sections, and analyze them deeply. Sadly, we don't have the time or the energy to analyze all requirements, but our methodology also tells us that we shouldn't. The point of the RUP was to start with the most crucial requirements in order to reduce risk early. Our analysis will be divided by modules again, and in addition to the discussion about the few important features, we aim to answer a few general question about each module:

- What kind of data is needed in addition to earlier modules?
- Are there external databases to get the data from, or do we need to collect and store it?
- What are the most common transactions in that module, and how much data transfer is required for one transaction?
- Generally, how would users use the module? How often would they visit this particular module?
- Does the module/feature require an interface to other services/applications?
- Does the module require a third party software?
- How complex are the actors and the use cases in the module?

The last question important in estimating the development effort of the module. According to the methodology, we need to divide actors and use cases into categories. Actors are divided into three categories:

- Actors with a well-defined Application Programming Interface (API). These actors are designed to receive requests, and developers can easily do so with a few lines of code.

- Protocol-Driven Actors (PD). These actors can communicate through protocols, which means the developers will have to implement that protocol in our system.
- Actors that require a Graphical User Interface (GUI). These actors are the most difficult of the three.

Use cases also have three categories based on the number of transactions they have. Here, transaction means a query-reply pair.

- Easy (1-3 transactions)
- Average (4-6 transactions)
- Complex (7 or more transactions)

During our analysis, we will estimate that each short use case has one more transaction than we described in [Appendix 8.](#), to account for additional error handling, optional, or alternative executions, and so on. Also, a use case with complex computation requirements will be complex regardless of the number of transactions.

Information Module

In the Information Module the most important use case is View Area Information. It is the feature that most of the users would access, it requires the most data, and its exact functioning is worth further consideration. Moreover, there are quite a few similar use cases so we are basically discussing three or four use cases in one. The most important requirement we have is that this use case must be very user friendly. We mentioned, that area information would be displayed over a map, which could be manipulated in a way similar to other map-based applications. On this map we would display sources of emissions, cities, natural capital, etc. All of these elements would be interactive.

The main piece of information available would of course be the concentration of pollution in that area. We should ask, however, how much and what kind of information we should display for the user. Sadly there is no easy answer for that. Every one of our users will be unique, with different background knowledge and capacity to understand information. But even if we suppose users with bounded rationality, we can expect them to be able to roughly decide how much information they want, given they are presented with the choice. To achieve that, we recommend using an information pyramid. At the top level of the pyramid, there would be one number representing the quality of the environment. As we go down to lower levels, the resolution of the information would grow larger and larger. At the second

level we could display the quality of different parts of the environment (air, drinking water, soil, resources, etc.), at the third level we could show the severity of the different harmful effects, while at the bottom level we could display information about individual pollutants. User could switch between pyramid levels, or *layers* with one click. At every layer we should display information that helps users understand the situation: What are the dangers or harms that affect me? Is a metric ton of CO₂ a lot or little? What are the damages caused by 0.05% increased chance of cancer? The layers could also display metrics like the Natural Capital Index (NCI), since there are databases on that as well. ([Biró 2012](#))

One possibility of displaying guidelines for these incredibly difficult questions is the use of relative scale. This means, that quantitative things, like the severity of an effect, or the concentration of a pollutant would be displayed in a bar. The completely full bar would at any time be the highest concentration or the strongest effect, and all other values would be displayed relatively to that. This means, that if we see a half-full bar of some effect, then that means that our area is halfway between the most serious and the least serious pollution. This would also provide psychological incentives to the most polluted areas to reduce their pollution, which would in effect push the entire scale down intensifying the pressure on somewhat less polluted areas. In [Appendix 4](#) we present diagrams for this use case, and an example user interface is provided in [Appendix 11](#).

In the following part we collect the answers to the questions we wished to answer with our analysis. First, we assess the complexity of actors, use cases, the amount of servers needed, user behavior, and the data transfer. We will also ask, whether we need to collect data that is not available in external databases, and whether we need third party applications, or a special support team for the module.

The most common things that users would do in this module are webpage downloads, and playing over the map. The latter would invoke very frequent image, and data requests from our servers. The data requests would often mean data from external databases. That requires a very large amount of web servers, and a large amount of database servers. However, this module would require very little storage (what little data we need can be stored on the many web servers), also no complex computation is required on our part. Our estimate for the number of servers (for a base number of 1,000,000 monthly users) can be found in the Appendix 12 for every module.

Based on the above estimation, we can also give an estimation of the main data transfer. Page downloads per user can be easily estimated by looking at data from web analytics websites ([Alexa](#)), while the average size of a website can be checked by saving a full webpage from a browser. (We tested it with relatively modern-looking newspaper websites.) Since this the basic module of this system, we estimate about 100% of monthly users visiting it. (One user is likely to visit more than just one module.)

Administration Module

In the administration module we will select three use cases for somewhat deeper analysis. These are the three use cases that the administrator will access to turn a reported emission into imission, effect, and damages. We have mentioned before that all three are difficult. For calculating imission from an emission we need propagation models and propagation data. Luckily, we have both, but we will need to think about how to implement this feature, since there are several applications already on that market that do just the same. The simplest solution would be to simply just use them. That would mean no extra costs, assuming that the government is already licensing a few of these solutions. However, in this case it would be the administrators' task to extract data from our system, put it into the third party software, calculate the results, and then import those results back into our system. This would be an unnecessarily long and useless activity, which would result in high administration costs. If we were to write our own pollution modeling software on the other hand, then development costs would be significantly higher. Fortunately, there is a middle solution: We could use third party software, but implement an automatic interface between the two applications, which would be the minimal cost solution, but this would require the consent and the cooperation of the maker of the third party software.

The second use case is Estimating Effects. This use case basically covers an automatic impact assessment, which could reduce transaction costs significantly, while improving understandability of the problems. To imagine this, let us construct a simple mathematical model of calculating the effect that one pollution has on one natural capital.

Let $|P|$ denote the magnitude of the pollution (the concentration of one imission unit), while the vector $\bar{P}$ would store all the possible effects of the pollutant, so that the i .th element of the vector $\bar{P}$ means how strong the i^{th} effect is. The magnitude of the j^{th} natural capital in the area is $|N_j|$, while the vector $\bar{N_j}$ stores the sensitivity of that natural capital to the

individual effects. The function $Mul(v1, v2)$ produces a vector by multiplying $v1$ and $v2$ element-wise. In this case we could calculate the unit effect $\bar{E}$, which would also be a vector with the following equation (1):

$$\bar{E} = |P| \sum_{j=0}^N Mul(\bar{Nj}, \bar{P}) * |Nj| \quad (1)$$

This is actually very similar to some of the manual environmental impact evaluation methods, only expressed with vectors. The problem is that we will need to collect all possible effects for all pollutants, and more importantly, the sensitivity of every natural capital type to every type of pollutant. One way to do this is to hire a few experts who could do that. There are two problems with that approach. First, if the number of pollutants and natural capital types are high, then we have an enormous amount of data to collect. If the number is lowered, then this method will be very inaccurate and unreliable. Second, if we asked the question whether the sensitivity of a forest to soil degradation is 0.89 or 0.95, then even an expert would have difficulties giving an answer.

Fortunately, computer science has a solution for these kinds of problems: Machine Learning (ML). Without going into details, a machine learning algorithm can perform a simple achievement: If given enough real world examples of questions (pollutions and natural capital present in an area) and the right answers (the actual effects), then the algorithm will adjust its parameters (the sensitivity of natural capital types), so that it could answer a maximum number of the given questions correctly. To work well in the future as well, this method requires this *training data* to be representative of future questions. So the only thing we need to do to train a ML impact assessment program is to collect a large number of previous assessments: The pollutants, magnitudes, the natural capitals present, and the estimated effects. One disadvantage is, that the more types of effects, and natural capital we have, the more training data we need to collect.

The third use case we will examine here, is Estimating Damage. Estimating the damage caused by environmental effects is very difficult. Marjainé ([Marjainé 2001](#)) describes the methods to associate a cost to certain natural resources in detail. But even if we used those we could only give good estimates once the system is in use for a while, which makes initial estimates really hard to get. Once the system is working for a little time, we can use user polls or past transactions to have an idea what value people associate with certain

environmental goods. There is a much better question, however: Should we use this function, and display this information to people? Because if we do, we can lower transaction costs by presenting better information and guiding bargaining. But at the same time, by presenting a value, we will influence people, almost certainly changing the value they would have associated with the given good. That would shift this system away from free bargaining to taxation, since the price of a good would be very much influenced by a “central authority”. The diagrams of the three use cases are presented in [Appendix 4](#).

This module would be rarely visited by users, so we do not need a large number of web servers. There would be some data storage needed on all of our users (rights, etc.), and also some use cases administrators would access, would need data from external databases.

Also we will dedicate a few servers to perform the complex computation tasks that administrators would need. We estimate that the only significant data transfer in this module would occur from page downloads. (Administrators would store much of the propagation data locally.)

