Enterprise Application Modularity

New features related to visualization and measurement of modularity within the Enterprise Architecture Analysis Tool (EAAT)

Alexandre Sirko

Degree project in
ICS
Masterthesis,
Stockholm, Sweden 2013

XR-EE-ICS 2013:012
Abstract

During my studies at the Ecole des Mines d'Albi-Carmaux, I did an internship at the Royal Institute of Technology (Kungliga Tekniska Högskolan) in Sweden to finalize my master.

The subject of this exercise is positioned as the junction of two different research areas. There is on one hand the development of new analysis method of Enterprise Architecture (EA) and on the other the analysis of software architecture modularity.

The entire mission concerns a method of analyzing the modularity of EA developed by Harvard Business School. The analysis method called “Hidden Structure” aims to evaluate the components of software and isolate vulnerable or key components.

This internship's main task is to develop the Enterprise Application Modularity module. It is a module dedicated to the modularity analysis according to the hidden structure method and its integration with the Enterprise Architecture Analysis Tool (EAAT) software developed by KTH. This project also needs to develop additional functionalities to support the calculations. Among these features, you can find support for instantiation of new models and automated generation of reports containing the results.
Résumé

Au cours de mon cursus à l’école des Mines d’Albi, j’ai effectué un stage de fin d’étude à l’université de KTH (Kungliga Tekniska Högskolan - Institut royal de technologie) en Suède.

Le sujet de cet exercice se positionne à la jonction de deux domaines de recherches différents. Il y a d’un côté le développement de nouvelles techniques d’analyse de l’urbanisation des systèmes d’information (Enterprise Architecture) et de l’autre l’analyse de la modularité des architectures logiciels.

L’ensemble de la mission s’articule autour d’une méthode d’analyse de la modularité des architectures logicielles développées à Harvard Business School. Cette méthode (la méthode d’analyse par la “Hidden Structure”) a pour but d’évaluer les composants d’un logiciel et d’en isoler les composants clés ou vulnérables.

Ce stage a pour principales missions de développer un module d’analyse de la modularité par la méthode de la “Hidden Structure” et son intégration au sein de la suite logiciel EAAT (Enterprise Architecture Analysis Tool) développé à KTH. Il a également fallu développer d’autres fonctionnalités pour accompagner et supporter le module d’analyse. Parmi ces fonctionnalités, on peut compter l’aide à l’instanciation de nouveaux modèles ainsi que la génération automatisée de rapports contenant les résultats.
Summary

Abstract ................................................................................................................................. i
Résumé ................................................................................................................................. ii
Summary ............................................................................................................................... iii
Introduction ......................................................................................................................... 1
Outline ................................................................................................................................. 3
I. Motivation and Objectives ............................................................................................. 4
   A. Enterprise Architecture (EA) .................................................................................... 4
   B. The study of software modularity ........................................................................... 6
   C. Why adding the Enterprise Application Modularity within EAAT ....................... 7
II. Definition and context ................................................................................................. 9
   A. Main definitions ....................................................................................................... 9
   B. General context ....................................................................................................... 10
III. The hidden structure method ................................................................................... 15
   A. The Design Structure Matrix (DSM) .................................................................... 15
   B. First metrics .......................................................................................................... 16
   C. What is the Core ..................................................................................................... 17
   D. Different types of architecture .............................................................................. 17
   E. The exposed hidden structure .............................................................................. 18
   F. An example .............................................................................................................. 20
IV. Methodology ............................................................................................................... 24
   A. General organization .............................................................................................. 24
   B. Getting started with the project ............................................................................. 25
   C. Planning ................................................................................................................. 26
V. Implementation .............................................................................................................. 29
   A. Data structure ........................................................................................................... 29
   B. File reading ............................................................................................................. 30
   C. Calculation prototype ............................................................................................ 32
   D. PDF generator ........................................................................................................ 36

Alexandre Sirko 2013
VI. Integration and Tests between hidden structure method and EAAT architecture 39
   A. Integration and adaptation to the EAAT structure ........................................ 39
   B. Test samples ........................................................................................................ 47

VII. Discussion and conclusion .................................................................................. 50
   A. Summary of results ............................................................................................... 50
   B. Main ambiguities .................................................................................................. 50
   C. Possible improvement ......................................................................................... 53
   D. Future work .......................................................................................................... 54
   E. Conclusion ............................................................................................................ 54

Bibliography ................................................................................................................. A
Table of illustrations ...................................................................................................... E
Table of tables .............................................................................................................. F
Appendices .................................................................................................................. G
Introduction

Today, companies and their information systems has become increasingly complex [1]. This complexity is costly and this cost is significant. Improving the management of information systems has become very important. That is why today it is vital to understand and analyze the EA. Paying attention to the EA can reduce cost and increase efficiency in companies.

Companies are often focused on improving profitability. One readable manner to increase profitability is to reduce costs. In addition, the information system of a company is becoming more expensive as it is very difficult to maintain and improve. This is why EA has become a topic of active research [2]. EA is largely a model-driven discipline. Models are used to document and model EA’s. But there are still few tools to do analysis.

EAAT is a solution proposed by KTH. EAAT is a complete tool. It starts with the definition of a meta-model, then instantiate a model to represent a real-world scenario and finally, it is possible to carry out calculations to analyze the model. It is a complete tool because it allows the user to define the structuring rules of an EA, documenting instances of these rules and the last but not least to analyze the system.

Today information systems have completely merged with companies. Each transformation of the company involves impact on the information system. In addition, one of my teacher (Didier Gourc from the Ecole des Mines d'Albi) told us that 70% of IT projects end up later than expected.

It is therefore very important to assess the modifiability of the EA. The modifiability is complex to evaluate [2]. It is therefore interesting to analyze other criteria involved in the modifiability of EA. For example, if an EA has a high modularity level then it is likely that its modifiability level is also high. But it is important to emphasize that there is no reciprocity between modifiability and modularity.

The analysis of modularity has already been studied in many other areas, especially in the IT field. The hidden structure method was developed at Harvard Business School. Its main purpose is to analyze the modularity of software architectures. This method focuses
on the analysis of the role of each component in terms of its links with other system components.

The purpose of this master thesis is to implement this method within EAAT, while adapting it to EAs. This project is called Enterprise Application Modularity.
Outline

This report is divided in seven sections:

1. **Motivation and Objectives:** This section provides more details on the progress of research on two topics: the first is the study of the EA and the second is the analysis of the structure of complex software. Both subjects form the theoretical basis of this master thesis.

2. **Definition and context:** The purpose of this section is to clarify some points. This information is a little more technical and bring the following sections.

3. **The hidden structure method:** This section details the main steps of the hidden structure method. This method is the basis of all implemented calculations.

4. **Methodology:** Here is shown how the project was organized. There are the organization between the different actors of this project and the planning of tasks.

5. **Implementation:** The "Implementation" section describes how the calculation prototype was realized. This prototype contains the calculation method, the basic functionality to read Excel files and the PDF generator.

6. **Integration and Tests between hidden structure method and EAAT architecture:** This section targets to present how the prototype was integrated within the EAAT software. Integration is also accompanied by sets of tests.

7. **Discussion and conclusion:** The purpose of this section is to give a quick overview of the main results and what remains to be done.
I. Motivation and Objectives

The purpose of this master thesis is to combine two subjects. The topics are: enterprise architecture and the study of modularity applied to the software industry.

This section will first present each of these areas, then how and why these two subjects are related.

A. Enterprise Architecture (EA)

Because of my studies and interest, I had a very IT-oriented vision of the Enterprise Architecture (EA). From this background, I thought that EA was dedicated to monitor and modify an information system in order to effectively and efficiently support and accompany missions of an organization and their transformations. But a more complete definition is given by [3]:

*Enterprise Architecture is an approach for managing the organization’s information system portfolio and its relation and support to the business. At the base of the approach lies an architectural model incorporating concepts such as software components, connectors, functions, information, business processes, organizational units and actors.*

The information systems of a company allow the company to realize its missions. They contain all the information involved in the life of the company. In addition, the computer system is in the process of merging with information systems. Therefore, the proper functioning of a company depends on the operation of several programs and the proper working of the whole company depends on its EA.

The management of EA is now a strategic hub for companies. When a company grow, EA is a way to reduce costs while increasing the effectiveness of its services. This can help to gain a competitive advantage [1].

Countries have realized the critical aspect of this issue. For example, the United States include a definition of EA (U.S.C. Title 44, Chap. 36, § 3601):

*“Enterprise architecture”*

(A) Means:

(i) A strategic information asset base, which defines the mission;
Motivation and Objectives

(i) The information necessary to perform the mission;  
(ii) The technologies necessary to perform the mission; and  
(iii) The transitional processes for implementing new technologies in response to changing mission needs;  

(B) Includes:  
(i) A baseline architecture;  
(ii) A target architecture; and  
(iii) A sequencing plan.

It is for these reasons that many research works have been done on this subject which have even become a discipline on its own [4]. EA is a large subject, it is interested in planning, designing, documenting and communicating IT and business related issues [5].

Today, it is clearly becoming a model-based subject. For example, it is possible to model a company from its information system point of view but it will be different from its process point of view. Each type of modeling bring a different point of view. As this field is mainly oriented on modeling, it is very important to carefully choose the meta-model. This meta-model includes all the rules of the EA model. The meta-model represents the chosen point of view.

The choice of meta-model is a critical point. Following the meta-model, resulting models will be more or less accurate. Each meta-model defines a different way of observing at the studied system. And this meta-model must be both precise (to help to create a model with many details) and generic (to be able to adapt the same theoretical basis to several different real cases) [2].

According to [2], most viewpoints are designed from a model entity point of view, rather than a stakeholder concern point of view. It is important to analyze how the system is built but do not overlook the fact that the system is never autonomous. It interacts with many stakeholders. The system should be adapted to the user (human or machine), and not vice versa. Take the vision of the stakeholder can highlight inconsistencies between the system and its users.

Originally the Industrial Information & Control Systems (ICS) mainly used the Probabilistic Relational Model (PRM) standard. PRMs contain attributes which are causally related. It is a standard that offers to create probabilistic models. They provide a sound and coherent foundation for dealing with the noise and uncertainty encountered in most real-world domains [6]. In addition, It can be used for a variety of tasks, including prediction, explanation, and decision making.
Motivation and Objectives

[2] and [7] worked on a way to measure the modifiability of EAs. Modifiability is the ability of a system to be changed. The analysis of the modifiability of a system is used to assess the costs of the development of the EA. Moreover, today, companies are changing very quickly (merger, partnership, new business, etc.). This requires that the information system is still evolving at the same rate. The analysis of modifiability is also a way to improve the design and improvement of the information system.

[7] describes how the Object Constraint Language (OCL) is more suitable than PRMs. OCL is adapted to UML for adding logical constraints on all the elements of modeling. The decisions that led to this development, the transition from the PRM standard to the OCL standard (with the UML), will not be detailed here. However, the changes implied by this choice are very important for this current project (“Enterprise Application Modularity”). Indeed, the UML standard describes the structure of all models and meta-models handled in this project. The traces of the standard PRM have not been completely erased by the passage of the UML standard (eg, the name of classes is still PrmClass).

B. The study of software modularity
This master thesis is based on studies done at Harvard Business School ([8], [9] and [10]).

In large IT projects, modularity is a key factor for the survival of the project. Indeed, the maintenance and evolution of software can be much more complex if all components of applications are tightly coupled. On the other hand, if each feature is independent of the others, then their changes become much easier (and less expensive).