Community Module

The purpose of the community module is twofold. First, it aims to decrease transaction costs by allowing people living in certain areas to negotiate as one, significantly reducing the severity of the “too many parties” problem. The second aim is to try and provide a platform for groups of people that can reduce the chance of a prisoner’s dilemma occurring. To achieve both, we need to provide an efficient way to decision making for each group, therefore we will closely examine the use case Vote/Make a Decision. But before describing the use case itself we will need to dig into decision making and group decision making a bit more deeply.

Decision making can be individual or multi participant. [\(Marrerios\)](#) Multi-participant decision making can be further divided into the unilateral case in which one person decides, and negotiated, in which more members make the decision. There are two types of negotiated decision making, one in which members of the group are not equal (organization), and one in which they are (group). This structure is illustrated by Figure 5. [\(Marrerios\)](#) In this part of the paper we will talk about group decision making.

In order to have a good understanding of what our system is supposed to do, let’s start with the individual decision making process. Fülöp [\(Fülöp\)](#) collects the steps of this process:

1. Problem definition
2. Determining Requirements
3. Establishing Goals
4. Identifying Alternatives
5. Defining Criteria
6. Selecting a decision making method
7. Evaluating against criteria
8. Validating solution against problem

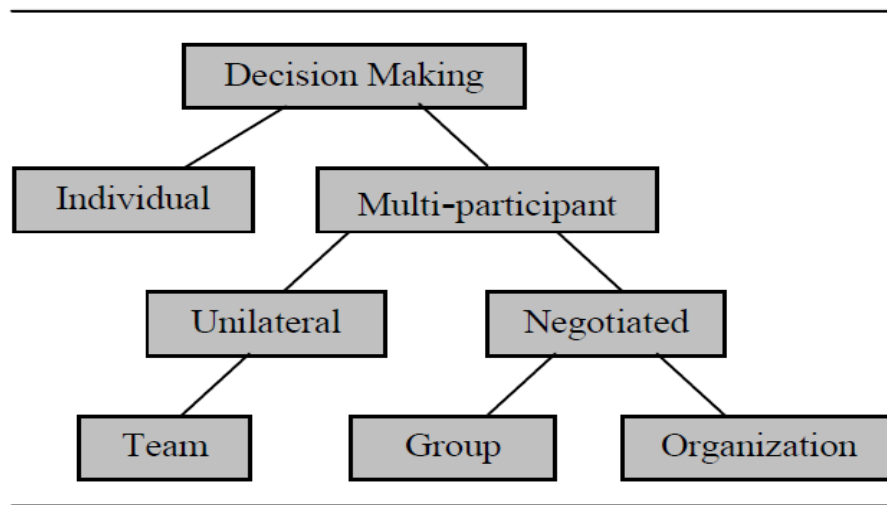


Figure 5: Decision Making Structures
Source: Marrerios

The part that is of outmost importance from our perspective is the decision making method. There are several of these methods described in (Fülöp). There are some simple heuristics-based methods, like collecting Pros and Cons of an alternative, or the maximin strategy, that tries to select the alternative, that has the best worst-case performance. There are some more complicated methods, which aren't based on heuristics. These methods include Cost-Benefit Analysis, or SMART. SMART is a very simple multi-attribute decision making method, where the score x_j of the j^{th} alternative A_j is the following:

$$x_j = \sum_{i=1}^m w_i a_{ij} / \sum_{i=1}^m w_i, \quad j = 1, \dots, n.$$

Figure 6: The Score of an Alternative with SMART
Source: Fülöp

Where a_{ij} is the performance of A_j on the i^{th} criteria, and w_i is the weight (importance) of the i^{th} criteria. In other words this is a simple weighted average of the criterion, normed by

the sum of the weights. There are also methods to find the weights based on relative importance of the criteria, for example the Analytic Hierarchy Process (AHP) ([Fülöp](#)).

While here we are talking only about individual decision making, it is important to note, that group decision making can be expressed as the averaging of individual preferences into a single group preference ([Fülöp](#)). However, even if the members of the group know their collective preference, they need a method to arrive at a decision. The simplest method to achieve that is voting. However, voting does not provide a platform for discussion. One solution to that is the Delphi technique:

“The Delphi technique is a widely used and accepted method for gathering data from respondents within their domain of expertise. The technique is designed as a group communication process which aims to achieve a convergence of opinion on a specific real-world issue.” ([Hsu 2007, page 1](#))

In a short summary, the Delphi technique asks experts of their opinion in rounds. In each round the experts will see the opinions of others, and can use those to refine their own opinion. The two major disadvantages of this technique, that is supposes that its participants are experts, and the time to reach a consensus can be really long.

Another method is Dotmocracy. ([Diceman](#)) Dotmocracy is a simple solution to prioritize ideas or solutions in a large group, using special sheets to evaluate ideas. The sheet is shown in Figure 7. The point of dotmocracy is that people in the group would post several ideas to solve a problem independently of each other. The group would then evaluate and discuss these ideas by signaling how much they agree with the solution (dotting), while also being able to voice concerns, strengths, weaknesses, etc. At the end of the dotting process a small number of the members use the results to formulate a common solution.

Write one idea here in large letters:

Dotmocracy Sheet 3.2 Creative Commons Attribution License. Copy & share freely. Download copies at www.dotmocracy.ca

 Strong Agreement	 Agreement	 Neutral	 Disagreement	 Strong Disagreement	 Confusion
					

非

[illegible]

systems consist of three important parts. ([Druzdzel 2002](#)) The first is a database management system, which helps the user access to all the relevant information. The second is a model management system that allows the user to use and create models that will help in the decision making, while the third is an easy-to-use interface. It is very important that some DSS systems are able to generate models automatically, which means that the user does not need to be and expert in decision-making.

- **Level 1 GDSS** - provide technical features aimed at removing common communication barriers such as voting, electronic message exchange.

- **Level 2 GDSS** - provide decision modelling and group decision techniques aimed at reducing uncertainty that occur in the group's decision process.
- **Level 3 GDSS** - machine-induced group communication patterns and can include expert advice in the selecting and arranging of rules to be applied during a meeting.

Now, based on all the knowledge we have collected about group decisions, we can describe our solution. First, we need to identify the problem we want to help groups to solve: “What is the best possible deal we can strike with another party?” What are our goals? First, we need a deal that is accepted by the group, and the other party as well. We also need the deal to be fair, and to handle the type of pollution well. We also know by now, that our alternatives are different offers (contract drafts). Our system should make the initial alternatives (templates), while the criteria should also be established by our software. Users could tell the system the relative importance of the criteria, which the system would use to evaluate alternatives with the preferences of the individual, the group, and also of the other party. Our system could also suggest modifications that would improve the performance of the alternative. Also, the group decision making process would work similarly to Dotmocracy. Members could post ideas (offers), vote on them, discuss them in (voice concerns), create alternatives, or offer modifications to an offer. When a consensus is reached the group would perform a simple yes-or-no vote on it. It is important to note, that since the final decision of a group is binding to all of its members, individuals would no longer have a reason to slack. However if a group makes a decision to invest into a green technology in a nearby factory, then a local authority should collect the funds required from *all the people in the area*, not just from those who represent themselves in the system. The reason for this is that otherwise people could still slack by simply not representing themselves in the system. But in this case the decision making process needs to be completely transparent.

It has become obvious in the above discussion that the use case called Vote/Make a Decision is in fact much more than a single use case. As a result we will divide this use case into several sub-use cases, and we draw diagrams for each. [Appendix 4](#) only contains the diagrams for the main use case, but the remaining ones can be found in [Appendix 9](#). The sub-use cases are:

- View Alternatives
- Add New Alternative
- View Concerns

- Add New Concern
- Evaluate Alternatives
- Set Preferences

It is easy to see, that the community module is by far the largest and most complicated, but perhaps that follows from the complexity of the problem that it tries to solve. Here, as usual we give the result of the analysis of the entire module. An example user interface for Vote is provided in [Appendix 11](#).

The community module would be the other most frequently visited module next to Information. However it doesn't require as many web servers, since the main data transfer is page downloads. There will be some database requirement as well, like searching communities, or members of communities. Storage is not really needed, however we will need to dedicate quite a few servers to run the forums.

Rights Trading Module

We don't really have extremely important use cases in the rights trading module, since it only serves as an interface for an already existing system, the e-Exchange ([theICE](#)). Therefore we will provide the general answers immediately.

The visitors of this module would not come frequently, that means only a few web servers are needed. We need a few extra of them to the permit monitoring. A few database servers are also needed to access missing rights, and for the search function. The main part of the data transfer would come from page downloads and permit monitoring.

Contract Module

The purpose of the contract module is to try and decrease the difficulty of striking a bargain by automating most of the contracting process. It is a really ambitious goal, perhaps a bit too ambitious. However, a simple idea might make this part of the system significantly more plausible. To explore those ideas, let us visit the use case Make an Offer.

In this use case one party can make a contract, and present it to the other party, who then might make a counteroffer, while exchanging messages in the meantime. If the two sides agree to the final contract, they will conclude, and it would go into effect. This basically describes the negotiating process. While we cannot present software solutions to make the parties behave less opportunistically or to negotiate in good faith, we can make the

contracting process significantly easier, and safer. The tradeoff for that would come in the form of somewhat reduced choice.

The simple idea we mentioned earlier is the following: We could use templates to aid the contracting process. How would this work? First of all for some of the more common scenarios, there would be entire pre-made contracts, where the users would only have to fill in the case-specific details. That could easily be turned into a contract by a document automation software, like the one in [\(Contract\)](#). This software also includes a function that alerts the legal team, if the contract does not meet some pre-defined rules. Another important feature would be to allow users to build their own templates using contract elements provided by the system. These elements would be “smart”: Conditions and rules could be added to these elements, while some other elements would have parameters (the amount compensation for instance). These parameters could depend on variables that the contract element is linked to (the magnitude of a pollution for instance).

Functions like that would solve two problems: First, common people could easily create contracts, and they could create several different contracts “for the price of just one” (with parameters and conditions). The second important effect is that the legal team only has to take care of problematic contracts, whose number will be low hopefully. There is one more thing to note here, which is that by allowing contract templates, and template contract elements, we have allowed several transactions to be made with the same (or very similar) contract. By doing so, we have virtually increased the frequency of transactions, while templates also allow us to effortlessly handle idiosyncratic situations. The diagrams of this use case are in [Appendix 4](#), while an example user interface is provided in [Appendix 11](#).

This module would be rarely visited, which means we only need a few web servers or database servers. We need a few storage servers for new templates, and contracts. We dedicate a few other servers to run document automation and automatic contract checking. The main data transfer in this module comes from page downloads.

Product Information Module

In the Product Information Module there is one use case that we’d like to point out mostly because of its “cool factor” that would make it an effective advertising material. This use case is Identify Product. Identify product would require a mobile device with a camera and a client application running on it. The user would show the product to the camera, and the device

would recognize the product, showing information about it. There are two ways to achieve this: The first is barcode scanning, which is relatively easy, and could be done on the device. The disadvantage of this approach is that the barcode has to be visible on the product, which makes the feature a little bit harder to use. The other approach is to recognize the product based on visual clues. This is significantly easier to use (and also cooler), but much more difficult computationally. Luckily, there are third party services available, like [Catchoom](#) ([Catchoom](#)), which are really successful. It is worth noting, that since the recognition would not take place on the device, data transfer, and server side computation is needed.

It is also important, that this module can function like a revenue source. With green technologies the module will function like an online store, while we can charge premiums for accessing advanced advertising tools, as well as a favorable position among search results. Also, by making a platform for finding green technologies easily, we decrease the costs of finding alternatives for polluters.

This module would be frequently accessed by users in order to get information about products, which means we need a moderate number of web and database servers. Product data would also have to be stored (independently of the number of users), for which we dedicate a few storage servers. For server-side object recognition we will need a moderate amount of servers. The main data transfers in this module would be page downloads, and sending images, or image metadata to the servers for recognition.

Measurement & Monitoring Module

As with the rights trading module, we don't really have use cases that would need further discussion.

This module is hardly-ever visited by users, its most frequent visitors are motes, and just one web and database server will be enough. Assuming, that every mote sends a measurement every hour, the monthly transfers per mote is 1440. We dedicate a few storage servers to the measured data, and a few servers for computation, since area monitoring can be computationally difficult.

5. Results

In this chapter we will present the results of our cost-benefit analysis, and our risk analysis. At the end we will be able to answer several important questions: We will be able to tell the financial and social benefits of the software, as well as the major risks that we will need to assess. We will be able to tell whether we managed to solve at least some of the problems with transaction costs we mentioned in chapter two, as well as the most important problems of stakeholders (efficiency).

5.1. Cost-Benefit Analysis

In this section we will analyze the costs and benefits of the proposed system using the methodology we introduced in chapter three. In addition to that, the Appendix contains a qualitative assessment of the possible macroeconomic effects of the system. The questions we aim to answer are the following:

- What are the financial benefits of establishing this system?
- What are the social benefits of establishing this system?
- Is it possible that those effects are large enough to matter?

Before we begin to explore our results, we will need to explain some general choices. Some of the costs will depend on how large the “jurisdiction” of the system is, some others will depend on how many monthly users the system will have. This means, that we will have to make some assumptions. To solve the first problem, we will consider two cases:

- The system is created and run in Hungary
- The system is created and run in the entire EU

This means, that costs that depend on the area (data collection, and mote equipment), or the average salaries will be different in the two cases. We have good data on average salaries in Hungary, and we will assume that the average salaries in the EU can be estimated by multiplying Hungarian salaries with the ratio of average price levels. Legislation costs will also be different, since the enabling legislation would be passed by different parliaments.

We will also need to estimate the number of monthly users we will have, since this determines many costs and benefits. To do that, we will make a pessimistic, a realistic, and an optimistic estimate for both Hungary and the EU. The pessimistic Hungarian estimate is the

number of current monthly users of the e-Government portal Ügyfélkapu. In the past few months the lowest monthly number of users of magyarorszag.hu was 40000. Sadly, to access the reference for this, registration is needed on the site heatsync.com (as of this writing it's free). About half of the visitors to magyarorszag.hu go on to ugyfelkapu.magyarorszag.hu ([Alexa](https://www.alexa.com)), so we give an estimate of 20000 monthly visitors. We decided to use a realistic estimate that is five times the pessimistic, and an optimistic one that is ten times the pessimistic. In case of the EU we multiply the number of users with 50, since the population of the EU is about 50 times the population of Hungary.

In this paper we will try to give a worst-case estimate of the benefits, which means we'd rather seriously underestimate the benefits, than overestimating them. So we need to hedge ourselves against the risk of overestimation by increasing all costs by 50%, and decreasing all costs by 50%. Since development costs are much more uncertain (as explained in chapter three), the uncertainty of development and maintenance goes up from 50% to 100%.

Among our general data we will store other general information as well. We have mentioned in chapter three, that we will estimate benefits in a 10-year period. That means, we will have to choose a discount rate: this will be 5.5%. We will also store exchange rates between currencies (all costs will be in \$), assumptions of the number of working hours and days, etc., as well as the tax an employer has to pay after an employee's salary. Tables containing our data or detailed estimates can be found in Appendix 13.

5.1.1. Cost of Development & Maintenance

In order to estimate the amount of development and maintenance effort based on the method we referenced in chapter three, we need the number and difficulty of actors, and use cases, the technological factors and the experience factors. Since the experience factors depend on the actual development team, we can't do better than to try to guess an average team.

We will have to make a few other assumptions: Our estimation method doesn't account for design effort, since it assumes that the software has already been fully designed. To account for that, we estimate the design effort as a percentage of development effort. The method also needs the amount of reporting and management duties that developers have to do, which we also don't know, so we have to give another estimate. Finally, the method does

not account for the development of SDKs or the inclusion of third party software. However, we can include those in the model with a small trick without giving estimate. Since SDKs are not really more, than simple user interfaces for other platforms, we can model them like they were extra actors needing a graphical user interface. Since we said earlier that four SDKs will be needed, we will add four GUI actors to every module. We also need an interface between the System and SDKs (but only one, regardless the number of SDKs), so we add one actor with API difficulty to every module. With third party software, we can use the same trick: We add one actor with Protocol Driven difficulty for every third party software.

Since our method returns the development effort in man-hours, we looked up average salaries of software engineers, and calculated the hourly cost of employer according to the following equation (2):

$$\text{Average Hourly Cost} = \frac{\text{Average Monthly Salary} * \text{Employer Tax}}{\# \text{ of Working Days} * \# \text{ of Working Hours}} \quad (2)$$

Finally, we need estimates for how much maintenance will cost each year: In the economics of maintenance ([Jones 2006](#)) there is an estimate for the yearly maintenance costs of an average project, given as a percentage of development cost. However, it only estimates the first five years, which means we will have to estimate the last five years, to get a ten-year estimate. To do that, we will continue the trend shown in the estimate.

We can now calculate the total costs of development and maintenance. The actual tables with the estimates for development effort can be found in [Appendix 10](#). From those man-hours estimates, we calculate a final estimate the development costs, reporting & management costs, and the maintenance costs, based on equations (3-6):

$$\text{Final Man Hrs} = \text{Estim. Man Hrs}(1 + \text{Design Effort})(1 + \text{Uncertainty}) \quad (3)$$

$$\text{Development Cost} = \text{Final Man Hours} * \text{Average Hourly Cost} \quad (4)$$

$$\text{R\&M Cost} = \text{Development Cost} * \text{Percentage of Reporting} \quad (5)$$

$$\text{Mainten. Cost in Year } N = \text{Dev. Cost} * \text{Mainten. Effort in Year } N \quad (6)$$

Table 6: Estimates for the Total Development & Maintenance Costs

Module	Design & Development Man-hours	Dev. Costs	Reporting Costs	Maintenance Costs	Total
Total	44,031	\$642,958	\$257,183	\$949,666	\$1,849,807

The total maintenance cost is the discounted sum of the yearly costs. The total cost of the D&M is the sum of all costs. Table 6 shows the D&M cost breakdown by module. We get an estimate about \$2,000,000, which might seem a really large number at first, but consider two things: First, creating such a large software is really expensive, second, we have deliberately overestimated the costs by 100%. Also if you look at our estimate in man-hours, 44,000 man-hours roughly means that a team of 25 people will work on this software for 14 months (reporting, and management efforts included), which seems realistic.