[8] highlights this idea. The more software grows, the more important it is to have a modular architecture. It also illustrates this assertion with examples which come from open source ¹ project and proprietary project. The Mozilla Firefox is one of these examples. In their studies, we can clearly see that a great effort is made to make the program more modular. These efforts are made between versions date of 08/04/1998 and date of 12/11/1998, just after that Netscape’s Navigator browser was released under an open source license (March 1998). Among other things, they have halved the number of dependencies between files that make up the browser.

[8] also supports the idea that there is big differences between open source and proprietary projects. Indeed, in the case of open source projects, the distance between the developers, decision makers and testers is often wide. This is why it is very important

---

¹ Generally, open source refers to a program in which the source code is available to the general public for use and/or modification from its original design. Open source code is typically created as a collaborative effort in which programmers improve upon the code and share the changes within the community.
Motivation and Objectives

for functionalities to be independent on others. Thus, these functionalities can be managed by different groups. The organization therefore requires smaller groups who have less need to communicate among themselves.

The last years of research on the topic were used to develop a method to analyze the modularity within software. The steps were first extracted from the experimental results [8]. Then the method evolved gradually giving more information on the studied applications [9]. Finally, they created the hidden structure method [10].

The purpose of this document is largely based on this method. It consists in determining the type of each system component and then determining the structure of the application. Some groups of components are closely interlinked. These interdependencies reveal the importance of some components. We can also talk about key functionalities or core functionalities.

This method is suitable for very complex software. For example, the method handles more than 1500 components to analyze the Mozilla Firefox browser [8].

C. Why adding the Enterprise Application Modularity within EAAT

This master thesis aims to adapt and integrate the method of modularity analysis [10] within EAAT.

As said earlier, some of the latest research on the EA conducted by Industrial Information and Control Systems at the Royal Institute of Technology (KTH) focuses on modifiability ([2] and [7]). EA modularity is an important point of modifiability. The more modular a system is, the more simple it is to make modifications.

However, the analysis of the modularity of a system is not sufficient to completely describe the modifiability of the system.

One of my main master thesis goals is to allow the use of the hidden structure method ([10]) for analyzing the structure of an information system. It is particularly interesting to determine the key elements that carry strategic functionalities of a company. The understanding of information systems and EA will be facilitated.

The project is conducted in collaboration with Assistant professor Robert Lagerström and PhD student Markus Buschle of Industrial Information and Control Systems at the Royal Institute of Technology (KTH) and Professor Carliss Baldwin and Associate professor Alan MacCormack, Harvard Business School [11].
Motivation and Objectives

Dr. Robert Lagerström has been working for several years on the analysis of the EA modifiability [2] and [7]. While Carliss Baldwin and Alan MacCormack worked on the analysis of the modularity of software [8], [9] and [10].
II. Definition and context

A. Main definitions

1) Important words

<table>
<thead>
<tr>
<th>Word</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>A system is a set of elements which mutually interact according to certain</td>
</tr>
<tr>
<td></td>
<td>rules. A system is determined by:</td>
</tr>
<tr>
<td></td>
<td>● The nature of its components;</td>
</tr>
<tr>
<td></td>
<td>● Interactions between them;</td>
</tr>
<tr>
<td></td>
<td>● Its boundary, that is to say, the criterion of belonging to the system</td>
</tr>
<tr>
<td></td>
<td>(for determining whether an entity belongs to the system or on the contrary</td>
</tr>
<tr>
<td></td>
<td>of its environment).</td>
</tr>
<tr>
<td>Enterprise Architecture</td>
<td>Enterprise Architecture is a software engineering discipline of monitoring and modifying an information system in order to support and accompany effectively and efficiently goals of an organization and their transformations.</td>
</tr>
</tbody>
</table>

2) Ambiguity between words

This document, being a link between two different fields of research, uses two different lexical fields. That is why I must differentiate some words. Because, even if they look like synonyms, there are always some nuance in their meaning.

A component and an element are two synonyms, they both represent an elementary unit within a system. But in this document, I will exclusively use component to designate elements in an IT system. The word element will keep a general meaning.

An association, connection and dependency, all represent a link between two elements. Association and connection are two words coming from the UML standard, this standard

---

2: The “definition” word is unsuitable in this table. This is more my interpretation of the field after working on this subject and based on my personal background. These definitions are intentionally incomplete and reflects notions as I have understood and manipulated in this project.
Definition and context

is applied in the EAAT software. An association represents a link between two classes (in the meta-model) and a connection represents an instantiation of an association, it relates two objects (in the model). On the other hand, dependency comes from the hidden structure paper [9]. I will try as much as possible to exclusively use association or connection in the EAAT context and dependency in hidden structure context.

An Enterprise Architecture is a system which represents the structure of a company and generally focus on its information system. I will use system as a generic term except when I explicitly want to refer to the Enterprise Architecture.

3) Abbreviation

Table 2: Abbreviation

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Complete expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSM</td>
<td>Design Structure Matrix</td>
</tr>
<tr>
<td>EA</td>
<td>Enterprise Architecture</td>
</tr>
<tr>
<td>EAAT</td>
<td>Enterprise Architecture Analysis Tool</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>KTH</td>
<td>Kungliga Tekniska Högskolan – the Royal Institute of Technology</td>
</tr>
<tr>
<td>ICS</td>
<td>Industrial Information &amp; Control Systems</td>
</tr>
<tr>
<td>PRM</td>
<td>Probabilistic Relational Model</td>
</tr>
<tr>
<td>OCL</td>
<td>Object Constraint Language</td>
</tr>
</tbody>
</table>

B. General context

1) Kungliga Tekniska Högskolan (KTH): an international university

The Royal Institute of Technology (Swedish: Kungliga Tekniska Högskolan, abbreviated KTH) is a university in Stockholm, Sweden. KTH was founded in 1827 as Sweden's first polytechnic university and is one of Scandinavia's largest institutions of higher education in technology. KTH accounts for a third of Sweden’s technical research and engineering education capacity at university level. KTH offers programs leading to a Master of Architecture, Master of Science in Engineering, Bachelor of Science in Engineering,
Definition and context

Bachelor of Science, Master of Science, licentiate or doctoral degree. The university also offers a technical preparatory program for non-scientists and further education.

There are a total of around 17,000 students. KTH is one of the leading technical universities in Europe and highly respected worldwide, especially in the domains of technology and natural sciences.

2) Industrial Information & Control Systems (ICS)

More specifically, my internship took place at the department of Industrial Information and Control Systems (ICS).

This department is targeting the development of complete and cost-effective IT-based operation support systems for complex industrial processes. With the vision of making successful IT implementations commonplace, the department’s research includes the system management process from conceptual planning to operations. All research at ICS is carried out in close collaboration with industry.

ICS provide courses on an undergraduate level within the areas of IT management, requirements engineering, project management, and IT applications for power systems.

3) EAAT project

EAAT is a software package for modeling and analyzing EA. Markus Buschle mainly drives this project. Unfortunately, there are many methods and standards to model an EA but there are only few tools for their analysis.

It is in this gap that this project fits. EAAT is a flexible modeling tool. EAAT is able to define the structuring rules of an EA, documenting instances of these rules and the last but not least to analyze the system.

Like all model-driven disciplines, the creation of a meta-model is a key point of the analysis. To propose a single meta-model for all users is both dangerous and not very flexible. It is mainly because it is almost impossible to define all the possible case in a single meta-model. Therefore, the meta-model must be able to evolve. But if there is one meta-model for all types of systems, the meta-model is probably both too complex and not specific enough. This is why the creation of the meta-model became a step of the analysis process.
Definition and context

a) The modeling process

The classic use of EAAT follows a simple process divided into three steps:
1. First, creating the meta-model in the Class Modeler, this task of requires an EA expert. This step can be split into:
   a. Identify all types of components (list classes);
   b. Characterize each class (list attributes);
   c. Identify the associations that exist between classes;
   d. Add the logical layer (e.g.: how to determine one attribute from others). This step corresponds to the addition of the constraints with OCL. The OCL layer will not be detailed in this document. First, because it does not relate directly to the project (the Enterprise Application Modularity uses no information from the OCL layer). Then, because this step is very complex and may bring confusion.
2. Then in the Object Modeler (based on the already defined meta-model) can create a representation (a model) of the real world, the model, this requires to:
   a. List the objects that compose the EA;
   b. Add connections between them;
   c. Fill all known observable attributes;
3. Finally, it is possible to carry out calculations and analysis.

b) The Class Modeler

The Class Modeler is one of the tools that compose EAAT. This tool allows the users to define a meta-model of the enterprise architecture. This software was previously called "the Abstract Modeler" that illustrates its goal: creating the abstract rules that respect the EA. This tool gives really flexible possibilities to define structuring rules of architecture from really different enterprises. The designed meta-model respects the UML (class diagrams) and OCL standards. The UML language defines the relationship of the various classes (that represent actors or components) of the company. Each class has attributes. The OCL language is used to define the logical rules for analyzing the models of an enterprise. This language removes all ambiguities and adds a logical level to the modeling.

Figure 1 shows a screenshot of the Class Modeler. It is possible to see the common interface of the tool as well as a meta-model created for testing.
c) **The Object Modeler**

The Object Modeler uses the meta-model (designed by the Class Modeler) to instantiate the rules of the model. From this set of rules, it is possible to instantiate several objects to represent the studied system. So the first goal of this tool is to model a system and from this model it is possible to process some calculations (which are defined into the OCL part of the meta-model) in order to analyze the system. These calculations' main goal is to calculate unobservable attributes from observable ones.

Figure 2 shows a screenshot of the Object Modeler. It is possible to see the common interface of the tool as well as a model just created for the screenshot. The main GUI components are:

- In the middle, there is the editor of views
- On the left, there are trees to navigate into the model;
- On the right there is the property view.
4) BioPharma case
In the following document, the BioPharma case will be cited a lot as an example. It is one of the main sets of tests. It has quickly become the main one because it contains more elements and is more complex than others.

It contains 477 elements spread over 14 different classes with five different types of association.

More details will be given in the "VI.B. Test samples" section.
The “Hidden Structure” method

III. The hidden structure method

This chapter is divided into six sections, the first 5 show the main steps of the analysis. To get a better understanding of how works the method, refer to the last section (“III. F. An example”), it explains this method with a simple and concrete example.

A. The Design Structure Matrix (DSM)

A DSM is primarily a matrix that shows the structure of a system. It is a square matrix where each studied element is represented by a row and a column. The order of elements is the same for both rows and columns.

The values in the matrix are Boolean. If there is a "1" (true value) in cell [i, j] (i-th row, j-th column), this means that the i-th element depends on the j-th but it also means that the j-th element is used by the i-th. Also, all elements depend on themselves, thus the main diagonal is filled with "1"s.

To graphically represent a DSM, each dependency is replaced by a dot. The first element is placed in the upper left corner.
The “Hidden Structure” method

This matrix allows to visualize the direct dependencies. However, if an element "A" depends on an element "B" which depends on an element "C" then it is obvious that if the element "C" does not work then element "A" may also be impacted. Therefore, in some cases, it is also interesting to consider all the dependencies: direct as indirect. The matrix of indirect dependencies is called the "visibility matrix".

The hidden structure method uses these two matrices.

**B. First metrics**

From these two matrices, it is possible to calculate the first set of important metrics of the system and its elements.