We can also see, that the community module is by far the most expensive, followed by information. It is worth noting, that these are also the two of the three essential modules in our system (contract being the third).

5.1.2. Costs of Server Upkeep

In this part we will estimate the number of servers needed for the six different estimate of monthly users. We have already estimated the number of servers needed with 1,000,000 monthly users in chapter four for every module, and assuming that the number of servers scale linearly (twice the users, twice the servers) we can estimate the number of servers needed. Our methodology estimates the 10-year Total Cost of Ownership (TCO) of a server rack, so we will have to convert the number of servers into the number of rack. We assume server size of 1 Rack Units, and an average 42-unit server rack can fit around 40 servers (by volume it can obviously fit 42 servers, but the real limit is the power that the servers use), so we get the number of racks by dividing the number of servers by 40.

The estimate we found is 7 years old ([Determining... 2005](#)), so we will use the web tool introduced in the paper to make a new estimate for the 10-year TCO of the server rack. The paper also claims, that approximately 50% of the TCO is operating costs, while the remaining 50% are capital costs. We assume that the operating costs divide evenly in the 10 years, while the capital costs are all immediate, and we discount the half of the TCO (the paper doesn't mention discount therefore we assume it does not discount). With these choices we can get the 10-year TCO of our server park. We estimated, that with 10,000,000 users the total number of servers needed is about 3,000. To put that into perspective, that is an above average number of servers. The websites we had server data on used at most an order of magnitude fewer servers, since they had somewhat fewer users, smaller websites and relatively little media. In comparison internet giants, like Google, Facebook or Apple have

hundreds of thousands, or millions of servers. We can also see, that most of the server capacity is required by information, community and product information modules. Note, that the number of servers doesn't scale linearly when the number of users is small, because we set a minimum number of servers for each module (usually 1).

Table 7: Total 10-Year Server Costs

Number of Users	20,000	100,000	200,000	1,000,000	5,000,000	10,000,000
Total	\$76,946	\$96,183	\$153,893	\$688,242	\$3,441,211	\$6,882,422

5.1.3. Cost of Carbon Emission

As we have mentioned in chapter three, we will use the estimate of GHG emission per data transfer estimated in [\(Taylor 2008\)](#). That estimate was made in 2006, and the technology used since then has changed considerably. Since we know how much the power use has decreased between 2000 and 2006, we can estimate the decrease between 2006 and 2013, and the future decrease between 2013 and 2022.

Now, we need to use the estimates made in the previous chapter about the monthly data transfer needed for each module to estimate the yearly data transfer we will produce, and from that we can estimate the 10-year costs by discounting the yearly costs. (Table 8)

We use this data to estimate the GHG emission in each year, by the equation (7):

$$GHG \text{ Emission in Year } N = \frac{GHG \text{ Emission in 2006} * 0.2}{10^{\frac{N}{10}}} \quad (7)$$

$$Cost \text{ of GHG in Yr. } N = \text{Yearly Data Transf.} * GHG \text{ Emiss. in Yr. } N * Price \quad (8)$$

We can see, that the cost of carbon emission is similar in magnitude to the server upkeep costs, which means that it was important to include it in the discussion. The most expensive module in this category is Information, which clearly accounts for more than 50% of the total costs. You will also notice that the carbon emission of the Measurement & Monitoring module does not scale with the number of users: That is because - as we will later see - the number of nodes (and thus the amount of data transfer) depends on the area of the system (Hungary or EU) but not on the number of users.

Table 8: Total 10-Year Emission Costs

Number of Users	20,000	100,000	200,000	1,000,000	5,000,000	10,000,000
Total	\$11,228	\$55,972	\$111,902	\$561,252	\$2,798,467	\$5,594,987

5.1.4. Cost of Data Collection

If we look over our analysis in chapter 4.3.3, we will see that there are three kinds of data missing: area data, rights data and data to run the automatic impact evaluation. One insight we make here is that the effort to collect area information and rights data depends on the size of the area, and not on the number of users. This is obvious in the case of area information, and if you consider that rights data will have to be collected for everyone (not just users), then it becomes obvious in the second case as well. This means, that there will be a Hungary and an EU estimate for area sizes of 96 and 4511 thousand square kilometers. For area information, data on administrative areas is available in other systems (local administrative systems, or INSPIRE), but we need to collect those, and divide them into much smaller areas, with the smallest one being just one property. In the case of rights data, if we have legislation, that binds protection rights to some property, or some other already existing right, then the data we need to determine protection rights is also available, so only collection is needed. In this case we will simply estimate the effort (in man-months) to determine areas and rights on an area of 100,000 km².

Collecting the data needed for automatic EIA does not depend on the number of users or on the size of the area our system will work in. However, it depends on which of the methods described in chapter 4.3.3 we choose: In the expert case, we need experts collecting data that is hard to determine for human beings. In the Machine Learning case we need administrative workers to collect training data, but we need engineers to implement the machine learning algorithm. Here, we will also give an estimate of the efforts in man-months.

It is easy to see, that the effort to make the automatic EIA is very small compared to other parts of the costs. (Table 9) Though it matters little, in the future we will assume that the AI-based solution is used, since it appears to be less expensive. The costs are calculated with equations (9-11).

$$Area\&Rights\ Data\ Costs = \frac{Effort * Area * Monthly\ Costs}{100,000} \quad (9)$$

$$EIA\ Costs\ (Expert) = Pollution\ Effect\ Collection * Expert\ Cost \quad (10)$$

$$EIA\ Costs\ (ML) = Effort\ of\ ML * Eng.\ Costs + Effort\ of\ Coll.* Admin\ Costs \quad (11)$$

Table 9: Data collection Costs

Title	Value (HUN)	Value (EU)
Area (1000 km ²)	96	4,511
Cost of Area Information Collection	\$9,869	\$724,579
Cost of Rights Data Collection	\$14,803	\$1,086,869
Cost of expert-based solution	\$48,937	\$76,464
Cost of ML-based solution	\$30,840	\$48,188

5.1.5. Cost of Legal Support

We have found during our analysis, that a legal support team will be needed for the system to run smoothly. To estimate the costs of the support team, we will estimate the number of lawyers needed. We can easily turn that into a ten-years total cost by calculating the yearly salary expenses, and discounting it.

To estimate the number of lawyers, we will have to realize that lawyers are asked to provide legal support when a transaction doesn't go well, which means that the number of legal support requests depends on the number of transactions. The monthly number of transactions will be estimated in a later chapter, so for now let's suppose that we know it. To arrive at a number, we estimate the percentage of transactions that will require legal help, and we estimate how many requests a lawyer can satisfy in a year. From this we can easily calculate the number of lawyers needed (Equation 12), and from that we have the yearly cost of the legal support team (Table 10).

Table 10: Total 10-Year Cost of Legal Support

# of Yearly Requests	48	240	480	2,400	12,000	24,000
# of Lawyers Needed	1	1	1	5	25	50
Language Correction	1	1	1	10	25	50
10-Year Total Cost	\$271,204	\$271,204	\$271,204	\$4,237,559	\$10,593,898	\$21,187,795

There is one more important consideration: In the case of the EU-wide system, we also estimate that a minimum number of 10 lawyers will be needed no matter what, since it is essential that legal support is available in all languages of the EU (and assuming a lawyer speaks about 2-3 languages, 10 will be enough). It is easy to see, that with 10,000,000

monthly users, and 24,000 yearly legal support requests the ten-year cost of the legal support is gigantic relative to costs we have encountered so far.

$$\text{Number of Lawyers} = \frac{\text{Yearly Number of Trans.} * \% \text{ of requests}}{\text{Number of requests a lawyer can satisfy}} \quad (12)$$

5.1.6. Cost of Administration

The next element we will have to estimate is the cost of administrators. This case is very similar to legal support, since, the number of administrators needed depends on the number of transactions. The reason for that is that most features that administrators use, have something to do with a transaction. The most important of these use cases is the Calculate Emission, Estimate Effect, Estimate Damage trio. Based on the descriptions of these use cases, we estimate that an administrator will spend five minutes to complete one of these. This means that in an hour, an administrator can deal with 12 new emission reports (which will likely result in 12 transactions). Since we know the number of yearly transactions, we can calculate the number of administrators needed. From that we can calculate the ten-year total costs of administration the same way as we did with the legal support team. (Table 11)

Table 11: Total 10-Year Administration Costs

# of Admins	1	1	2	10	48	95
Total 10-Year Cost	\$116,230	\$116,230	\$232,460	\$1,816,097	\$8,717,264	\$17,252,919

5.1.7. Cost of Legislation

We have mentioned in chapter three, that the methodology we referenced ([Wilson 2012](#)) is a long-term estimation of the costs of passing more laws, and that the short-term costs of passing just one more law are likely to be zero. Also, it is not likely that the system would need a large legislation, and is very likely to work without any. However, since we are trying to make a worst-case estimate, we will include this in the total costs. The cost of passing one law is calculated by equation (13) from the data shown in the Appendix 13.

We first estimate the cost of passing one law in the Hungarian Parliament. From the 2012 budget we collect the costs of running the Országgyűlés (Parliament), but exclude those costs that are clearly not the costs of making a law (the Parliament Budget includes things like support of state media, financial regulators (PSZÁF), etc.). However, the budget does not have an entry that specifies the amount of money spent on policy advice, so that will be

excluded from our estimation. The third important data, namely the number of sitting days, and the average number of yearly laws passed are available on the parliament's website.