The two first are quite similar: DFI and DFO. The DFI is the Direct Fan-In and the DFO is the Direct Fan-Out. These two metrics represent information about system components. These are integer values. The DFI is the number of elements, which directly depend on the studied element. Unlike the DFO, which represents the number of elements that directly depends on the studied element. It is easy to calculate the DFI by summing the "1"s in a column of the direct dependency matrix (the original DSM). It is easy to calculate the DFO by summing the "1"s on each row of the direct dependency matrix.
There are also the VFI and the VFO metrics. The VFI is the Visibility Fan-In and the VFO is the Visibility Fan-Out. They represent similar metrics as the DFI and DFO except that it concerns direct and indirect dependencies. Therefore these values are calculated with the visibility matrix respectively by summing the “1”s on columns and rows.

Finally, there is the "Propagation Cost". It is a metric that represents the whole system. This is the rate of dependencies of the number of possible dependencies. It can be calculated by summing the VFI (or VFO) and dividing by the square of the number of elements.

\[
\text{propagationCost} = \frac{\sum_{i=1}^{n} VFI_i}{n^2} = \frac{\sum_{i=1}^{n} VFO_i}{n^2}
\]

where \( n \) is the number of elements within the system.

High "Propagation Cost" represents a system where all elements are highly dependent on others. Such system is not considered as flexible.

**C. What is the Core**

In a system, there are functionalities that are key functionalities but they also represent weaknesses of the system. They are widely used and use many other functionalities. If one of these key functionalities were to be faulty then the whole operation of the company would be in danger.

With the information obtained above, we can easily identify some elements as very important in the system.

Beyond the elements widely used, there are certain features that are obtained in multi-elements. By analyzing the structure of a system, it is very easy to find these clusters of elements. In the hidden structure method, clusters are called "cores" and the largest cluster is called the "Core".

The cores are sets of components that depend a lot on each other. In the hidden structure paper [10], it focuses on the cyclic groups. Therefore, the clusters are considered as cyclic groups in the rest to the report.

The "Core" represents all the essential features for the system. All elements of a cluster depend on each other and the Core is the largest clusters. The Core is an important part of the analysis, it can quickly differentiate architectures of different nature. This is why the following of the analysis depends on the Core.

**D. Different types of architecture**

The first step to analyze the architecture of an application is to find the Core.
The “Hidden Structure” method

Then, determine if the size of the "Core" is sufficiently significant compared to the rest of the system. In [10], they decided that the "Core" must contain at least 5% of system components to be considered as having a sufficient size.

If the "Core" is too small, then we consider that the system has a "Hierarchical" architecture. Each element is independent from the elements, which control it.

Once identified that the core plays an important role in the structure of the program, it is necessary to know if it is dominant in the system. In fact, it is important to check if the "Core" is larger than the other cores otherwise the key functionalities of the system are spread over several clusters. In this case, it is a "Multi-Core" architecture. But if the "Core" is large enough in comparison with the other cores then it is a "Core-Periphery" architecture. In the hidden structure paper, they decided that the "Core" must contain at least 1.5 times more elements than any other core to be considered as having a sufficient size.

![Decision tree to determine the type of architecture](image)

**Figure 4: Decision tree to determine the type of architecture (extract from the hidden structure paper ([10]))**

**E. The exposed hidden structure**

To reveal the hidden structure described in the paper, it is important to understand what is desired. The goal is to separate the different elements depending on their role:

- Those that are used by many elements, but that depend on almost no other element, they are "Shared" element;
The “Hidden Structure” method

- Those that are used by many elements and that also depend on many elements, it is usually the elements that make up the "Core";
- Those that are hardly used and that depend on almost no other element, they are "periphery" elements;
- And those that are hardly used but depends on many elements, they are "Control" elements.

Recall that the VFI and VFO allow measurement of ideas like: "is widely used" and "depends on many elements". The VFI is an integer that gives the number of elements using the studied element, the higher the value, the more shared it is. The VFO is an integer that describes the number of elements that depend on the studied element, the higher the value, the more it controls other elements.

Thus, one way to reveal the hidden structure of the system is to sort the elements that make up the system according to their VFI and VFO. The paper suggests to sort by VFI ascending and then VFO descending. This sort will order the elements from the most "Shared" elements to the least "Shared" elements. The approximate order will be:

1. Shared;
2. Core;
3. Periphery;
4. Control.

The order is approximate because the list of elements is first sorted according to the VFI and then to the VFO. It is possible to have some elements that should not be first differentiated by their VFI but by their VFO. For example, some "Control" elements can appear before the "Periphery" elements.

The “Core-Periphery” architecture is the most encountered architecture. [10] presents more information on how to study this kind of architecture:

- The paper gives precise definitions of different types of elements based on metrics:
  - Core elements are members of the largest cyclic group. All Core elements have the same VFI and VFO, denoted VFI_{C} and VFO_{C} respectively;
  - Shared elements have VFI ≥ VFI_{C} and VFO < VFO_{C};
  - Peripheral elements have VFI < VFI_{C} and VFO < VFO_{C};
  - Control elements have VFI < VFI_{C} and VFO ≥ VFO_{C}.
- It is also possible to add a sort constraint forcing the elements to appear in the order: "Shared", "Core", "Periphery" and finally "Control";
- It is possible to calculate the architecture flow, this metric represents the proportion of non-peripheral elements in the whole system.
In the figure 4, there are four element types, which are clearly visible on the main diagonal. The order is (from above left to right in the bottom): Shared, Core, Periphery and Control.

With the sort method mentioned above, we can see that the matrix was transformed into block lower triangular matrix. In theory, only the clusters have some dots located above the diagonal.

With this arrangement, the architecture is revealed (I will always use the convention [row, column] to select a value in the matrix):

- The Shared column (the first one) contains lots of dots because those elements are widely used;
- The Control row (the last one) contains lots of dots because those elements depends on many other elements;
- Block [3, 2] (= [Periphery, Core]) does not contain any values, otherwise the Periphery elements would become Control elements.

F. An example

Here is a short example of a system with 6 elements (named from A to F):
The “Hidden Structure” method

In this case, each arrow represents a dependency (A depends on B, etc.), therefore the input DSM is:

\[ \begin{array}{ccccccc}
A & B & C & D & E & F \\
A & 1 & 1 & & & & \\
B & 1 & 1 & & & & \\
C & & 1 & 1 & & & \\
D & & 1 & 1 & 1 & & \\
E & & & & & 1 & \\
F & & & & & 1 & \\
\end{array} \]

And we can calculate the DSM of direct and indirect dependencies (it is calculated via matrix multiplications):

\[ \begin{array}{ccccccc}
A & B & C & D & E & F \\
A & 1 & 1 & 1 & 1 & 1 & 1 \\
B & 1 & 1 & 1 & 1 & & \\
C & 1 & 1 & 1 & 1 & & \\
D & 1 & 1 & 1 & 1 & & \\
E & & & & & 1 & \\
F & & & & & 1 & \\
\end{array} \]
The “Hidden Structure” method

From this matrix we can deduce that B, C and D form the only one cluster (sub square matrix among the main diagonal filled up of “1”). Therefore this cluster is the Core. The Core is larger than 5% of the system (50% > 5%) and larger than 1.5 times the second largest cycle (3 > 1.5*0) thus this system has a “Core-Periphery” architecture.

From this information we can deduce individual information. The DFI and DFO are obtained by summing the “1”s, respectively, on columns and rows of the direct dependency DSM. The VFI and VFO are obtained by summing the “1”s, respectively, on columns and rows of the visibility matrix (the matrix of direct and indirect dependencies). The type of items can be given because it has a Core-Periphery architecture. Elements with a larger VFI and a smaller VFO than the Core are of the Shared type. Elements with a smaller VFI and a larger VFO than the Core are of the Control type. And finally elements with a smaller VFI and VFO than the Core are of the Periphery type. Here, the order is the one recommended by the hidden structure method ([10]), VFI descending and VFO ascending:

Table 5: List of Elements (hidden structure classification)

<table>
<thead>
<tr>
<th>Element name</th>
<th>DFI</th>
<th>DFO</th>
<th>VFI</th>
<th>VFO</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>Shared</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>Core</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>Core</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>Core</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Periphery</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>Control</td>
</tr>
</tbody>
</table>

This table is also used to rearrange the input matrix (by reordering the elements of the matrix with the same order as in the table). The new matrix reveals the hidden structure of the system:
The “Hidden Structure” method

Table 6: Rearranged DSM (explicit representation)

<table>
<thead>
<tr>
<th></th>
<th>E</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>F</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The propagation cost is \( \frac{19}{36} = 52.8\% \) (it is the proportion of “1”s in the visibility matrix) and the architecture flow is \( \frac{5}{6} = 83.3\% \) (it is the proportion of non-periphery elements).

In this case, the propagation cost is very high (>50%). Elements of the system are very dependent on each other. Statistically, if an element does not work in the system, it will have an impact on 52.8% of the system. And architecture flows is also very high, which means that only very few elements of the system are easily editable.

The first elements to be improved are the elements that constitute the Core (B, C and D).
IV. Methodology

A. General organization

1) The team
During this internship, I was in close collaboration with three people:

- Robert Lagerström is the project owner. He is directly associated with Harvard Business School professors who created the hidden structure method. He is also my main supervisor. From the beginning of my internship to July, he was in Boston to work with Harvard Business School on this project;
- Markus Buschle is a PhD student and his project is the EAAT project. He is the architect, project manager, direct contact (within the department and for external parties) and requirements manager;
- Khurram Shahzad is the EAAT full-time developer, therefore he is the technical expert.

2) Communication
This project is independent of other EAAT projects. This gave me both freedom and responsibility for my decisions.

Having a lot of autonomy and being relatively inexperienced, I decided to always maintain a good level of communication. This decision was very important. Especially as the project owner and I were separated by about 6000 km. It starts with E-mails. Several times a week I did a summary of tasks completed, ongoing and future. For technical questions, I could directly see Khurram Shahzad, his office was really close. I could then discuss difficulties I had. Finally, I used essentially the office suite provided by Google Drive.

3) The Workspace
The project was divided into two distinct phases:

1. Create a complete and autonomous prototype
2. Integrate this prototype within the existing EAAT software

At each of these stages, I used a different workspace and different Eclipse version. I first used a version of Eclipse with no plugin to provide the least external conflicts in the creation of whole prototype (reading of the file, calculating and reporting generation).
Methodology

Then I used the version of Eclipse made available by the developer, this release contains all the necessary plugins to install and develop the application.

During development phases, I used the revision control software: Subversion (SVN). This type of software has two main aims:

- The first is to allow collaborative work between several developer (significantly reducing the integration time of the developed components)
- The second is to bring security to the developer, indeed, this software allows you to store all changes (it is possible to go back if there is a regression) and to store all the code on several machines (all developer computers and the storing server which is safer).

As my project is proceeding in parallel with the development of EAAT, Khurram (the responsible of the development) created me a branch. Branches allow multiple groups to work on a project while going in different directions. It is possible to merge these branches. This action allows integrating the work of all stakeholders.

**B. Getting started with the project**

1) The documentation

The first real mission of this project was to understand the subject but also the context of the master thesis. The project itself is a step in the study of EA. This step is not the first and it will not be the last. It is part of a logical sequence of events that led to this project. Beyond this idea, this project is a junction between two lines of research.

So I had first to familiarize myself with the subject of research of the ICS research team. It focuses on the analysis of EA and more recently the analysis of modifiability of the EA ([2] and [7]).

Then I learned how to use the hidden structure analysis developed at Harvard Business School ([7], [8] and [9]).

Uncovering these two research areas was a difficult task. Mainly because of the vocabulary. Some different words in the two areas mean the same thing and sometimes the same words represent different concepts. An important issue was therefore to assimilate without misunderstanding useful information in both areas.

2) Detail the needs

Discussing the needs of the project with Robert Lagerström had two goals. The first is to provide more detail and explicitly express the needs that could be implied in the first formulation. This step prevents any misunderstanding. The second concerns my
Methodology

involvement: the discussion about the needs allowed me to rephrase these needs. Consequently, I felt massively involved in the project. The reformulation of needs gave me an initial understanding of the subject.

The distance between the project owner and myself was huge, I relied primarily on two tools to do this task:

- Emails in order to ask questions primarily on the needs that I had not understood (for lack of information or incomprehension);
- Google docs (dynamic documents enabling simultaneous work), with such document it is possible to never have to separate the writing version and the correction. You can continue writing paper while someone else comments errors.

So I created a Google doc that collected reformulation of needs. This document also contained some propositions that I have re-demonstrated in order to have a better understanding of the functioning of the hidden structure method.

The needs are clearly divided into three major parts:

- Data import, allow the Object Modeler to be able to complete (or instantiate) the model from an external data source with a structure close to the DSM size;
- Calculation, allow the Object Modeler to perform on its own model (or part) calculations advocated by the hidden structure method;
- Report output, aggregate the results obtained by the method in a report, the report must contain at least the main charts, general metrics and detailed information for each object.

These needs have hardly changed over the project, but they have gradually detailed in their realization. The main changes in requirements have added new needs (GUI component, etc.).

One constraints not developed in these needs is that the method should be implemented in EAAT. Indeed, EAAT is a Java program that uses the Eclipse RCP framework. These is a constraint of the project because this choice must be respected and followed. This choice had only few impacts except when I needed to add some GUI components. I then had to learn how to use SWT (part of the RCP framework for creating GUI components).

C. Planning

1) First planning

I had originally planned to separate the internship period in three phases:

- Discover context and detailing the needs;
- Create a complete prototype;
- Integrate the prototype to the EAAT project.
Methodology

The two last phases of development should be conducted in parallel with the drafting of the thesis report.

2) Final planning
Here is the sequence of tasks as they actually happened. We can see that the chain has remained fairly linear (only few tasks performed in parallel). However, the drafting of the report was not made in parallel with the development of the application. I think the writing of the report should be made in parallel of the advancements and it was a mistake to do it at the end. But we must also realize that the understanding of the subject and the context of this project were made gradually.
**Methodology**

*Figure 9: Real planning (GANTT 1/2)*

- Start
- Read documentations
- Detail needs
- Test EAAT
- Create algorithm to find all clusters
- Reading file functionality
- Model structure
- Implement calculation process
- Test calculation results
- Development of the PDF functionality

*Figure 10: Real planning (GANTT 2/2)*

- Test calculation results
- Development of the PDF functionality
- Integration into the existing model
- Dissociate file reading and Hidden structure method
- Solve the incompatibilities between excel file and the model
- Test on the BioPharma case
- Add OUI component
- Correction and adaption
- Master Thesis Report
V. Implementation

A. Data structure

In all IT projects it is very important to choose the data structure. In my case, I program in Java which is an object-oriented language. It is therefore crucial to define the classes that will be used before to start anything. In addition, the prototype must be integrated within the EAAT application. Therefore, the structure of classes must be coherent with the existing structure in the program.

To carry out the various steps of calculation and analysis by the hidden structure method ([10]) I had to create several classes:

- **BooleanSquareMatrix**: As the name suggests, it mainly contains a two-dimensional array of Boolean, which represents a square matrix. This object has exactly the same structure as a DSM that greatly simplifies the calculations. Once created, this matrix should not be changed (it is important to protect data consistency). This is why I decided to build this class as an immutable class;
- **Element**: this class contains all the useful attributes of an object for the calculations (object name, class name, dfi, dfo, vfi, vfo and type), later this class will be replaced by the class representing objects in EAAT model. The type of the element is given by an inner enumeration. I also decided to store the index of the object in the input DSM and in the visibility matrix in order not to depend on the order of Elements in the listOfElement that composes the analysis (in fact, the order of this list is handled in several places);
- **Dependency**: this class shows the dependencies represented in a DSM matrix, later this class will be replaced by the class representing the instantiation of associations in EAAT model;
- **Cluster**: this class represents cluster of elements which have dependencies between them that form cycles;
- **DSM**: the latter class is the core of all my work: this class performs all calculations. I built this class as a singleton\(^3\) to protect the application to use several results of calculations on different subsystems at the same time. Thus, the user cannot mix several different results;

---

\(^3\) In software engineering, the singleton is a design pattern that restricts the instantiation of a class to one object.
Implementation

To give more flexibility to the method, I isolated the architecture types in an enumeration. So if the hidden structure method changes or if we create new types of architecture, it will be easy to modify the code. Indeed, this enumeration contains everything related to the architecture.

![UML representation of the data structure in the prototype](image)

In the rest of this section, I will use the names of these classes (with capital letters) to denote the technical element manipulated.

**B. File reading**

Reading a file is one of the basic features that were requested. The need is to help the user to faster complete the model. Indeed, the analysis of modularity has real interest
when the model is really large. The creation of such a model is a long and difficult task to achieve within the Object Modeler.

To automatically instantiate a model (or part of a model), it is necessary to have two types of information:
- Information about objects to instantiate (name of the object and of the class it refers to);
- Information on relationships between objects (names of two associated objects, the type of association and direction of the association).

A DSM matrix can well describe a system. It is why the project owner (Robert Lagerström) suggested to use this format for the input file. Not only is it a simple format, but it is the data format that Robert Lagerström received from companies. This format contains almost all the necessary information.

It was necessary to add the class name and the names of associations, though at first I ignored the name of the association. When I created the first version of reading, my only interest was to collect the information necessary for the calculation prototype. This prototype is interested in the presence of associations and their directions. Type of association and multiple associations had no interest. That is why I did the mistake not to handle association names. I will detail the addition of association names later in Section 5 Part 1.

For the file, the excel sheet has quickly established itself as an appropriate solution. This is a technology mastered by everyone. It lets define large tables. We then had to decide whether to read XLS and / or XLSX files. For simplicity, I focused on XLSX files. This solution was accepted by Robert Lagerström. In addition to limiting the reading to a single format, the XLSX format is simpler because it follows the XML standard.

The structure of tables will follow some simple rules:
- The table is composed by two heading columns and two heading rows;
- The first give the name of the classes which are represented into this file (this name must exist in the meta-model with exactly the same spelling and case);
- The second heading row and column give the name of the objects represented;
- Each cell in the body of the table can only take the values “0”, “1” or empty;
- The “1” value in the i-th row and the j-th column represent a dependence, it means that the i-th element depends on the j-th;
- The object name cannot be empty.
Implementation

Table 7: Example of input table for the reading file function

<table>
<thead>
<tr>
<th>Object model name</th>
<th>Class X</th>
<th>Class Y</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object</td>
<td>Object</td>
<td>Object</td>
</tr>
<tr>
<td>Class X</td>
<td>x₁</td>
<td>x₂</td>
<td>xₙ</td>
</tr>
<tr>
<td>Object x₁</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object x₂</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Object xₙ</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this example, “Object x₁” cell represents an object that instantiate the class “Class X” and “Object y₂” depends on “Object x₂”.

This table must be in the first excel sheet.

To read a file, the choice was quickly oriented to the POI API. This API is often cited as a solution to the problem of reading Microsoft Office files (XLS, XLSX, etc.). This API is free and open source and is maintained by the Apache Software Foundation⁴. Therefore, POI is a reliable choice.

The main steps of the file reading are:
1. Open the file and go to the first sheet;
2. Read the first two header lines and extract the list of objects. In order to define an en-point for the reading of these two lines, I added a rule: the cell, after the last object name on the 2nd row, must be empty;
3. Read the body of the matrix to generate the associations.

This feature was very simple in the prototype. It only read information but during the integration, this feature became much more complex. Indeed it was needed to compare the read information with the existing information in the model (see Section VI.B.).

C. Calculation prototype
To call the calculation process, the program only has to call the calculateDSM method (ArrayList <Element> listofElement). This method then proceeds to analyze the sub-model represented by the list of Elements sent in input. The calculation process simply goes one step after the other. The main steps have no loopback and there is only one conditional branch conditioned by results.

⁴: The Apache Foundation is a very large group participating in many IT projects. It is known and recognized in the community to provide very important and stable tools (Apache server, for example).
The main steps are performed in this order:

1. Initialize the indexInputDSM within the list of Elements;
2. Create the input DSM from the model, the model is represented by the list of Elements and their Dependencies (see the data structure);
3. The DFIs and DFOs are directly extracted from this matrix;
4. The visibility matrix (DSM of direct and indirect dependencies) is calculated;
5. The VFI, VFOs and the propagationCost are directly extracted from this matrix;
6. From this metrics, it is possible to sort the list of Elements by VFI descending and VFO ascending. During this step, it is necessary to initialize the indexRearrangedDSM in Elements;
7. Find clusters;
8. Sort them from the largest to the smallest. The first one, the largest, is the “Core”;
9. Determine the architecture of the system;
10. If the system has a “Core-Periphery” architecture, there is some additional calculations to do:
   a. Determine the type of each element (Shared, Core, Periphery or Control);
   b. Sort by Type (Shared, Core, Periphery and the Control);
   c. Calculate the architectureFlow (the proportion of non-peripheral Elements).

Most of these steps are both fairly simple and well explained in the hidden structure paper [10]. I would like to focus on a few key points in this process. This is where the hidden structure method steps do not specify a recommended solution or steps do not correspond to the needs of the project owner.

The first is the different sorting algorithms selected in the project. I focused on two different types of sorting, quick sort and selection sort. Quicksort is used for lists that are supposed to be long (such as lists of elements). Given that sort lists have no pre-established order, quicksort is both one of the easiest and one of the most efficient sorting algorithms. Selection sort is in turn used for lists supposed shorter (such as lists of clusters). This sorting algorithm is more suitable because it is simpler. Therefore, I used quicksort to sort lists of Elements. These lists are usually long. Unlike lists of clusters, with which I used the selection sort.

By doing the calculation prototype, I realized that it was possible to create a system where some elements will fit with any of the types established by the hidden structure method in the case of a Core-Periphery architecture. Indeed, the Core represents the largest cluster. Shared elements have a larger VFI and a smaller VFO. Control elements have a smaller VFI and a larger VFO. Finally the Periphery elements have a smaller VFI and a smaller
Implementation

VFO. But it is possible to build some cases where VFI and VFO are both larger than those of the Core.

![Diagram](image)

*Figure 12: Theoretical case with a Core-Periphery architecture (the element C does not belong to any type)*

To ensure that the calculation prototype is never faced with a case not covered by the algorithm, I decided to add an element type. This type was named "Bottleneck" because there is a lot of information that flows through this type of element. But being a very marginal type, it is very unlikely to meet in real cases.

The last but not the least, in the method, the research for clusters is limited to finding the largest of them. In fact, some small clusters structure can hide other clusters. In the hidden structure paper [10], this situation is called a “Coincidence”. The paper also claims that "When VFI and VFO are large, the probability of coincidences is small and for practical purposes can be ignored". This solution does not suit to Robert Lagerström who has interest in finding all clusters. That is why I had to adapt the previous method to the research of all clusters.