When estimating the cost of passing a law in the EU, we similarly looked at the budget. The EU budget has an entry that includes the cost of running the parliament, including policy advice, but excluding clearly non-lawmaking costs like those in the Hungarian budget. When we counted the number of days spent on lawmaking, we included not only sitting days, but also committees and groups, since these are also attributable to lawmaking. We determined the number of texts adopted in a year by using the parliament's own search engine. We can see, that in both cases, the costs are relatively small compared to other factors, however, these costs don't depend on the number of users, which means, that with only a few users, these costs will dominate administration and support costs.

$$\text{Law Cost} = \frac{\text{Yearly Expenditures} * \text{Sitting Days \%}}{\text{Average Number of Laws}} \quad (13)$$

Table 12: Estimates for Legislation Costs

Title	EU	Source	HUN	Source2
Estimated Cost of a Law:	\$1,298,274	Calculation	\$507,165	Calculation

5.1.8. Cost of Mote Equipment

We have stated in chapter three, that when estimating the number of motes we will ignore two factors: First, we only estimate the cost of the devices, and not the costs of upkeep (power, replacements, etc.). Second, we will also assume that no measurement devices will be bought, and that devices that are already owned by the government will be installed on the motes. Both of these assumptions are made because there is no good way to estimate those factors.

To give a cost estimate, we will estimate the number of mote devices needed to cover a 100,000 km² area, and we will use real product data to estimate the cost of one mote. Of course the number of motes will be given for two area sizes (as with data collection). It is worth noting, that only 500 motes per a Hungary-sized area might seem very few, but the strength of these devices is that they are mobile. Simply by putting one mote onto a vehicle one can cover a huge area.

Table 13: Estimates for Mote Equipment Costs

Number of motes	480	22,555
Total Cost	\$49,344	\$3,214,088

5.1.9. Ad Revenue

To estimate the ad revenue from our website, we will use a website estimator that is based on statistical data. This estimator is called Ad Revenue Model for Small Websites ([Real Revenue ...](#)), and it is questionable, whether a website with 10,000,000 monthly users can be considered small. However, for now we will assume that the answer is yes.

The estimator needs the following data: The number of users, the content of the website (that is static content for the lack of a better alternative), the audience type, which is general public, the audience's willingness to spend (here we choose the second type: Wallets – Common Interest/Might Spend, since most of our users are interested in environmental protection), and the source of revenue. The source of revenue is “Ads and other programs”, where other programs means offering premium advertising possibilities for companies with products.

There is, however, a second source of revenue: Since we have green technologies in the product module, we can technically work like an online store, for our business users, who want to search and buy green technologies as a part of the deals they make. To estimate the second source, we estimate the number of our business users to be 10% of all users. At first, this might seem too high, but once we consider that businesses might be required by law to use the system, it's no longer outlandish. Businesses are also Buyers, not wallets, the website content changes to online store, the audience to corporate buyers, and the source of revenue is product sales.

Now, we have estimated two kinds of monthly revenue (shown in Appendix 13), and all monthly revenue is the sum of the two. From this we can calculate yearly revenue, which we can discount and turn into a ten-year ad revenue projection. It is worth noting, that as the number of users increases ad revenue starts to get gigantic compared to the costs we collected so far. (Table 14)

Table 14: Total 10-Year Ad Revenue

Number of Users	20,000	100,000	200,000	1,000,000	5,000,000	10,000,000
Monthly Revenue	\$5,100	\$25,500	\$51,000	\$255,000	\$1,275,000	\$2,550,000
Total 10-Year Revenue	\$307,535	\$1,537,676	\$3,075,351	\$15,376,757	\$76,883,783	\$153,767,567

5.1.10. Deal-Making Costs

The purpose of this part is to estimate the social benefits resulting from creating the system. As we mentioned in chapter three, we will estimate the difference of transaction costs before and after creating the system, and multiply it with the number of transactions in a year. Lower transaction costs will mean more transactions completed, however we will calculate with a fixed number of transactions to avoid complications.

But first, we need to give an estimate of the number of yearly transactions. It is obvious, that the number of transactions depends on the number of users. There might be, however, a scale factor, which means that twice the users wouldn't result in twice the transactions. On the other hand, introducing such factor would be incredibly arbitrary, therefore we will simply assume that one in every 50 monthly users would result in a transaction.

We have also mentioned before that estimating these costs is really difficult, since it is difficult to define what an average transaction is. To overcome that problem, we will break down deal-making costs into subcategories and estimate the effort for each subcategory before and after the system. Note, however, that all of these numbers will be based on estimates, and could largely reflect the author's optimism about how well the system would work. To hedge ourselves against that risk, we will apply the increased uncertainty (100%) to these estimates twice: We will multiply the deal-making effort after the system by two, and divide the effort before the system by two as well. This way we can make sure that the chance that we overestimated the system's benefits is very small. These estimates can be found in Appendix 13.

There are two sub categories of the deal-making costs that we will calculate based on some other estimates. The first is the lawyer costs of making a contract: The cost of making a contract after the system is calculated from the data we collected in the legal support part:

$$\text{Man hours} = \frac{\text{Monthly \# of requests a Lawyer can do} * \text{Chance of request}}{\text{Working hours a day} * \text{Working Days a Month}} \quad (13)$$

This equation will give the man-hours a lawyer will spend with one transaction. The before value is calculated similarly, except that the number of contracts a lawyer can do in a month is set to 10, and the chance that a lawyer is needed is set to 100%. The other cost we calculate are litigation costs. We use lawyer pricings [\(Pricing Example\)](#) to estimate the costs of litigation by the following equation:

$$\text{Litig. Man Hours} = \frac{\text{Avg. Value of S.M.} * \text{Lawyer Cost} * \text{Litig. Chance}}{\text{Cost of one Man Hour}} \quad (14)$$

The average value of the subject matter, and the chance of litigation are estimated, while the lawyer cost is real-world data and is given in a percentage of the subject matter. We estimate the costs of one man-hour by multiplying it with the average salary in Hungary (this is also adjusted by the price level in the EU case). The yearly social benefits are calculated based on equation (15):

$$\text{Yearly Benefits} = \# \text{ of Transactions} * (\text{Before} - \text{After}) * \text{Hourly Cost} \quad (15)$$

It is worth noting, that the social benefits seem to be 1-3 orders of magnitude larger than any other costs or benefits we encountered so far. This is relatively comforting, since it means that the social benefits are going to be clearly positive even in such a pessimistic estimate. It is also important to note, that the actual deal-making efforts before and after the system are 116 and 14.46 man-hours (before adjusting), which is an enormous 85% decrease. This does not mean, however, that transaction costs will decrease by 85%. It only means, that the parts of transaction costs, that the system affects will decrease by 85%. (Or by 50%, if we adjust them for uncertainty.)

Table 15: Total 10-Year Deal-Making Benefits (thousands of \$s)

# of Yearly Transactions:	4,800	24,000	48,000	240,000	1,200,000	2,400,000
Total 10-Year Benefits:	\$6,119	\$30,595	\$61,190	\$478,053	\$2,390,269	\$4,780,539

5.1.11. Total Costs and Benefits

In this short chapter we will present our final calculations for the total benefits of the system. We will present seven items in this chapter: Table 16 includes all costs and benefits, Figures 8 and 9 show the breakdown of costs in the Hungary and the EU cases (we use the realistic estimates).

We can see, that the 10-year social benefits are clearly positive in all six cases. However, when we switch to the EU-wide system we will see that the social benefits skyrocket. The reason for this is of course the very large number of potential users in the EU. We can also see, that the financial benefits are quite likely to be positive in the EU, while uncertain in Hungary for similar reasons. Another interesting comparison is the cost breakdown. Our estimates tell that the most important cost in the Hungary case is Development and

Maintenance, followed by the Cost of Legislation, while in the EU case Legal Support and Administration costs take the lead, since these are the ones that grow dramatically with an increased number of users.