The first question to answer is: How can we characterize Clusters?

- **Proposition 1:** all Elements of the same Cluster have the same VFI and VFO.
  - **Proof:** In fact, if A and B are two Elements of the same cluster, then A depends on B (we also can say that A uses B). Therefore, indirectly, A use the same Elements than B so VFO\(_A\) >= VFO\(_B\). But if A and B are two Elements of the same Cluster, then B also depends on A. Therefore VFO\(_A\) <= VFO\(_B\). We have VFO\(_A\) == VFO\(_B\). By the same reasoning, we can prove that VFI\(_A\) == VFI\(_B\) and by transitivity we can say that all Elements of the same Cluster have the same VFI and VFO.

- **Proposition 2:** all Elements that take part in a cluster have a VFI and a VFO strictly superior to 1.

- **Proposition 3:** We set M the *visibility matrix* and A and B two Elements of M, if M(A, B) = M(B, A) = 1 then A and B are in the same Cluster.

- **Proposition 4:** if A and B take part in the same Cluster and A and C take part in the same Cluster, therefore A, B and C also take part in the same Cluster.

- **Proposition 5:** if the Element E take part in the cycle C and if it does not exist any element E' \(\notin\) C such as E' and E form a subcycle (proposition 3) then C is the
Implementation

The largest cycle which contain E and it the same for all elements which compose C. We can say that the cycle C represent a Cluster. These four propositions are the theoretical bases to describe the algorithm below.

The proposition 1 is a necessary but not sufficient condition. It allows to reduce the range of the cluster research. Indeed, extracting subsystems which has the same VFIs and the same VFOs can quickly restrain research. The proposition 2 is also used to eliminate the Periphery Elements.

In some cases, it is theoretically possible that in the extracted sub-system, there is not the same sub-VFIs and sub-VFO for all Elements. That is why I recursively apply the algorithm for searching potential clusters until having the complete list of smallest potential clusters. I call potential cluster a subsystem, which has for all Elements the same sub-VFIs and the same sub-VFOs (the sub-VFI and the sub-VFO just represent the VFI and VFO recalculate in the sub-system).

This step uses the sorting method by VFI and VFO (the quicksort) which has a complexity of \( o(n \cdot \ln(n)) \). In general, the total complexity of this step will be the same as the sorting algorithm because recursion will almost never applied. Indeed, the cases representing the real world are often simple and recursion is rarely called more than once. However it may be possible to mathematically construct cases with the complexity \( o(n^2 \cdot \ln(n)) \).

So the first step of cluster research is to isolate and to list potential clusters. The second step is for each potential cluster to validate if this is a Cluster and otherwise extract Clusters that can be hidden within the potential cluster. Indeed, potential cluster can hide multiple real Clusters (it is exactly what was called coincidence in the hidden structure paper [10]).

In fact, the method that I have developed to extract clusters can be used on all systems (even on raw systems). However, its efficiency is very low in general cases. That is why it is applied on potential cycle because its efficiency increases when clusters represent a large part of the system. Later, I will come back on the complexity of the method.

The first step is to find all the elements of potential clusters that satisfy the proposition 3 with the first Element. So all these Elements (including the first one of the potential cluster) are part of the same cluster according to the proposition 4. All other Elements does not satisfy Proposition 3 so according to proposition 5, all previous Elements form the whole Cluster.
Implementation

Once a Cluster has been extracted, all Elements that compose this Cluster are removed from the potential cluster. Then this algorithm is recursively applied until there is no more Elements in the potential cluster. Of Course, if an Element never satisfies the proposition 3, this means that it is part of any cluster and it is removed from the potential cluster.

This step where clusters represent only a very small part of the system, the complexity tends to $o(n^2)$. However, when this algorithm is used on systems composed almost entirely of clusters (almost all potential clusters), the complexity tends to be linear and therefore it tends to $o(n)$.

D. PDF generator

File generation is an important feature. Indeed, the analysis by the hidden structure generates large results. It is difficult to transcribe all these results in the GUI. This is why at the beginning of the internship, Robert Lagerström had the need to generate a report summarizing the results. This report shall contain graphs representing the two important DSM: the inputted DSM and rearranged DSM. The format that was chosen is the PDF because it is now a universally recognized open standard.

During my previous internship, I worked with an API to generate documents (in various formats: PDF, DOC, DOCX, XLS, XLSX, RTF, HTML, etc.). So I quickly oriented my solution on this API that I already had used. In addition, I knew that this API is powerful, stable, trustable and accompanied by a large and active community.

This is the Jasper API [b]. This is an Open Source project around which one a consulting company was created. This API consists of three main components:

- The library is the main, there are several jar files that must be incorporated in the project to be able to access to Jasper features from the code, this library is called JasperReport;
- iReport is a WYSIWYG tool to facilitate the generation of report templates (I will come back on the concept of template), this tool is only used during the development phase;
- JasperServer is a server for hosting templates, the server is usually connected to the databases and can generates some reports without having to create a program to do it, this tool is dedicated only to web projects and it will not be used here;

These three tools are always developed in parallel, so each component is always compatible with each other component of the same version. I used version 5.0.4 which was the last stable version when I started using this API.

---

5 WYSIWYG: What You See Is What You Get, this designates tools with advanced graphical interface for the realization of low-level but difficult tasks (e.g. Word is a WYSIWYG tool for writing advanced report)
A little earlier, I mentioned the word "template". A template is the basic structure of the document that you want to create. This structure may be considered as the backbone of the report. In a template, there is in general:

- All static elements of the report (the elements which do not depend on information generated by the program);
- A description of the manner in which dynamic information will fulfill the report (the dynamic information is information that is calculated by the program and cannot be known until the runtime).

Using iReport, the creation of template is simplified. In addition, iReport can compile the template before adding it to the project. Let me explain, the template is described by an XML file that has the ".jrxml" extension. It is possible to compile this XML file in serialized Java objects that carry the ".jasper" extension. A ".jasper" file is already a Java object, therefore it is faster to load it than a ".jrxml". This step takes calculation time that is why it is very interesting to provide the program with the already serialized template (this simple action saves computation time when generating the report).

The three main steps that can generate a report from software are:

1. Create the template representing the report that you want to generate;
2. Fill the template from a data source (Jasper allows the use of many different data sources such as a database, a CSV file, a JavaBean collection, etc.) In our case, we use a collection of JavaBean\(^6\);
3. Generate the document, the generation can be done by saving the file under an available format (see above) or can be directly printed.

Editing report is a new feature in the EAAT tool suite, it is possible that in the future, this feature will be used by other application components. It is for this reason that I created a PDF factory independent from calculations made by my prototype. This decision has several advantages, this two components are not coupled, which gives a lot of flexibility. But it also helps to provide a good example to generate a PDF with this API. The "generatePdfReport" method takes for argument the template name, the parameters, the data source and the output path of the file and generates a PDF file.

A template is supposed to be simple, it should not process to any calculation. Therefore, the calculation data contained in the singleton DSM and Elements are converted into JavaBeans before being sent to complete the report. These JavaBeans are built from DataSourceFactory this class respect the rules of a factory and contain only simple information, which can be directly printed in the report.

\(^6\) JavaBean is a really simple type of object which is used to manipulate information in the software.
Implementation

The report contains a simple header with the title and KTH logo and is structured in three main sections:

- General information: this section provides general information such as the propagation cost, the architecture flow, architecture (in the case of architecture "Core-Periphery", there is also a summary detailing the number and the proportion of each types of elements) and two graphs (the inputted DSM and the rearranged DSM)
- The list of Elements that gives the total number of Elements and the detail of each of them;
- The list of clusters, which gives the number of clusters and for each its size and the list of Element that compose them.

There is a sample of report in the appendix A.

To generate the charts, I used a library that is used by JasperReport. It is the JFreeChart [c] library. It is a well-known open source project. This library is often cited as the most widely used library for generating charts. As this library is a dependency of JasperReport, I did not have to import new jars in the project.

The DSM are represented by some scatter charts. The y-axis is oriented downwards. The coordinates of the dots are based on indexInputDSM of each Elements for inputted DSM and the indexRearrangedDSM of each Elements for the rearranged DSM. I also removed the display of the axes, the legend and the colored background.
VI. Integration and Tests between hidden structure method and EAAT architecture

A. Integration and adaptation to the EAAT structure

1) Integration of reading files
When I started to integrate the reading files functionality into the Object Modeler, it seemed obvious to me that this feature should not be coupled with an analysis of the hidden structure. The reading has for main purpose in to help users to instantiate their model. This is a feature that any user may need.

So these are the main rules that I imposed to this feature when I integrated it:

- This feature can be used to instantiate a model or to complete a model, so it should not delete any information;
- This feature can be used to add new information (objects or associations between objects) to existing model;
- As it is possible to add association between existing objects, the feature must be able to manipulate existing objects, without duplicating objects.

To understand how reading file is integrated within EAAT, it is important to have a global vision of the modeling rules applied by EAAT. EAAT software is divided into two tools. The first one (the Class Modeler) handles the class diagrams to create the meta-model. The second program (the Object Modeler) manipulates objects diagrams. I am actually working on this particular type of diagram. Basically, an object diagram is represented by its objects and the associations between these objects. There may be several associations between two given objects. But an association of one type can be instantiated only once between two identical objects.

There is a major problem with the DSM format: it does not contain details about associations. The only information contained is if there is a connection between two objects. We cannot know what kind of connection it is or how many connections are concerned. All connections are instances of associations. This information gives a meaning to the connection. Therefore, the “1” in Excel files do not give enough information. It is important to have the list of associations instantiated between this to objects.

Alexandre Sirko 2013
Integration and Tests between “Hidden Structure” method and EAAT architecture

In the excel file, the “1” value which represents a connection with a list of association names separated by a simple space. Thus it is possible to add a list of defined associations. This requires from users to name all the associations they want to import (it was not required before). Adding associations into the file made it more complex.

One of the most important parts of the integration into EAAT is implementing numerous checks to ensure that the added objects are consistent with what already exist. The first thing to check is the existence of classes. If one of the class names in the file does not exist, it must generate an error and interrupts the reading. Then it must check whether the read objects already exists or not. There are three different cases:

- The name of the object is unused (there is no object with this name), then a new one is created;
- An object exists with the same name and the same class then this existing object is used without creating new one;
- An object exists with the same name but it instantiates a different class, then a new object is created but a warning is also emitted to the user.

When we read the names of associations, if one of the names of the associations does not exist then the program generates an error and terminates the process. All these checks make the file reading slower (in the case of BioPharma, file reading takes around 20 seconds to about 500 objects).

During the integration step, it was also necessary to adapt the object and association creation in the model to the data structure in the code. The Elements became IPrmClasses and Dependencies became IAssociations.

This feature is highly dependent on the file structure. If the user does not provide a well-formed file, the program must detect and capture the error. That is why I've listed and characterized all the possible errors during the reading:

<table>
<thead>
<tr>
<th>Exceptions Names</th>
<th>Origin</th>
<th>Impact</th>
<th>Reaction</th>
<th>Type of the Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileNotFound</td>
<td>The specified path to the file is wrong</td>
<td>High</td>
<td>interruption of reading</td>
<td>error</td>
</tr>
<tr>
<td>IOExceptions</td>
<td>problem when the program try to open the file</td>
<td>High</td>
<td>interruption of reading</td>
<td>error</td>
</tr>
<tr>
<td>InvalidFormatException</td>
<td>the file is unreadable (it is not a XLSX)</td>
<td>High</td>
<td>interruption of reading</td>
<td>error</td>
</tr>
</tbody>
</table>
Integration and Tests between “Hidden Structure” method and EAAT architecture

<table>
<thead>
<tr>
<th>Exceptions Names</th>
<th>Origin</th>
<th>Impact</th>
<th>Reaction</th>
<th>Type of the Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileMalformed</td>
<td>the file is not structured as expected (a problem with the heading rows in the file)</td>
<td>High</td>
<td>interruption of reading</td>
<td>error</td>
</tr>
<tr>
<td>ClassNameNotExist</td>
<td>a class name does not exist into the meta-model</td>
<td>High</td>
<td>interruption of reading</td>
<td>error</td>
</tr>
<tr>
<td>ClassNotMatch</td>
<td>an existing object has different class than specified in the file</td>
<td>Low</td>
<td>creation of a new object with the new class</td>
<td>warning</td>
</tr>
<tr>
<td>FileMalformed</td>
<td>empty row in the middle of the file</td>
<td>High</td>
<td>interruption of reading</td>
<td>error</td>
</tr>
<tr>
<td>AssociationNotFound</td>
<td>an association is not available for the current objects</td>
<td>Intermed</td>
<td>ignore this association and continue the reading</td>
<td>error</td>
</tr>
</tbody>
</table>

2) Integration of calculations
The integration of the calculation prototype was a relatively simple task. The two main reasons are:
- The calculations are completely independent of the rest of the applications, only the data coming from the model are useful;
- The model has already an orientation for associations in the diagrams (although it is not required by the UML standard).

I mainly had to adjust the IPrmClass class (representing objects) so that it can handle the major parameters of the hidden structure method (DFI, DFO, etc.). In the prototype, it was also necessary to change the references to the classes Element and Dependency by IPrmClass and IAssociation and calling new methods.

3) Integration of report generation
To make the generation of the report the most flexible possible, I have isolated this functionalities into two new plugins of the project:
- The first plugin stores the library. Therefore if we want to update the library, there will just requires to change this plugin;
- The second is the reporting plugin. There are all the necessary elements to create data sources, graphics and report parameters. There is also the PdfFactory.
4) Adding GUI component
In order to interact with the analysis module, I added some GUI components.

In the following section, there is a word that has two meanings. It is the word "view". This word can be used to represent a concept belonging to the Eclipse RCP framework. The whole EAAT project is based on this framework. In fact, in the Eclipse RCP framework, a view is a type of GUI component. It represents a special type of windows. It is typically used to navigate in a hierarchy of information, open an editor, or display properties. But the word "view" is also a concept used in the EA modeling. Indeed, a view represents a subset of the EA model. Each allows you to represent a sub-list of objects with some of their connections.

a) Basic interactions
The first components added can load files to complete the model and run the calculations on the whole model. I decided to regroup these features in the "DSM" submenu of the "File" menu. So I created the "DSM" sub-menu and I added two actions:

- "Load Excel"
- "Calculate DSM"

![Figure 13: Screenshot of the "DSM" submenu](image)

b) Calculations on a sub-model
Another important need of this project is to allow the user to perform an analysis of the hidden structure of a sub-model. A sub-model simply consists of list of objects. This list can be any subset of the system. In fact, the calculation prototype is already able to perform calculations on a list of objects (IPrmClasses), the problem is to select the object list.
Integration and Tests between “Hidden Structure” method and EAAT architecture

To keep a good level of flexibility in the application, it is interesting to select the list of objects just before the execution of the calculations.

To do this, I first looked at the notion of view in EAAT application. A view can represent a subsystem of the model. Indeed, a view focuses on a specific subset of the system, it can represent a branch of a hierarchical system, a set of elements involved in the same function or just some types of elements (instantiation of some specific classes). This notion of view (as defined in EAAT) is exactly the idea of subsystem. So I added the "Calculate DSM" action in the "Explorer View" context menu. The "Explorer View" is a view (GUI component) which allows you to navigate between views (as defined in EAAT).

This idea perfectly meets the need in the theory, in practice, the construction of views is done manually which can be a very long task. Indeed, models that are studied by analyzing the hidden structure are usually very large. It is therefore important to support the construction of such views (as defined in EAAT) by users. It is important to automate the creation of views. By analyzing this problem, it appeared that there are three major types of views:

- Those based on classes, which contain all the instances of one or more classes;
- Those based on associations, which contain all the object instances using any of the associations studied;
- Those are based on a list of any object.

It is possible to select such a list in the “Object Model Explorer”. The "Object Model Explorer" is a view (GUI component) that let the user interacts with the sets of all the created instances. The instances are sorted by class and for each one, it is possible to display the association instances (also called "connection").
Integration and Tests between “Hidden Structure” method and EAAT architecture

Because this view (GUI component) contains the information needed to automatically create a view (as defined in EAAT), I added the action "Generate or update a view" to the context menu of the “Object Model Explorer”. This action will generate a view based on the selection. The selection can be of three types:

- A list of classes, the generated view will contain all the objects that instantiate one of these classes;
- A list of connections, the generated view will contain all the objects that use one type of connection used by the selected connections;
- A list of objects, the generated view will contain all these objects.

If the selection does not meet one of these three formats, the action "Generate or update the view" is not enable.

After some tests on broad views, a problem has been highlighted: the program begins to have some lags when manipulating views consisting of many objects. This problem comes from the module dedicated to optimize the display of associations in the interface. This module calculates and optimizes the shape of lines between objects that symbolize
Integration and Tests between “Hidden Structure” method and EAAT architecture

display the connections. When there are too many connections, the computation time becomes longer than the response time and the program starts to generate lags.

Disabling optimization module was not the best solution. Finally, I also added the ability to generate calculations directly from the same type of selection, which allow to generate views.

![Figure 15: New context menu in the “Object Model Explorer” (with the “Generate or update a view” and the “Calculate DSM” actions)](image)

![Figure 16: New context menu in the “View Explorer” (with the “Calculate DSM” action)](image)

**c) New views**  
To dynamically and directly visualize the results in the interface, I added two new views (GUI component).

"DSM Results" is the first view, it gives the general results of calculations like the propagation cost, the architecture flow and the two graphs (input DSM and rearranged DSM).
The other view (GUI component) is called "DSM Element" and summarizes the most important information related to the selected object. The selection of the object can be performed from the "Object Model Explorer" and from a view (as defined in EAAT).
For the generation of the PDF, it was decided that as the generated report contains all relevant information, report generation will be invoked systematically. Once the calculations made, a dialog box appears asking the user to save the resulting report. If the user decides to save the PDF, it will be saved to the specified address otherwise, if the user cancels, all actions related to PDF generation will not be performed.

This dialog box opens for all types of calculations (on the whole model, on sub-model based on a selection of views or on sub-model based on a selection made in the "Object Model Explorer").

**B. Test samples**

1) Telecom case

During the development of the calculation prototype, Robert Lagerström gave me a full case study. This study contains both all input data required for analysis by the hidden structure and also all the results obtained by the analysis. This case has also been studied in detail in the working paper [12].

This study was forwarded to me in the format of an Excel file containing all pieces of information. The file is divided into several worksheets, each of these sheets contain different types of information (input DSM, overall results, individual information, etc.). In addition to be a set of tests, this file was an important document for understanding the operation of the method [10]. This file was a very important reference in the development of prototype calculation. At each step, I checked if my results do not deviate from those expected (in the file).

Some subtle rules of operation of associations and connections have was highlighted with this test set. The file contains 103 elements named "sc1" to "sc103". To respect the constraints of EAAT, I added them all to the same class: "Sc". I created a really simple meta-model with only one class Sc and only one association named "a" that links Sc to
Integration and Tests between “Hidden Structure” method and EAAT architecture

itself. When I tested, I discovered that some connections were not created. When two objects with the same combination used in both directions (e.g. sc4 → sc10 and sc10 → sc4) only the first association that was read in the file was created. From this statement, I added a second association meta-model called "a2". In the input file, I also replaced all the "a" associations were above the main diagonal by “a2" association.

2) BioPharma case

This second case is much more complex. It contains 477 elements spread over 14 different classes with five different types of association. The first step is the development of the meta-model to respect the classic cycle for modeling with EAAT.

To create the meta-model, it is first necessary to extract the possible associations between each object. I first thought about doing it manually, but the model was too large (477 * 477 = 227,526 cells). So I wrote a small program that can read and analyze the file. This initial analysis helped me to realize that some associations were used by several classes. The modification of the file is required to respect the modeling rules of EAAT application. I first tried to differentiate each different associations.

To do this, I first listed the couple of classes that form each association. Then I delete from the list associations that exist only for a couple of classes. This manipulation allowed me to get a list of problematic associations (used by several different pair of classes). To differentiate the associations, I added a number to their names. For example the association "Uses" was differentiated for each pair of classes "Uses1", "Uses2", etc.

This first modification resulted in a file without ambiguity between class couples and associations used. Only, a couple of classes is oriented. It is therefore necessary to differentiate couple that can be used in both directions.

This step had the same pattern as the previous step, I first identified all problematic associations then I differentiated them. When an association is used in both directions, I differentiate by adding "-a" or "-b". For example, "CommunicatesWith5" was used in both directions between two object that instantiate the class "ApplicationComponent", therefore the association was divided into "CommunicatesWith5-a" and "CommunicatesWith5-b".

Finally, the program provides an exhaustive list of associations along with their associated couple of classes. This list has provided all the necessary pieces of information to create the meta-model.
As BioPharma case is larger and more complex than the Telecom case, it quickly became a standard to measure the computing time in my project. All computation time of this report are based on this case. I also used its DSM charts as examples.

This case has not yet been detailed in a paper, so I'll give some figures to give some pieces of analysis: it contains 477 elements and has a "Core-periphery" architecture. The largest core contains 30.61% of EA elements (5.0% to have a sufficient size). The largest core is 36.50 times larger than the second one (1.5 to be considered predominant). The propagation cost is 23.71% and the architecture flow is 72.75%.
VII. Discussion and conclusion

A. Summary of results
In this master thesis, the method of hidden structure has been implemented and integrated in EAAT. It analyzes the architecture of a system by analyzing the structure of dependencies that exists between elements that compose the system. It is able to identify clusters of elements interconnected and widely used. These clusters are key functionalities and also represent system vulnerabilities.

This method is supported by two other features which were also developed during this project.

The first of these functionalities is dedicated to help the user to instantiate a model from an excel file. This feature allows the user to quickly instantiate a very complex model. The system can then be described in one or more Excel files respecting the DSM format described earlier. This feature does not depend on the hidden structure analysis, so it can be used by all EAAT users although it was initially thought as part of this project.

Then there is the report generation under the PDF format. It allows the user to create a report aggregating the data calculated during the hidden structure analysis. This feature has been developed to be flexible in the future. This means that if a future version also requires generating a PDF, the current PDF generation functionality can be used by making only very low modification.

B. Main ambiguities

1) Association and their names
The OCL (which allows the user to add logical information to meta-models) requires oriented associations between two classes. This orientation is transferred to all instances in models. These constraints should be considered carefully because it is impossible to instantiate two similar connections between the same two objects.

For example, when two objects a1 and a2 are two instantiations of the class A and it is required to create two connections a1 → a2 and a2 → a1, it is necessary that A has two associations (with different names) with itself. Otherwise a1 → a2 and a2 → a1 could not
coexist at the same time (because they represent the same association \( A \rightarrow A \) and \( A \rightarrow A \)). These two connections represent the same association \( (A \rightarrow A) \) between the same pair of objects (a1 and a2). I have noticed this problem with the test sets, my results was different from expectations.