Table 16: Breakdown of Total Estimated Costs and Benefits (Thousands of \$)

Costs						
Number of Users	20,000	100,000	200,000	1,000,000	5,000,000	10,000,000
Total Costs	\$2,937	\$3,001	\$3,231	\$16,565	\$34,813	\$60,180
Benefits						
Ad Revenues	\$307	\$1,537	\$3,075	\$15,376	\$76,883	\$153,767
Deal-Making Benefits	\$6,119	\$30,595	\$61,190	\$478,05	\$2,390,269	\$4,780,539
Total Benefits	\$6,426	\$32,133	\$64,266	\$493,430	\$2,467,153	\$4,934,306
Net Benefits						
Total Social Benefits	\$3,489	\$29,131	\$61,034	\$476,865	\$2,432,340	\$4,874,126
Total Financial Benefits	-\$2,629	-\$1,463	-\$155	-\$1,188	\$42,070	\$93,587
TFB without Uncertainty	-\$1,188	\$613	\$2,767	\$12,503	\$92,598	\$191,012

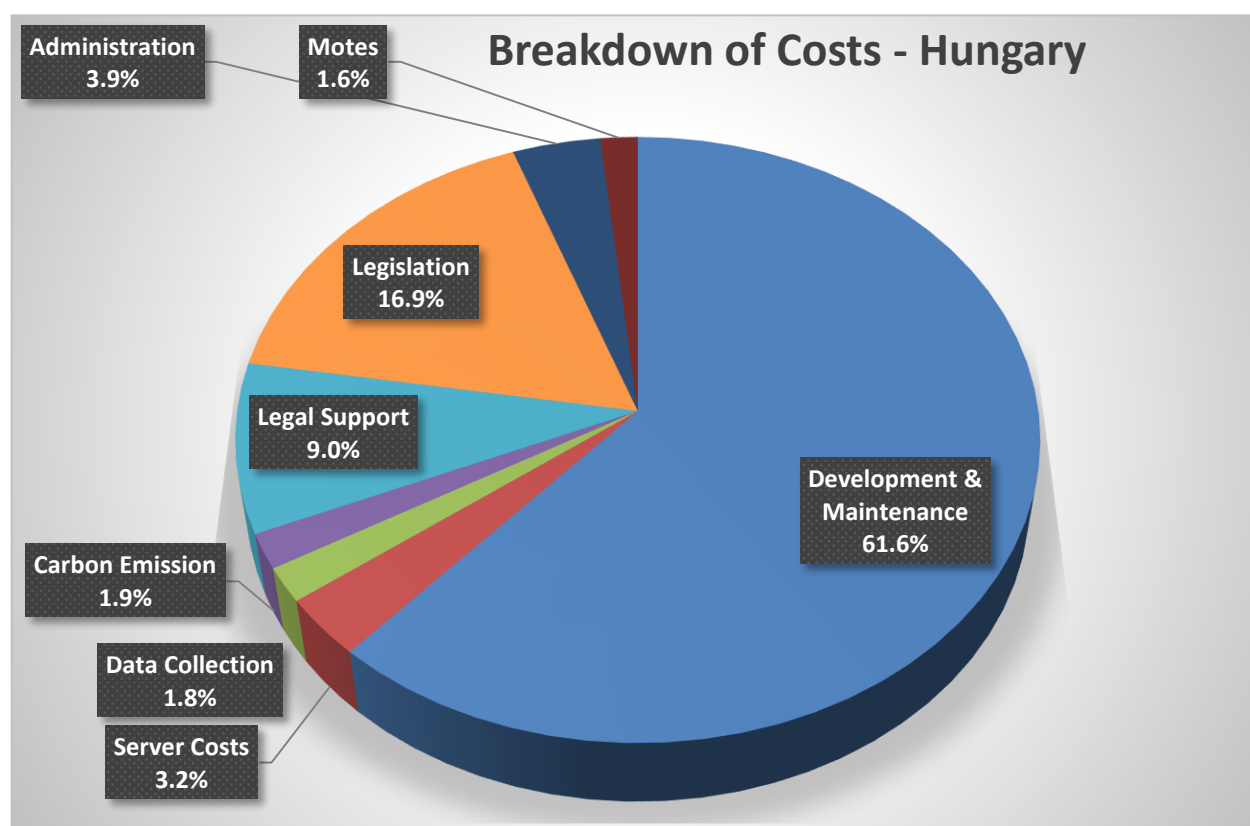


Figure 8: Breakdown of Costs – Hungary

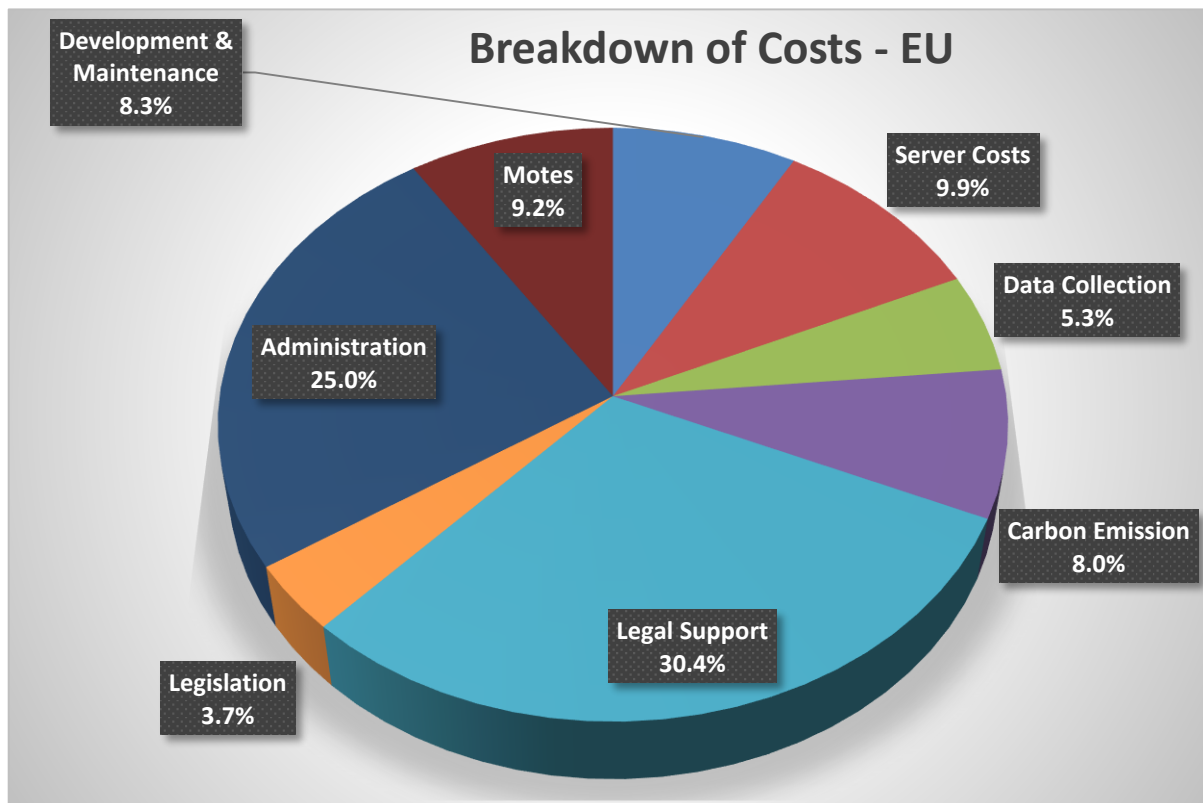


Figure 9: Breakdown of Costs – EU

5.1. Risk Analysis

In this chapter we will collect several types of risks that might affect our project, and analyze them shortly. If we found serious risks we will also try to offer solutions to reduce them. The first type is technological risks: The risk here is that there is no technology to satisfy one or more requirements. There are a few features in our system that could prove to be impossible with current technology, like the ML-based automatic EIA, automatic imission calculation or automatic contracting.

Luckily, there are examples of similar systems working and while these would require some innovation, we assign a **very small** probability to this risk. However, the consequences of this risk would be **serious**, since it affects quite a few of the core elements of the software.

The next type is People risks: It might be hard to find engineers with special skills, luckily however, we don't really need many of them. The probability of this happening is **moderate**, while the consequences are **tolerable**, since this would only make the development a tiny bit more expensive. The other People risk is that it might be hard to find skilled workers at all. The probability of that happening is **small**, while the consequences are **tolerable** for the same reason we had previously.

The third risk type is Organizational: The important risk we have here is legislative barriers: One of the system's features might be blocked by some law. The probability of occurrence is **small**, since we suppose that the government is making the system. If it still occurs, however, the consequences would be **serious**, since it could affect important features.

The fourth type is Requirements risk. We know since chapter three, that requirements **will** change. The probability of this risk is **large**. However, we have chosen a software design methodology that handles changing requirements very well. At this point additional requirements could easily be added. They would modify costs slightly, but pose no other danger, therefore the consequence is **tolerable**.

The last type is Estimation risk, namely the underestimation of the effort and costs. Since we are in an early phase of a software design process, the probability of this is still **very large**. Moreover, the consequences are **serious**, since we are not talking about a 5% or 15% error, we are talking about 50% or 100% errors. Luckily, we already have offered a solution to reduce the expected value of this risk by consistently overestimating costs, and underestimating benefits (this is precisely why we did that). With this we can safely say, that we reduced the probability to **moderate** and the consequence to **tolerable**. It is now visible on figure 10, that there are no extreme risks for the ones with large consequences have low probability, and vice versa. The collection of risks is displayed in table 17.