It may seem strange to have to define two associations that have the same goal. Especially since it is only to allow two connections to represent the same thing simultaneously. Constraints that respect EAAT also become my constraints. They are absolute and I cannot transgress them. Given that it does not affect the proper functioning of the tool, they are retained. It is important that every user knows this fact in order to avoid mistakes.

There is another important constraint, which impacts the project: two associations that relate to the same class (e.g. A, B and C are three classes, ass1 represents the association \( A \rightarrow B \) and ass2 represents the association \( B \rightarrow C \)) cannot have the same name. This rule affects the project because it often happens that two objects of different classes depend on each other through connections with the same name.

The problem appears more clearly in an example: let A and B be two classes a and b, respectively, two instances of A and B. Both connection \( a \rightarrow b \) and \( b \rightarrow a \) are instances of associations \( A \rightarrow B \) and \( B \rightarrow A \) that are two different associations. In this situation, the two associations should have different names. This requires a lot of discipline for the user to define different association names for each case that may exist.

2) Object Modeler and Class Modeler dissociate

The entire master thesis has only brought changes to the Object Modeler. This is primarily because the technique of analysis by the hidden structure does not depend on the meta-model used. This is both a constraint and a goal to separate the hidden structure analysis and the creation of the meta-model.

It also should be remembered that this method is not a goal in itself, it is a step in the understanding of the EA. This implies that some of the results can be used in other analyzes that will be added later. It is therefore sometimes necessary to interact with the meta-model. The meta-model defines the structure of the model, so it defined the attributes and how the calculations use them. The method is independent of the meta-model so it cannot interact with the attributes that can take any form.

It is important to keep in mind that Enterprise Application Modularity is evolving. It is also possible that the results (numerical parameters as VFI and VFO) are used for future work. It is therefore important to make the variables calculated by Enterprise Application Modularity available for further calculations on the model.
Discussion and conclusion

The first choice to do is: determine what variables must be accessible. It is possible to separate all parameters into two categories.

- General variables (such as the propagation cost), these variables concern the whole study. They are not link with any component of the model. Therefore it is really difficult to link them with the model (they are only accessible through the singleton DSM that contain them);
- Specific variable (such as the VFI), these variables only concern one object (IPrmClass). These variables are directly included into the model. It is therefore interesting to make them available for future improvement.

The second choice concerns the accessibility to this information. Calculation methods are defined in the meta-model. This is the logical layer (OCL) that defines the calculations. This logic layer handles classes and attributes. To use the data calculated by Enterprise Application Modularity in the logical layer. The results of Enterprise Application Modularity must be parsed with classes or associations. In fact, the specific parameters calculated by Enterprise Application Modularity (DFI, DFO, VFI, VFO and Type) are very similar to the attributes created in the meta-model. They are also attributes but they are directly defined in the code.

During the calculation runtime, if some specific attributes exist in a manipulated objects, the calculated values are assigned to them. For example, if an object has the "VFI" attribute and it is concerned by the calculation of the hidden structure then the value of the "VFI" attribute will be updated.

Here is the complete list of attributes that can be manipulated in and out of the hidden structure module:

- DFI
- DFO
- VFI
- VFO
- TYPE

These names are not case dependent (the user can write DFI, Dfi or dfi, it will be the same).

**Be careful with the "TYPE",** it is only possible to create Boolean or numeric attributes. Therefore, the type of an object is converted to an integer (1 for Shared, 2 for Core, 3 for Periphery, 4 for Control and 5 for Bottleneck).
Discussion and conclusion

We must be careful never to use these attribute names for anything other than their meaning in the hidden structure method when DSM calculations are performed. If their values are overwritten with the values calculated by Enterprise Application Modularity.

C. Possible improvement

Some points can be improved in the project. For the most part, they have not been carried out for two reasons. The first is that they are not a priority and the second is a simple lack of time.

Reading the file is long. It might be interesting to optimize this reading. The steps, which increase the complexity of the reading are those that verify the existence of elements. It would be interesting to factor these steps, especially if the size of file increases. For example, for each element name in the file, a series of verification is carried out to check if an element exists in the model with the same name.

Following the same idea, it would also be interesting to remove the "list of Elements" section in the report. Indeed, when the studied system is composed of many elements, the generation of this section is very long. It is therefore important to bring new thinking to the limit number of elements before removing this section in the report. Especially when there are a lot of elements, they are very difficult to analyze.

None of the functionalities developed is multithreaded. This means that when an action takes time, the software stops working until it has finished its calculations. In the case of the hidden structure analysis, it often happens that the models is very large, so the calculations take more time. It is therefore common to wait a few second for the program to respond. In the case of BioPharma analysis (about 500 objects), reading the file takes about 20 seconds, the calculations take less than 5 seconds and report generation takes about 15 seconds. During these steps, the software is no longer usable. It might be interesting to inform the user about the progression to avoid that he interrupts the process.

It would be interesting to be able to quickly view the cores of the system. Indeed, these clusters contain the key functionalities of the system, therefore, the study could focus on them. This proposal has not been considered because the clusters, especially the Core have very large sizes. But EAAT views already have some problems to handle lots of elements.

Which brings us to another proposal: optimizing rendering views that contain many elements. Thus it would be possible to create large views. However the interest of large views is limited because it is difficult to see important information on them. Disabling the optimization of the lines could reduce the lags of these large views (lines represent associations and connections in the diagrams).
Discussion and conclusion

It is also possible to interact with the graphics generated in the GUI (both DSM in the "DSM results" view). It is already possible to zoom, save it as an image directly from the GUI and command their printing. It might be interesting to color or highlight the column and row that represents a selected element.

D. Future work
Paul Gaborit (one of my teachers) told me once that programs never have errors, bugs or any security breaches, until someone finds one. Like all IT projects, maintenance of the code cannot be avoided. Hoping this will not be a heavy task.

The method of hidden structure continues to be developed by Harvard School Business. It is likely to evolve in the future. It is possible that other metrics will be developed to analyze the studied system.

One of the topics that were discussed was the importance of clusters and especially the Core. In future developments of the method, cluster analysis will become more and more important. New steps in the hidden structure method will probably be added to focus on the cluster analysis.

It is also possible to use the results in DSM other type of calculation. The data calculated by the hidden structure method (DFI, DFO, etc.) can be associated with attributes defined in the meta-model. This junction between the method and EAAT may allow reuse of data. For example, the VFO could become an observable parameter of the modifying cost analysis of an EA.

E. Conclusion
The introduction raises an important question: "Is the architecture of EAAT compatible with the hidden structure method?". This question was never clearly answered in this document. There are several reasons for this. The first is that EAAT respects the UML standard (class and instance diagrams). This standard is used to represent a system as described in [10]. It was obvious that the two subjects were compliant. Therefore, the project has been quickly focused on the implementation and integration. The question, considered as trivial, has been put aside in favor of the tool realization.

It should also emphasize an important point: the steps that lead to the instantiation of the model should be made with care. It important to create a meta-model that respect the rules of EAAT. It should mainly pay attention to naming associations and think to double associations that can be instantiated in both directions. But the user must also be very careful about the direction of associations. If the direction of an association is not the same as a connection (instance of association) then it may generate errors.
Bibliography

Scientific papers

[1]: Scenario-based Architectural Decision Support within Enterprise Architectures (Concept and Implementation of a Prototype), Diploma Thesis (March 2009) written by Markus Buschle, Torsten Derlat and Daniel Feller. This document details the steps for creating, testing and validating the first versions of the EAAT tool (previously called EAT).

[2]: Architecture analysis of enterprise systems modifiability - Models, analysis and Validation from The journal of Systems and Software (Vol. 83, 2010, 1387-1403) written by the Robert Lagerström, Pontus Johnson and David Höök. This article detail how the modifiability monitoring is important for business and IT management and try to explain how PRMs can be used to perform such calculations.


[7]: Assessing Modifiability in Application Services Using Enterprise Architecture Models - A Case Study, from the proceedings of the TEAR workshop (2012), written by the PhD student Magnus Österlind and the Robert Lagerström and Peter Rosell. This article details how the modifiability monitoring is important for business and IT management and explains that the OCL is more adapted than PRMs to analyze the modifiability.

[9]: The report *The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry* (draft version 4, August 2009) written by Alan MacCormack. This document introduce for the first time ideas like VFI and VFO or “Core”.

[10]: The *Hidden Structure: Using Network Methods to Map System Architecture*, a working paper (draft, February 2013) written by the Carliss Baldwin, Alan MacCormack and John Rusnak. Detail a method which is the results of several years of study about the modularity into software projects.


[13]: *The Enterprise Architecture Analysis Tool – Support for the Predictive, Probabilistic Architecture Modeling Framework*, a working paper (August 2013) written by Markus Buschle, Pontus Johnson and Khurram Shahzad. This paper presents the proceedings of the Nineteenth Americas Conference on Information Systems. It describes last works done on EA and focuses on decisions made in the EAAT project.


[15]: *Visualizing and Measuring Software Portfolio Architecture: A Flexibility Analysis*, a working paper (June 2013), written by Robert Lagerström, Carliss Baldwin, Alan MacCormack and David Dreyfus. It gives some explanations for the use of the “HiddenStrucure” method for EA.
[16]: Automatic data collection for enterprise architecture models (Software & Systems Modeling), regular paper (May 2012), written by Hannes Holm, Markus Buschle, Robert Lagerström and Mathias Ekstedt. It describes a methodology to automatically generate a model.


[18]: A Tool for Enterprise Architecture Analysis Using the PRM Formalism from the Information Systems Evolution (108–121, 2010), written by Buschle M., Ullberg J., Franke U., Lagerström R., and Sommestad T.. This paper describes the first stages of development of EAAT.


Technical documents and Web sources

[a]: http://poi.apache.org/, it is the website of an Open Source project under the Apache License, Version 2.0. It is a project run by the Apache Software Foundation that provides pure Java libraries for reading and writing files in Microsoft Office formats, such as Word, PowerPoint and Excel.

[b]: http://community.jaspersoft.com/, it is the website of an Open Source project under the GNU GPL License. JasperReports is a Java reporting tool that can write to a variety of targets, such as: screen, a printer, into PDF, HTML, Microsoft Excel, RTF, ODT, Comma-separated values or XML files. It can be used in Java-enabled applications, including Java EE or web applications, to generate dynamic content. Jaspersoft provides commercial software around the JasperReports product, and negotiate contracts with
software developers that wish to embed the JasperReports engine into a closed source product.

[c]: http://www.jfree.org, JFreeChart is an open-source framework for the programming language Java, which allows the creation of a wide variety of both interactive and non-interactive charts. This project is under the GNU LGPL License.

[d]: http://www.eclipse.org/, Eclipse is a multi-language Integrated development environment (IDE) comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. It is the IDE of the EAAT project.

[e]: http://www.eclipse.org/home/categories/rcp.php is a project of Eclipse Foundation. It is an answer to the developer community need: reuse the source code of Eclipse (IDE) for rich client project. RCP means “Rich Client Platform”. It is the main framework used in the EAAT project.

[f]: http://www.eclipse.org/swt/, SWT is an open source widget toolkit for Java designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented. It is one of Eclipse RCP tools.

[g]: http://www.developpez.net/forums/ it is the main forum on which I asked my questions when I had difficulties. It is a recognized French forum for an active community that I thank for its help.
Table of illustrations

Figure 1: Class Modeler .............................................................................................................. 13
Figure 2: Object Modeler .......................................................................................................... 14
Figure 3: Input DSM in the BioPharma case .............................................................................. 16
Figure 4: Decision tree to determine the type of architecture .................................................. 18
Figure 5: Rearranged DSM in the BioPharma case .................................................................. 20
Figure 6: Illustration of a simple system .................................................................................. 21
Figure 7: Initial planning (GANTT 1/2) .................................................................................. 27
Figure 8: Initial planning (GANTT 2/2) .................................................................................. 27
Figure 9: Real planning (GANTT 1/2) ..................................................................................... 28
Figure 10: Real planning (GANTT 2/2) ................................................................................... 28
Figure 11: UML representation of the data structure in the prototype ..................................... 30
Figure 13: Theoretical case with a Core-Periphery architecture .............................................. 34
Figure 14: Screenshot of the "DSM" submenu ........................................................................ 42
Figure 15: Screenshot of the "Object Model Explorer" .............................................................. 44
Figure 16: New context menu in the “Object Model Explorer” ............................................... 45
Figure 17: New context menu in the “View Explorer” .............................................................. 45
Figure 18: DSM Results view ................................................................................................. 46
Figure 19: DSM Element view ............................................................................................... 47
Table of tables

Table 1: Definitions................................................................................................................................. 9
Table 2: Abbreviation ................................................................................................................................. 10
Table 3: Input DSM (explicit representation) .......................................................................................... 21
Table 4: Visibility matrix (explicit representation) ................................................................................. 21
Table 5: List of Elements (hidden structure classification) ..................................................................... 22
Table 6: Rearranged DSM (explicit representation) ............................................................................... 23
Table 7: Example of input table for the reading file function................................................................. 32
Table 8: Reading exceptions.................................................................................................................... 40
Appendices

Appendix A: Report sample
Appendix B: Enterprise Application Modularity (User Guide)
Appendix A: Report sample (BioPharma case, partial calculations: only the “ApplicationModule” class)

DSM analyze of: test2.iEaat (Classes: ApplicationModule)

General information

The propagation cost is 9.97% and the architecture flow is 31.58%.

<table>
<thead>
<tr>
<th>Classification</th>
<th>No. of</th>
<th>% of total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared</td>
<td>0</td>
<td>0.00 %</td>
</tr>
<tr>
<td>Core</td>
<td>3</td>
<td>15.79 %</td>
</tr>
<tr>
<td>Periphery</td>
<td>15</td>
<td>68.42 %</td>
</tr>
<tr>
<td>Control</td>
<td>3</td>
<td>15.79 %</td>
</tr>
</tbody>
</table>

The system has a “Core-periphery” architecture. The largest core contains 15.79% of EA elements (5.0% to have a sufficient size). The largest core is 1.50 times larger than the second one (1.5 to be considered predominant).
DSM analyze of: test2.iEaat (Classes: ApplicationModule)

Input matrix
Rearranged matrix

Analyze of each Element

There are 19 Elements

<table>
<thead>
<tr>
<th>Object name</th>
<th>Class name</th>
<th>DFI</th>
<th>DFO</th>
<th>VFI</th>
<th>VFO</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>689</td>
<td>ApplicationModule</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>CORE</td>
</tr>
<tr>
<td>686</td>
<td>ApplicationModule</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>CORE</td>
</tr>
<tr>
<td>685</td>
<td>ApplicationModule</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>CORE</td>
</tr>
<tr>
<td>461</td>
<td>ApplicationModule</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>645</td>
<td>ApplicationModule</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>PERIPHERY</td>
</tr>
</tbody>
</table>
### DSM analyze of: test2 ejac (Classes : ApplicationModule)

<table>
<thead>
<tr>
<th>Object name</th>
<th>Class name</th>
<th>DFI</th>
<th>DFO</th>
<th>VFI</th>
<th>VFO</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1204</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>644</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>448</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>410</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>515</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>564</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>712</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>1187</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>1190</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>1203</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>1197</td>
<td>ApplicationModule</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>PERIPHERY</td>
</tr>
<tr>
<td>1194</td>
<td>ApplicationModule</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>CONTROL</td>
</tr>
<tr>
<td>1195</td>
<td>ApplicationModule</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>CONTROL</td>
</tr>
<tr>
<td>1196</td>
<td>ApplicationModule</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>CONTROL</td>
</tr>
</tbody>
</table>

### Analyze of Clusters

There are : 2 Clusters :
- The Core has 3 elements :
  685, 689 and 686.
- There is a cluster of 2 elements :
  645 and 451.
Appendix B: Enterprise Application Modularity (User Guide)

General Instruction

For what could I use the Enterprise Application Modularity?
The Enterprise Application Modularity is a set of features to analyze the structure and modularity of an EA.

It is possible to use these functionalities on all meta-models. There are some constraints on the meta-model they are described in the last section.

How can I know if my EAAT is able to process to the hidden structure analyze?
If you have the "DSM" submenu that appears in the "File" menu of the "Object Modeler", that means that your version of EAAT is compatible with the hidden structure analysis.

How instantiate a model from an Excel Sheet (with the DSM format)
The type of the file
The file format must be XLSX.

Rules to have a readable file
- the DSM must be in the first spreadsheet;
- 2 headings of rows and columns:
  - the 1st contains names of the class (it is possible to merge some cells);
  - the 2nd contains names of the object;
- There is a symmetric order between rows and columns;
- The cell, after the last object name on the 2nd row (header of object names), must be empty;
- A cell contains the list of associations used, these are separated by a “ “ (a simple space);
- An association name can't be “0” or “1”, when an association cell contains “0” or “1”, this cell will be considered as empty;
- An association in a cell means that the object represented by the row of the cell depends on the object represented by the column of the cell.

Why does the software freeze?
The software cannot be used while it is reading a file. For example: for a file that contains around 500 elements, it takes 20 seconds to be read (this figure is not up-to-date, this time has been reduced).

How can I execute the calculation?

On the whole model
File → DSM → Calculate DSM
From a list of classes, objects or associations

In the Object Model Explorer (as follow), the user can select a list of classes or a list of objects or a list of connections.

It is important that all items of the selection have the same type (classes, objects or connections).

In the case of a selection of classes, calculations will be executed on all objects that instantiate at least one of the selected classes. In the case of a selection of objects, the calculations will be directly done on the selected objects. In the case of a selection of connections, the calculations will be performed on all objects that are connected by a connection that instantiates one of the associations supported by the selected connections. In the latter case, only the selected types of connections are considered as dependencies.

After selecting the desired list, and in order to start the calculations, the user must then press the right mouse button (to access the context menu) and activate the "DSM Calculation" action.

From a list of views

In the View Explorer (as follow), the user can select a list of views and use the "DSM Calculation" action.

Calculations will be performed on all objects that are included at least in one of the selected views.

To start the calculations, the user must then press the right mouse button (to access the context menu) and activate the "DSM Calculation" action.

Automatically generate a view

It is also possible to automatically generate views from the "Object Model Explorer".
The process is almost the same as the realization of the calculations for the "Object Model Explorer". The user must use the action "Create or update a view" instead of "DSM Calculation".

If a view has been created with exactly the same selection, this view will not be duplicated but the contained elements will simply be updated.

Why does the software freeze?
The software cannot be used while it executes calculations. As calculations take time, the user can have the feeling that the software has crashed. For example: for a system that contains around 500 elements, it takes 5 seconds to perform calculations.

Where can I see the results?

In the report
When the user performs calculations and at the end of these calculations, a dialog box opens to provide to the user to record results in a report. If the user does not want to create this report, he can cancel.

The report contains all the information calculated by the hidden structure method and it is illustrated by charts.

Why does the software freeze?
The creation of the report is an action that can be very long. The duration depends on the number of elements in the system. For example: for a system that contains around 500 elements, it takes 15 seconds to create the PDF report.

In the GUI
To consult results of an analysis, there are two views (GUI component).

"DSM Results" is the first view, it gives the general results of calculations like the propagation cost, the architecture flow and the two graphs (input DSM and rearranged DSM).
It is possible to copy the text (as in any editor) and it is also possible to right click on charts to access to their own context menu. This menu contains option to save, print and zoom on the chart.

The other view (GUI component) is called "DSM Element" and summarizes the most important information related to the selected object. The selection of the object can be performed from the "Object Model Explorer" and from a view (as defined in EAAT).

**Generate the charts**

It is possible to directly generate the charts from the GUI. To do this, the user can use two actions:

- Generate Input DSM
- Generate Rearranged DSM

These two actions are included in the DSM submenu.

**How configure the Enterprise Application Modularity**

It is possible to modify some parameters of the analysis Enterprise Application Modularity. The user can interact with these parameters through only one dialog. The user can access to this dialog via the action "Properties" contained in the "DSM" submenu.

Users should be aware that it is impossible to save the settings. Parameters always revert to their default values when the application is restarted.

Thanks to this dialog, the user can modify three options, these options are detailed below.

**Disable the list of elements in generated reports**

This option allows the user to remove the list of elements of the report generated by
the calculations. The default value include the list of element.

By disabling this option, the user can reduce the time of report generation.

It is possible that generating a complete report is not possible for too large models. It is advisable to remove the list of elements.

Increase the resolution of charts
DSM charts resolution is the second controllable parameter. Thus, the user can adjust the resolution of the graphics according to his needs.

These graphics are square therefore the user can only interact with a single value. This value must be an integer greater than 100 (this limit avoids having unreadable graphics).

This parameter controls the resolution of the graphics generated in reports and those generated as a PNG image file.

Manage the sorting algorithm of the input DSM
The last but not least, it is possible to change the sort method of the Input DSM. Thus, the user can arrange the matrix from the point of view of the architect. You can group objects by classes in the matrix and then order these classes.

To do this, the user can give a priority level to each class. All classes that have the same priority level are ordered alphabetically. The priority level is represented by an integer. Smaller is the integer, higher is its priority level (it therefore be in the upper left corner of the graph).

To change the priority level dedicated to a class, the user can use a special dialog view. This dialog is accessible from the "Properties" dialog by pressing the "Open Dialog" button.

In this view, each class is present. In the example on the right, the Application Components has a priority level of 2 when all other classes have a priority level of 1.
That means that all objects that instantiate the Application Components class in following studies will be pushed to the bottom right corner in the input DSM.

Some rules to have a proper meta-model

Basic rules
There is some simple rule that are required to respect to create a proper meta-model
- Two different associations must have different names;
- Two associations that rely different classes are considered as different, for example: let A, B and C be three classes A → B, B → C and B → A are three different associations;
- An association can be instantiated only once between the two same objects this implies that if the user cannot instantiate two connections of the same association on two objects (whatever the directions of connections). For example: let a1 and a2 be two instances of the class A, it is impossible to create the two connections a1 → a2 and a2 → a1. Indeed, they both represent the association A → A (or A “depends on” A) between the two same objects. If such connections between objects need to be created, the meta-model must contain two associations that represent the possible two connections.

Special attributes
There are some attributes with which the implementation of the hidden structure method can interact.
- DFI
- DFO
- VFI
- VFO
- Type
These attributes are not case dependent, so it is possible to write in capitals or lower case letters. The presence of attributes are not mandatory.

These attributes must be numeric. In the calculations, the value they carry is updated by the program. There is no attribute of type String, which is why the "Type" attribute is numeric (the value 1 is “Shared”, 2 is “Core”, 3 is “Periphery” and 4 is “Control”).