Table 17: The Risks of the Project

Risk	Probability	Consequence
No technology available to satisfy requirements	Very Small	Serious
Legislative barriers	Small	Serious
Hard to find engineers with special skills	Moderate	Tolerable
Hard to find highly skilled workers	Small	Tolerable
Requirements will change	Large	Tolerable
Underestimation of Effort and Costs	Very Large	Serious
Underestimation with Uncertainty in Estimates	Moderate	Tolerable

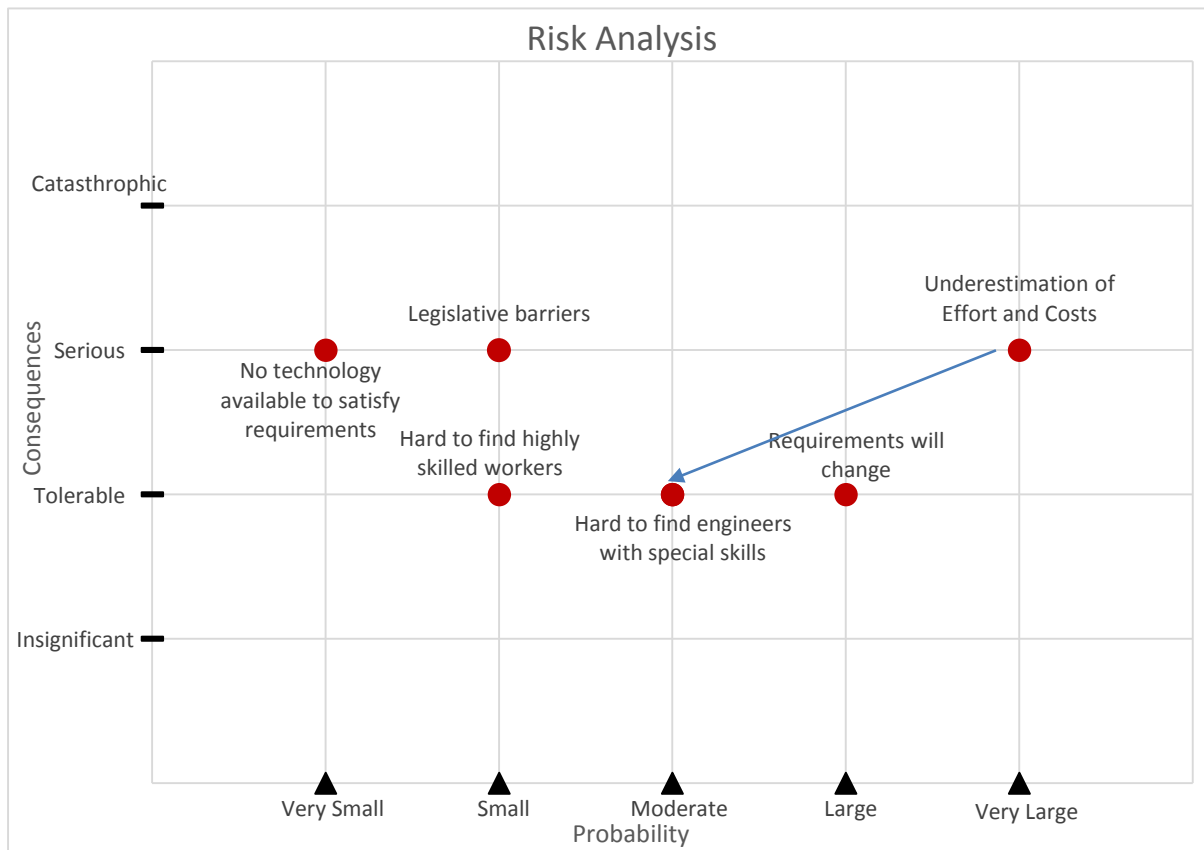


Figure 10: Risk Analysis

6. Conclusion, Recommendations

Throughout this paper we have discussed the Coase-theorem, the importance of transaction costs, and the variables that make transaction costs high. We also briefly reviewed the currently operating software systems that help environmental protection. We then went on to design and analyze a new software system that is meant to significantly decrease the transaction costs of Coaseian bargaining by providing an exchange system for these transactions. We described its most important features in depth, and analyzed the transaction costs of establishing and maintaining the software and the infrastructure around it. Finally, we gave a short assessment of risks.

We have found in our preliminary analysis that the system is likely to be successful in reducing transaction costs of bargaining, while the community module also might offer a solution to reduce the chance of the problem of the commons occurring. We have found that even in a pessimistic case the project is likely to have positive social benefits in a ten-year period, while the sign and the magnitude of the financial benefits will largely depend on the number of users that the system will attract. We have found no significant risks that could not be easily averted.

We can see, that we have successfully provided solutions to solve the problems that we collected in chapter two. Our use of templates and contract automation can help us reduce the problems of high asset specificity and low frequency by virtually making both better, while the features of the Community Module can significantly reduce the problem of too many parties. We have included basic tools to make administration and monitoring easier, which could be investigated further. We have also included the possibility of connecting the system to emission trading systems.

Based on these results we recommend further investigation of the possibility of establishing this system. On the design side, there are certainly requirements we missed, further problems to solve. A more detailed analysis of the requirements would also allow us to make a more accurate cost estimation. Also, in the CBA part we have used estimates in quite a few places (especially the deal-making costs), which are no more than educated guesses. Our reason for using them was that giving an accurate estimate of those factors would amount for an entire paper. However, it would greatly reduce the uncertainty of our estimate.

Another subject that would require further exploring is the legal and institutional background that is needed for this system to perform well. During our transaction cost analysis we have assumed, that an average-sized law needs to be passed for the system to work, but that was nothing more than a pessimistic assumption. In chapter 2.3.1, we have briefly discussed different methods to distribute rights in the context of emission trading, and have seen that our choice of method has large impact on the effectiveness and social justice of the system, which is probably true in our case as well. However, we have not discussed this in great depth. We have also mentioned a few possibilities of using features of this software for commerce, but we have hardly detailed all the possibilities. We think these would all be great future research subjects.

References

- (Allen 1999) Douglas W. Allen (1999) [*Transaction Costs*](#),
- (Barbier, 2011) Edward B. Barbier (2011) Transaction costs and the transition to environmentally sustainable development, *Environmental Innovation and Societal Transitions* 1, 58–69
- (Bartus et. al., 2012) Bartus Gábor - Szalai Ákos (2012) Környezetgazdaságtani problémák elemzése Közpolitikai eszközök és joggazdaságtani magyarázatok, *Pázmány Law Working Papers*, 2012/42
- (Bartus, 2006) Bartus Gábor (2006) *A Hulladékgazdálkodás Alapjai*, Nemzeti Szakképzési Intézet, Budapest
- (Beckmann 2002) Volker Beckmann (2002) [*Transaction Cost and Environmental Economics: Towards a New Approach*](#),
- (Beliczay et. al., 2003) Beliczay Erzsébet - Szabó Zoltán (2003) Az Éghajlatvédelem Gazdasági Eszközei: Az Emisszió-Kereskedelem, *Levegő Munkacsoport*
- (Biró 2012) Marianna Biró, Zsolt Molnár (2012) [*Hazánk természeti állapota. Az élőhelyek természetessége, veszélyeztetettsége és a természeti tőke index*](#), Magyar Környezeti Nevelési Egyesület
- (Carroll 2005) Edward R Carroll (2005) [*Estimating Software Based on Use Case Points*](#), OOPSLA '05, 257-265
- (Coase, 1937) Ronald Coase (1937) The Nature of the Firm, *Economica*, 4, 16, 386-405
- (Daly 2007) Aaron Daly, Paolo Zannetti (2007) [*Air Pollution Modeling – An Overview*](#), In: P. Zannetti, D. Al-Ajmi, and S. Al-Rashied, (eds.), *Ambient Air Pollution*, 2, 15-28
- (DeLong-Summers 2012) J. Bradford DeLong, Lawrence H. Summers (2012) [*Fiscal Policy in a Depressed Economy*](#), Brookings Papers on Economic Activity, Spring 2012, 233-297
- (Determining... 2005) [*Determining Total Cost of Ownership for Data Center and Network Room Infrastructure*](#), APC, 2005
- (Diceman 2007) [*Dotmocracy Handbook*](#), Jason Diceman, 2007
- (Druzdzel 2002) Marek J. Druzdzel, Roger R. Flynn (2002) [*Decision Support Systems*](#), In: Allen Kent (ed.), *Encyclopedia of Library and Information Science*
- (Ebel 2009) Renate Ebel, Thorsten Schlachter, Werner Geiger, Rainer Weidemann (2009) [*Integration of Environmental Information in Federal States' Environmental Portals*](#), In: Volker Wohlgemuth - Bernd Page - Kristina Voigt (Eds.) *EnviroInfo* 2009, 2, 357-362
- (Evolutionary, 2008) [*Evolutionary Software Development*](#), RTO Technical Report, 2008
- (Fülöp) [*Introduction to Decision Making Methods*](#), János Fülöp
- (Group Decision Support...) [*Group Decision Support Systems*](#)

- (Hsu 2007) Chia-Chien Hsu, Brian A. Sandford (2007) [The Delphi Technique: Making Sense Of Consensus](#), Practical Assessment, Research & Evaluation, 12(10)
- (James 2001) I.D. James (2001) [Modeling Pollution Dispersion, The Ecosystem And Water Quality In Coastal Waters: A Review](#), Environmental Modelling & Software, 17, 4, 363-385
- (Jamous 2012) Naoum Jamous, Samir Al Badawi Dahma, Albert Isaac (2012) [Green Logistics' Tools as Corporate Environmental Management Information System \(CEMIS\)](#), In: Hans-Knud Arndt, Gerlinde Knetsch, Werner Pillmann (Eds.), *EnviroInfo* 2012, 1, 707-717
- (Jones 2006) Capers Jones (2006) [The Economics Of Software Maintenance In The Twenty First Century](#)
- (Juergensmeyer, 1967) Julian Conrad Juergensmeyer (1967) [Control Of Air Pollution Through The Assertion Of Private Rights](#), *Duke Law Journal*, 1126-1155
- (King 2007) Andrew King (2007) [Cooperation Between Corporations And Environmental Groups: A Transaction Cost Perspective](#), *Academy of Management Review*, 32, 3, 889
- (Konstantinidis 2009) Stefanie Konstantinidis, Fred Kruse, Martin Klenke (2009) [Current state of the German Environmental Information Portal PortalU®](#), In: Volker Wohlgemuth - Bernd Page - Kristina Voigt (Eds.) *EnviroInfo* 2009, 1, 363-368
- (Kruse 2009) Fred Kruse, Stefanie Konstantinidis, Martin Klenke (2009) [PortalU®, a Tool for Building a Single Information Space in Europe \(SISE\) for the Environment](#), In: Volker Wohlgemuth - Bernd Page - Kristina Voigt (Eds.) *EnviroInfo* 2009, 1, 357-362
- (Marjainé 2001) Marjainé Szerényi Zsuzsanna (2001) [A természeti erőforrások pénzübeli értékelése](#), *Közgazdasági Szemle*, 48, 114-129
- (Marrerios, 2006) Goreti Marreiros, Paulo Novais, José Machado, Carlos Ramos, José Neves (2006) [A Formal Approach To Argumentation In Group Decision Scenarios](#), In: Ramos C. Vale Z. Faria L. (Eds.), *Knowledge And Decision Technologies*, Porto: IPP, ISEP, 2006
- (McCann 2005) Laura McCann, Bonnie Colby, K. William Easter, Alexander Kasterine, K.V. Kuperan (2005) [Transaction Cost Measurement For Evaluating Environmental Policies](#), *Ecological Economics*, 52, 4, 527-542
- (Pillmann 2009) Werner Pillmann, Jiří Hřebíček, 2009, [Information Sources for a European Integrated Environmental Information Space](#), In: Volker Wohlgemuth - Bernd Page - Kristina Voigt (Eds.) *EnviroInfo* 2009, 2, 335-340
- (Rational 1998) [Rational Unified Process: Best Practices for Software Development Teams](#), IBM, 1998
- (Riecken 1995) Sarah Riecken (1995) [A Compendium of Water Quality Models](#), Government of British Columbia, 1995
- (Schabauer 2012) Johannes Schabauer, Gerald Schimak, Gerhard Dünnebeil, Martin Litzemberger (2012) [openUwedat – A Toolbox Solution for Integrated Air Quality and Traffic Monitoring](#), In: Hans-Knud Arndt, Gerlinde Knetsch, Werner Pillmann (Eds.), *EnviroInfo* 2012, 1, 695-706

- (Stafford & Hartman 2001) Stafford E., Hartmann (2001) Greenpeace's greenfreeze campaign, *Eco-Efficiency in Industry and Science*, 6, 107-131
- (Steuer 2010) Horst Steuer, Martin Kunert, Matthias Schulz, Prof. Matthäus Schilcher, (2010) [*Standards Based And Automated Processing Of Forest Spatial Indicators In The Context Of GMES And INSPIRE*](#), In: Klaus Greve, Armin B. Cremers (Eds.), *EnviroInfo 2010*, 1, 17-22
- (Szlávik 2005) Szlávik János DSc. (2005) [*Fenntartható fejlődés vagy növekedés?*](#), In: Dombi Ákos (Ed.), *Gazdasági növekedés Magyarországon*, Műegyetemi Kiadó, 2005
- (Taylor 2008) Cody Taylor, Jonathan Koomey (2008) [*Estimating Energy Use and Greenhouse Gas Emissions of Internet Advertising*](#)
- (Williamson 1981) Oliver E. Williamson (1981) [*Economics of Organization: The Transaction Cost Approach*](#), *The American Journal of Sociology*, 87, 3, 548-577
- (Williamson 1975) Oliver E. Williamson (1975) *Markets and Hierarchies. Analysis and Antitrust Implications*, Free Press, 1975, referenced by: (Beckmann 2002)
- (Williamson 1985) Oliver E. Williamson (1985) *The Economic Institutions of Capitalism*, Free Press, 1985, referenced by: (Beckmann 2002)
- (Williamson 1991) Oliver E. Williamson (1991) *Comparative Economic Organization: The Analysis of Discrete Structural Alternatives*, *Administrative Science Quarterly*, 36, 2, 269-296, referenced by: (Beckmann 2002)

Web References

- (Alexa) [Alexa: http://www.alexa.com/](http://www.alexa.com/)
- (bBlue ...) [bblue: http://www.bblue.cc/](http://www.bblue.cc/)
- (Brad DeLong, 2013) [*The Economist As...?: Draft Talk: Notre Dame Public Intellectualism Conference* http://delong.typepad.com/sdj/2013/04/20130423-the-economist-as-presentation-script-notre-dame-public-intellectualism-conference.html](http://delong.typepad.com/sdj/2013/04/20130423-the-economist-as-presentation-script-notre-dame-public-intellectualism-conference.html)
- (Benyó, 2013) [*Software Design, Balázs Benyó:* https://www.iit.bme.hu/~bbenyo/SE/SwEng_05_02_v01.pdf](https://www.iit.bme.hu/~bbenyo/SE/SwEng_05_02_v01.pdf)
- (Catchoom) [Catchoom Image Recognition: http://catchoom.com/](http://catchoom.com/)
- (CEIP) [CEIP http://www.ceip.at/unece-clrtap-emep-tfeip/](http://www.ceip.at/unece-clrtap-emep-tfeip/)
- (Contract) [Business Integrity Document Automation: http://www.business-integrity.com/](http://www.business-integrity.com/)
- (Database ...) [*Household Products Database:* http://householdproducts.nlm.nih.gov/index.htm](http://householdproducts.nlm.nih.gov/index.htm)
- (Environmental ...) [*Database of Environmental Information for Products and Services:* http://yosemite1.epa.gov/oppt/eppstand2.nsf](http://yosemite1.epa.gov/oppt/eppstand2.nsf)
- (EPA-Air) [EPA-Air: http://www.epa.gov/ttn/scram/dispersion_prefrec.htm](http://www.epa.gov/ttn/scram/dispersion_prefrec.htm)
- (EPA-Water) [EPA-Water: http://water.epa.gov/scitech/datait/models/index.cfm](http://water.epa.gov/scitech/datait/models/index.cfm)

(GMES) [Global Monitoring for Environment and Security](http://gmesdata.esa.int/web/gsc/home)
<http://gmesdata.esa.int/web/gsc/home>

(INSPIRE) [Inspire directive](http://inspire.jrc.ec.europa.eu/) <http://inspire.jrc.ec.europa.eu/>

(OKIR) [Országos Környezetvédelmi Információs Rendszer](http://okir.kvvm.hu/) <http://okir.kvvm.hu/>

(PortalU) [PortalU](http://www.portalu.de/) <http://www.portalu.de/>

(Pricing Example) <http://www.drdudas.hu/dijak>

(Real Revenue ...) [Real Revenue Model Statistics for Small Websites:](http://www.marginhound.com/revenue-model-study-for-small-websites/)
<http://www.marginhound.com/revenue-model-study-for-small-websites/>

(Standish Group 2010) [Standish Group Chaos Reports Revisited:](http://theagileexecutive.com/2010/01/11/standish-group-chaos-reports-revisited/)
<http://theagileexecutive.com/2010/01/11/standish-group-chaos-reports-revisited/>

(Structural reform ...) [Structural Reform of the European carbon market:](http://ec.europa.eu/clima/policies/ets/reform/index_en.htm)
http://ec.europa.eu/clima/policies/ets/reform/index_en.htm

(TEIR) [Területi Információs Rendszer](http://www.teir.hu/) <http://www.teir.hu/>

(theICE) [ICE Emission Data system](https://www.theice.com/emissions.jhtml) <https://www.theice.com/emissions.jhtml>

(Török 2010) [A természeti tőke csökkenése világszerte - a csökkenés okai és következményei](#),
Török Katalin, 2010

(Vibro) [Vibrocomp:](http://www.vibrocomp.hu/hun/soundplan.html) <http://www.vibrocomp.hu/hun/soundplan.html>

(What the web's...) [What the web's most popular websites are running on:](http://royal.pingdom.com/2007/02/26/what-the-webs-most-popular-sites-are-running-on/)
<http://royal.pingdom.com/2007/02/26/what-the-webs-most-popular-sites-are-running-on/>

(Wilson 2012) [Estimating the cost of new public health legislation](#), Nick Wilson, Nhung Nghiem, Rachel Foster, Linda Cobiac, Tony Blakely, 2012

(Zipp ...) [Zipp: Megállapodást kötött a Sziget és Újpest önkormányzata:](http://zipp.hu/belfold/2011/04/07/megallapodast_kotott_a_sziget_kft_es_ujpest_onkormanyzata)
http://zipp.hu/belfold/2011/04/07/megallapodast_kotott_a_sziget_kft_es_ujpest_onkormanyzata